TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Major Project Mid-term Report

On

VQA Voyager: Voice-Based Visual Question Answering for Cultural Heritages in Kathmandu Valley

Submitted By:

Arnab Manandhar	(THA077BEI008)
Chandra Mohan Sah	(THA077BEI017)
Looza Subedy	(THA077BEI024)
Santosh Acharya	(THA077BEI040)

Submitted To:

Department of Electronics and Computer Engineering

Thapathali Campus

Kathmandu, Nepal

January, 2025

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

THAPATHALI CAMPUS

A Major Project Mid-term Report
On

VQA Voyager: Voice-Based Visual Question Answering for Cultural Heritages in Kathmandu Valley

Submitted By:

Arnab Manandhar	(THA077BEI008)
Chandra Mohan Sah	(THA077BEI017)
Looza Subedy	(THA077BEI024)
Santosh Acharya	(THA077BEI040)

Submitted To:

Department of Electronics and Computer Engineering
Thapathali Campus
Kathmandu, Nepal

In partial fulfillment for the award of the Bachelor's Degree in Electronics Communication and Information Engineering

Under the Supervision of

Er. Suramya Sharma Dahal

January, 2025

ACKNOWLEDGEMENT

We express our deepest gratitude to the Department of Electronics and Computer Engineering and the Institute of Engineering, Thapathali Campus, for providing us with an excellent platform and the opportunity to apply and enhance our skills. The support extended by this esteemed institution has been invaluable and inspiring to us.

We extend our heartfelt appreciation to Er. **Suramya Sharma Dahal** for supervising our research and to all the contributors of the published research papers and articles that have be crucial to us for the selection of this project.

We would also like to thank all the teachers and faculty members whose guidance and expertise have significantly contributed to our growth and learning, alongside our family and friends for their unwavering support and encouragement. Their constant assistance and motivation have provided us with the strength and confidence to pursue our goals.

Finally, we are grateful to everyone who, directly or indirectly, has played a role in our achievements. Your contributions have been a source of inspiration and have created a nurturing environment for our academic and personal development.

Arnab Manandhar	(THA077BEI008)
Chandra Mohan Sah	(THA077BEI017)
Looza Subedy	(THA077BEI024)
Santosh Acharya	(THA077BEI040)

ABSTRACT

This project aims to benefit people in identifying historical objects and artifacts in world heritage sites in Kathmandu valley and learning about their significance and history. In such a situation, the user can capture a real-time image of the objects and ask a question by speech input. The application uses the Visual Question Answering (VQA) tool to integrate image processing and natural language processing. The system is also capable of speech-to-text translation and vice versa which helps people, especially tourists to identify, recognize, and thus obtain details of any particular objects through voice output. The YOLOv8 object detection model with an F1 confidence score of 0.99 and mAP50 of 0.992, ensures accurate identification of objects in the captured image. Once an object is detected, the object name and the user's query are passed to the BART (Bidirectional and Autoregressive Transformer). The BART model, achieving a fuzzy match accuracy of 80%, semantic similarity of 86%, generates text-based answers, with a METEOR score of 79%, reflecting strong alignment with human-like responses. With this, the app is able to function based on the user's voice or text input and provide accurate answers to queries about the captured artifacts, eliminating most doubts regarding them and enhancing the overall user experience.

Keywords: BART, image processing, natural language processing, transformer, YOLOv8

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
BART	Bidirectional and Auto-Regressive Transformers
BERT-RG	Bidirectional Encoder Representations from Transformers – Reading Comprehension
BLSTM	Bidirectional Long Short-Term Memory
BNQC	Bounded Normalized Quantile Criterion
BQC	Bayesian Quantile Criterion
CLIP	Contrastive Language-Image Pre-Training
CMQEF	Context-aware Multi-Level Question Embedding Fusion
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CV	Computer Vision
FQC	Frequentist Quantile Criterion
FFNN	Feed Forward Neural Network
GLoVe	Global Vectors for Word Representation
GPT-3	Generative Pre-Trained Transformer 3
GRU	Gated Recurrent Unit
GTTS	Google Text-to-Speech
IaFM	Intra-question Fusion Method
IaNF	Intra-question Non-Linear Fusion Method
LLM	Large Language Model
LSTM	Long Short-Term Memory
MMD	Maximum Mean Discrepancy
NLP	Natural Language Processing
QA	Question Answering

QE	Quantile Estimation
R-CNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit
ViLT	Vision-and-Language Transformer

1. INTRODUCTION

Artificial intelligence (AI) is not only saving us time by automating mundane and tedious tasks, but it is also opening up a world of possibilities by enhancing our understanding and interaction with the world around us. Computer Vision (CV) and Natural Language Processing (NLP) methods have the potential to significantly enrich the experiences of tourists by helping them identify and learn about various objects and artifacts at world heritage sites. Regarding this, Visual Question Answering (VQA) is one of the most promising CV and NLP based tasks. In the most common form of Visual Question Answering, the model is presented with an image and a free-form open-ended textual question about this image, whose answer is correctly predicted by the model. This project aims to assist tourists in UNESCO World Heritage Sites within Kathmandu valley by using VQA to provide detailed information about different objects and artifacts, enhancing their overall experience and understanding.

1.1 Background

Tourism is a significant aspect of cultural exchange and economic development, with millions of individuals traveling to explore new places and learn about different cultures. World Heritage Sites, in particular, attract a large number of tourists, eager to discover historical and cultural artifacts. However, the challenge often lies in providing detailed and accessible information about these artifacts in an engaging manner.

In Kathmandu, a city rich with heritage sites, tourists may find it difficult to understand the significance of various objects and artifacts they encounter. Traditional methods like guidebooks or plaques may not provide the immersive experience that modern travelers seek. With the advancement of technology, there is an opportunity to enhance tourists' experiences by using Artificial Intelligence (AI).

Visual Question Answering (VQA) leverages Computer Vision (CV) and Natural Language Processing (NLP) to enable users to ask questions about what they see and receive informative responses. By integrating VQA into a mobile application, tourists can capture images of objects and artifacts at World Heritage Sites in Kathmandu and receive detailed answers to their questions. This project aims to bridge the information gap and provide a more interactive and educational experience for tourists, enriching their understanding and appreciation of cultural heritage.

1.2 Motivation

Despite significant technological advancements, there remains a substantial gap in providing tourists with interactive and comprehensive information about cultural artifacts and heritage sites. Traditional methods of delivering information, such as guidebooks, plaques, or guided tours, often fail to engage tourists fully or provide the depth of knowledge they seek and the tour guides are quite expensive as well.

Kathmandu, with its rich cultural heritage, is a prime example of where tourists can greatly benefit from advanced technology to enhance their experience. Imagine visiting a World Heritage Site and having the ability to instantly learn about any artifact or object through simple interaction with your smartphone. This capability not only enriches the tourist experience but also fosters a deeper appreciation and understanding of the cultural significance of these sites.

This project aims to bridge this information gap by introducing a mobile-based visual question-answering system tailored for tourists. By leveraging cutting-edge AI technologies in Computer Vision (CV) and Natural Language Processing (NLP), we aspire to provide tourists with an intuitive and engaging tool that allows them to capture images of artifacts and receive detailed answers to their questions. This not only enhances their knowledge but also makes their visit more interactive and memorable, contributing to a more enriched and informed exploration of World Heritage Sites in Kathmandu.

1.3 Problem Definition

The existing technological landscape falls short of meeting the intricate needs of tourists who seek a contextually rich understanding of cultural artifacts and heritage sites. While traditional tools like guidebooks, plaques, and audio guides offer some assistance, they often provide only surface-level information, missing crucial details that tourists rely on for a comprehensive understanding of their surroundings. Current methods may label objects but cannot describe the intricate historical and cultural relationships between artifacts, leaving tourists without the depth of information they require.

Furthermore, there is a noticeable absence of technology that facilitates natural language interaction, where users can engage in dialogue based on the scene captured, posing queries and receiving detailed responses tailored to their environment. This project aims to address these shortcomings by developing a solution that not only recognizes objects but also offers detailed, contextually relevant descriptions and supports natural language interaction. This will empower tourists to engage more fully with their surroundings, enhancing their knowledge and appreciation of the cultural heritage sites in Kathmandu, and fostering a richer and more interactive exploration experience.

1.4 Objectives

The main objectives of our project are listed below:

- To develop a Visual Question Answering (VQA) tool that answers questions based on the object in the captured image.

- To capture images and create a voice-based app that allows the user to ask questions about the image.

1.5 Scope and Applications

This project possesses substantial market value and offers a wide-reaching impact for tourists visiting heritage sites, particularly in Kathmandu. Many tourists lack in-depth knowledge of the historical and cultural significance of the artifacts and objects they encounter. This project aims to inspire and empower these individuals by providing them with comprehensive and engaging information, thereby enhancing their travel experience.

The primary objective of this project is to enhance various aspects of the tourist experience at world heritage sites. By offering real-time descriptions of their surroundings, the project aids tourists in understanding the historical and cultural context of the artifacts and objects they encounter. This feature significantly improves their ability to appreciate and learn about the sites they visit.

Additionally, the project facilitates a more interactive and personalized exploration by allowing tourists to pose questions and receive detailed responses about specific objects or scenes. This includes understanding the significance, history, and cultural relevance of artifacts, thus enabling a more meaningful and enriching experience.

Moreover, the project promotes greater engagement by assisting tourists in navigating heritage sites with ease and confidence. This includes identifying key landmarks, understanding site layouts, and accessing relevant information seamlessly through their mobile devices.

In summary, this project is designed to enhance the quality of the tourist experience by providing them with the tools necessary to navigate and understand their environment, interact with the cultural heritage more deeply, and gain valuable insights, ultimately making their visit more memorable and educational.

2. LITERATURE REVIEW

The Visual Question Answering (VQA) task was first proposed by Malinowski and Fritz [1]. It intersects Computer Vision (CV) and Natural Language Processing (NLP). The aim is to let VQA models correctly answer natural language questions according to images. Therefore, it is required to process and understand text and visual content and join the two modalities representation in a common feature space. With the breakthrough of deep learning in the fields of both CV and NLP, VQA has attracted the extensive attentions of researchers and gained remarkable research progress. In this Section, we provide an extended background of the modules typically belonging to a VQA model and a summary of the most relevant approaches adopted to solve the VQA task.

A model proposed in 2015 [2] introduced a new dataset and a CNN-LSTM-based methodology for Visual Question Answering. The dataset introduced comprises approximately 0.25 million images, 0.76 million questions, and 10 million answers, derived from the COCO Dataset. Each image has three associated questions, each answered by 10 subjects with their confidence levels. This dataset includes free-form and open-ended question answering, encompassing inquiries starting with "What," "How," "Why," and "Where." The study addresses open-ended QA and a multiple-choice task, where a multiple-choice task requires algorithms to select from a predefined list of possible answers. For visual questions and answering it utilizes a combination of VGGNet and LSTM. VGGNet is used for extracting features from the image while LSTM is used for text encoding, and the image and textual encodings obtained are merged via element-wise multiplication, yielding an accuracy of 54.06%.

RNN is implemented with [3] LSTM to handle inputs(questions) and outputs(answers) of variable size in a model proposed in 2015. Questions and Image Feature (Produced by CNN pre- trained for object Recognition) are first fed to the encoder LSTM. Encoder processes input data and converts it into a fixed size summary called feature vector. Output from the encoder is passed to decoder which produces variable-length answers, one word per recurrent iteration. At each iteration the last predicted word is fed through a recurrent loop into LSTM until a special <END> symbol is predicted.

Another model proposed in 2015 [4] used two sets of image features as input. This feature of the image is fed into LSTM at two different points (at the start and at end of question sentence). The Bidirectional LSTM scans the question in both Forward and Backward directions. This bidirectional approach allowed the model to capture relationships and dependencies between words easily. Due to BLSTM it had better performance than the "Neural-Image-QA" model.

In another model proposed in 2016 [5], weights of certain layers in CNN are not fixed. Instead, they are dynamically adjusted based on specific questions being asked about the image. For such adaptive parameter prediction, they have employed a separate parameter prediction network which consists of gated recurrent units (GRUs, a variant of LSTMs), taking a question as input and producing candidate weights through a fully-connected layer at its output.This approach significantly improved in accuracy than the previous models.

A model was proposed that uses stacked attention networks (SANs) which learn to answer natural language questions from images using semantic representation of a question as query to search for the regions in an image that are related to the answer [6]. A multiple-layer SAN was developed in which query of an image was done multiple times to infer the answer progressively. The SAN consists of three major components: (1) the image model, which uses a CNN to extract high level image representations, e.g. one vector for each region of the image; (2) the question model, which uses a CNN or a LSTM to extract a semantic vector of the question and (3) the stacked attention model, which locates, via multi-step reasoning, the image regions that are relevant to the question for answer prediction. Experiments conducted demonstrated that the proposed SANs significantly outperformed previous similar approaches.

A novel combined bottom-up and top-down visual attention mechanism was proposed for more accurate captioning of images [7]. This approach enables attention to be calculated more naturally at the level of objects and other salient regions that are likely to be important. Within the approach, the bottom-up mechanism (based on Faster R-CNN) proposes image regions, each with an associated feature vector and the proposed captioning model uses a 'soft' top-down attention mechanism to weight each feature during caption generation, using the existing partial output sequence as context.

Another model, Using Deep Learning to Answer Visual Questions from Blind People was proposed [8] that addressed the many limitations of VizWiz dataset like high uncertainty of answers, the conversational aspect of questions, the relatively small size of the datasets. Data science pre-processing techniques were used to improve the VQA task. Uncertainty of each answer was computed using pre-processing step, a new pre-processing procedure was implemented which was able to augment the training set and almost double its data points by computing the cosine similarity between answers representation. An alternative question pre-processing pipeline was used in which conversational terms were removed. This boosted the accuracy of the model.

This paper [9] adopts a novel approach by employing BERT-RG, which delicately integrates pre-trained models to leverage the interaction between residual and global features in images and linguistic features in questions. Additionally, it incorporates a stacked attention mechanism to capture the relation between textual and regional features of an image, outperforming existing VizWiz VQA datasets specifically in yes/no questions. By focusing solely on yes/no inquiries, it employs multimodal strategies to exploit the relationship between text and images efficiently. It uses the BERT Model based on transformer architecture to extract the question encodings while the Image features are extracted using Resnet and VGG. The major contribution of this paper compared to previous ones is the stacked attention mechanisms which are employed to capture relationships between partial regional image features and textual keywords.

Another paper published in 2019 [10]introduces a method for multi-modal adaptation in open-ended visual question-answering tasks, achieving domain-invariant and task-discriminative feature embeddings. The framework includes a question encoder utilizing pre-trained GloVe embedding and a GRU cell, an image encoder based on Visual Genome dataset features, a multi-modal fusion module, and a classification module. Additionally, maximum mean distance (MMD) is used to minimize domain distribution mismatch across modalities. Experiments on sVQA and VizWiz datasets show promising results, with the highest accuracy of 70.06% achieved through data augmentation.

The described VQA system [11] leverages a CNN-LSTM model to answer open-ended, classification, counting, and yes-no questions by separately computing and later combining image and language features. Image features are extracted using a VGG16 CNN model trained on ImageNet, while language features are modeled using a Keras LSTM with GloVe word embeddings. The approach employs late fusion to integrate image and word embeddings, processed through a multilayer perceptron for result generation. GTTS is used for converting speech to text, and Kivy is utilized to develop a cross-platform application. This method shows an accuracy of 57%, which indicates further space for improvements, and future enhancements include the integration of proximity sensors for improved guidance.

The integration of Large Language Models (LLMs) like GPT-3 with visual question-answering (VQA) systems, exemplified by Hu et.al [12], significantly advances the field by enabling more accurate and contextually relevant responses. PROMPTCAP generates question-aware captions that incorporate crucial visual details, which, when combined with user queries, allow LLMs to produce precise answers. This capability is especially beneficial for follow-up questions requiring external knowledge and reasoning, such as a visually impaired person asking, “What food items are present in the image?” followed by, “What can I prepare with these food items?” While the initial question can be answered by the visual captioning model alone, the integration with LLMs like GPT-3 is essential for addressing more complex, context-dependent queries. This approach sets a new standard in knowledge-based VQA, making it a valuable tool for enhancing accessibility and usability for visually impaired individuals and other users.

Question models that are generally used for constructing Visual Question Answering (VQA) models attempt to exploit word context to extract multi-level concepts for modeling multi-level questions. However, they still have many defects leading to poor modeling of multi-level questions. To address these issues, a novel low-complex multi-level contextual question model, termed Context-aware Multi-Level Question Embedding Fusion (CMQEF) was proposed in 2023 [13]. It included mainly five modules: In QE module, the pre-training GloVe was used to encode each question into word embedding vectors, which were used as the input layer of IaF module and IaNF module. In IaF module, Intra-question Fusion Method (IaFM) was utilized to fuse

Forward Question Component (FQC) and Backward Question Component (BQC) for extracting parameter unshared low-level concepts at the word level. In IaNF module, Intra-question Nonlinear Fusion Method (IaNFM) was exploited to fuse Forward Nonlinear Question Component (FNQC) and Backward Nonlinear Question Component (BNQC) for extracting parameter-unshared high level concepts at the word level. In IrF module, two question modules were defined to extract low-level concepts and high-level concepts, respectively, and then Inter-question Fusion Method (IrFM) was adopted to fuse these two question modules for extracting parameter unshared multilevel concepts at the question level. InIaIrF method, IaFM, IaNFM and IrFM were connected to fuse IaF module, IaN Fmodule and IrF module for extracting parameter-shared multi-level concepts at the intra-and inter-question level.

In another study conducted in 2024 [14], Visual Mind focuses on Visual Question Answering (VQA) using the CLIP Model, augmented with additional layers. By leveraging the VizWiz dataset, the researchers employ the CLIP model with an extra linear layer, which is different from CNN-LSTM- based methods that relied on element-wise products for image and text-based features. CLIP, known for its Contrastive Language-Image Pre-training, serves as a versatile, multimodal, zero-shot model. The researchers utilize both image and text encoders of the CLIP model, concatenating features and passing them through additional linear layers with layer normalization and high dropout values to effectively capture multimodal representations. Notably, they solely train additional linear classifiers while keeping the pre-trained CLIP model weights frozen. Vision Transformer architectures outperform ResNet variants of the CLIP models in terms of VQA Accuracy, achieving test accuracy of 65.76%.

The paper "ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision" by Wonjae Kim, Bokyung Son, and Ildoo Kim presents ViLT [15], a model optimized for vision and language tasks without the need for convolutional networks or region-based supervision. ViLT is notably more efficient than CLIP due to its smaller model size, which makes it easier to train and test. This efficiency stems from its streamlined transformer architecture that directly processes raw image patches and text tokens, reducing the computational load. In contrast, CLIP is designed for zero-shot classification tasks and requires extensive computational resources for training due to its large-scale dual-encoder architecture that processes images and text separately before aligning them in a joint embedding space. This architectural difference makes CLIP computationally expensive to train, whereas ViLT's design focuses on simplicity and efficiency.

Most existing Visual Question Answering (VQA) systems treat VQA as a multi-class classification problem, which limits their ability to generate descriptive answers. In reference to the BART (Bidirectional and Auto-Regressive Transformers) paper by Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer [16], BART’s decoder is used to frame VQA as a generative answering task. BART, which is a denoising autoencoder for sequence-to-sequence models, combines a bidirectional encoder with an autoregressive decoder, making it highly effective at generating coherent and contextually appropriate text. By utilizing BART’s decoder, VQA systems can produce more detailed and contextually rich answers, thus overcoming the limitations of classification-based approaches and enhancing the system’s capability to provide generative and descriptive answers.

The paper Enhancing Real-time Object Detection with YOLO Algorithm by Sagar Dhanraj Pande and Gudala Lavanya [17] highlights the evolution of algorithms that enable real-time performance in critical applications such as video surveillance, computer vision, autonomous driving, and robotics. Among these, the YOLO (You Only Look Once) algorithm stands out as a highly regarded approach due to its ability to perform object detection in a single pass through the neural network, making it both efficient and effective. Unlike traditional methods that often treat object detection as a classification problem followed by localization, YOLO reframes it as a regression problem, predicting bounding boxes and class probabilities directly from full images in one evaluation. The algorithm’s innovative architecture, which leverages convolutional neural networks (CNNs), has paved the way for significant advancements in real-time detection tasks, leading to widespread adoption across various domains. Existing research consistently emphasizes YOLO's structured approach to object detection, noting its impact on the field through its variations and implementation strategies that enhance its accuracy and speed in real-world scenarios.

The paper [18], published about YOLOv9 highlights that this version incorporates Programmable Gradient Information (PGI) to address information loss during the feedforward process, ensuring accurate gradient updates. The model also integrates the Generalized Efficient Layer Aggregation Network (GELAN) for optimized lightweight models through gradient path planning, enhancing both accuracy and efficiency. Another paper [19], explains about YOLOv10 which advances the performance-efficiency boundary of YOLO models by eliminating the need for non-maximum suppression (NMS) in post-processing, thereby reducing inference latency. The model employs a holistic efficiency-accuracy driven design strategy, optimizing various components to reduce computational overhead while enhancing capability. Extensive experiments demonstrate that YOLOv10 achieves state-of-the-art performance and efficiency across various model scales. The YOLOv11 model demonstrates significant improvements in detection accuracy, missed detections, and false detections for both single and multiple power equipment object scenarios, indicating its substantial application value in power equipment target detection.

We used YOLOv8 for our project because it offers an excellent balance between accuracy and computational efficiency, making it suitable for object detection on resource-constrained devices like smartphones. While newer versions like YOLOv9, YOLOv10, and YOLOv11 provide incremental improvements, their advanced features often require more computational power and are often used for real-time object detection, which may not align with the needs of our application since we normally pass the captured images for object detection rather than using the model real-time. YOLOv8's proven reliability and ease of integration made it the ideal choice for our project.

3. REQUIREMENT ANALYSIS

3.1 Project Requirement

The Requirements for our project can be divided into Hardware requirements and Software requirements.

3.1.1 Hardware Requirements

For optimal performance of the software project, minimal hardware prerequisites are necessary. However, a vital component needed is a Graphics Processing Unit (GPU). This is essential for the efficient execution of the Machine Learning model, ensuring swift and effective processing.

3.1.1.1 Graphics Processing Unit

GPUs, or Graphics Processing Units, have become indispensable in the realm of machine learning (ML). Their parallel processing prowess accelerates tasks like model training and inference, significantly reducing computation time. By harnessing Google Colab, Kaggle's free GPU resources alongside the powerful NVIDIA RTX 2060, we propose to expedite our model development and training processes. This acceleration facilitates faster iterations and more efficient experimentation, ultimately enhancing the performance and capabilities of our voice-based Visual Question Answering system.

3.1.2 Software Requirements

The following are the software requirements of our project.

3.1.2.1 Python

Python is a popular, user-friendly programming language known for its simplicity and extensive libraries. We chose it as the main language for our model because of its flexibility and strong support for machine learning frameworks like TensorFlow and PyTorch. With Python, we can efficiently develop our voice-based visual question-answering system, benefiting from its clean syntax, rich libraries, and supportive community.

3.1.2.2 Google Colab

Google Colab, or Google Colaboratory, is a cloud-based platform by Google providing free access to computational resources for Python code execution. Widely used in machine learning and data science, it offers an interactive environment where users can write and run code in Jupyter notebooks hosted in the cloud, eliminating the need for local installations and allowing collaborative work. In our voice-based Visual Question Answering (VQA) project, Google Colab serves as the central hub for model development and experimentation, providing access to powerful GPUs for training deep learning models. We propose to leverage its capabilities to process spoken queries, generate responses, and integrate libraries like GTTS for converting text to speech, all within the Colab environment.

3.1.2.3 Tensorflow

TensorFlow, a powerful machine learning framework, enables the creation of voice-based Visual Question Answering (VQA) systems. By combining image processing and natural language understanding capabilities, developers can build sophisticated systems that analyze visual content and respond to spoken queries effectively. With TensorFlow's efficient model training and deployment options, these systems can be optimized and deployed across various platforms for widespread accessibility.

3.1.2.4 Pytorch

PyTorch is a widely-used open-source deep learning framework known for its flexibility, dynamic computational graph, and ease of use, making it ideal for research and development. We plan to use PyTorch to train and evaluate our Visual Question Answering (VQA) model by defining the model architecture, loading and preprocessing data, and implementing the training loop with automatic differentiation and optimization. This approach will enable us to develop a VQA model that effectively integrates image and text processing.

3.1.2.5 Numpy

NumPy is a fundamental library for scientific computing in Python, renowned for its support of large, multi-dimensional arrays and matrices, along with a vast collection of mathematical functions to operate on these arrays efficiently. We plan to use NumPy to handle data manipulation and preprocessing tasks for our Visual Question Answering (VQA) model. This includes operations such as normalizing image data, processing text input, and performing essential numerical computations that helps in model training and evaluation. NumPy's efficiency and ease of integration with other libraries will help streamline these processes, ensuring that our data pipeline is both robust and performant.

3.1.2.6 Flutter

Flutter is an open-source UI software development kit (SDK) created by Google for building natively compiled applications for mobile, web, and desktop from a single codebase. It is particularly known for its fast development process, expressive and flexible UI, and native performance. Flutter uses the Dart programming language, which enables hot reload, allowing developers to see the effects of their code changes in real time without restarting the app. This leads to a more efficient development process and quicker iteration. Flutter's widget-based architecture makes it easy to create complex UIs with customizable, reusable components, and it comes with a rich set of pre-designed widgets that adhere to both Material Design and Cupertino (iOS) standards. Furthermore, Flutter's growing ecosystem of packages and plugins simplifies integrating various functionalities like state management, animations, and network handling, making it a popular choice for developers looking to build high-quality, cross-platform applications. By using Flutter, we can design and deploy a user-friendly mobile app that allows users to interact with the VQA system seamlessly, providing spoken queries and receiving audio responses.

3.2 Feasibility Study

The feasibility analysis of our project can be divided into the following aspects.

3.2.1 Technical Feasibility

Developing a voice-based visual question-answering (VQA) system for identifying and learning about historical objects and artifacts at world heritage sites is technically feasible due to advancements in several key areas. The use of YOLOv8 for object detection provides high accuracy in identifying and localizing objects within images, with precise bounding box extraction. This capability is complemented by the BART encoder, which effectively handles the object name and the users' queries. BART's transformer-based architecture enables the system to transform the name of the detected objects and the user queries into suitable form for usage by the BART decoder. For text generation, BART decoder's proficiency in producing coherent and contextually accurate responses ensures that the answers are informative and well-structured.

Furthermore, the integration of text-to-speech (TTS) and speech-to-text (STT) technologies supports intuitive voice-based interactions, allowing users to capture images, ask questions verbally, and receive spoken responses. Combining these technologies—object detection with YOLOv8, VQA comprising response generation with BART, and voice interaction through TTS and STT—creates a cohesive system capable of delivering accurate and context-aware information in real time. Supported by extensive documentation and community resources, this integration promises to significantly enhance the accessibility and educational experience for users exploring historical sites, making the application both user-friendly and effective.

3.2.2 Economic Feasibility

The economic feasibility of developing a voice-based visual question-answering (VQA) system for identifying and learning about historical objects is significantly enhanced by utilizing open-source tools and freely available datasets, making a customized dataset for this project. By employing open-source models like YOLOv8 for object detection and BART for vision-language understanding, we avoid the high costs of proprietary solutions. Additionally, leveraging accessible text-to-speech (TTS) and speech-to-text (STT) technologies further reduces development expenses. This strategic use of open-source resources and technologies ensures that the project remains cost-effective and sustainable, aligning with our objective of creating a user-friendly VQA system.

4. SYSTEM ARCHITECTURE AND METHODOLOGY

4.1 System Block Diagram

The diagrammatical representation of our system is given below:

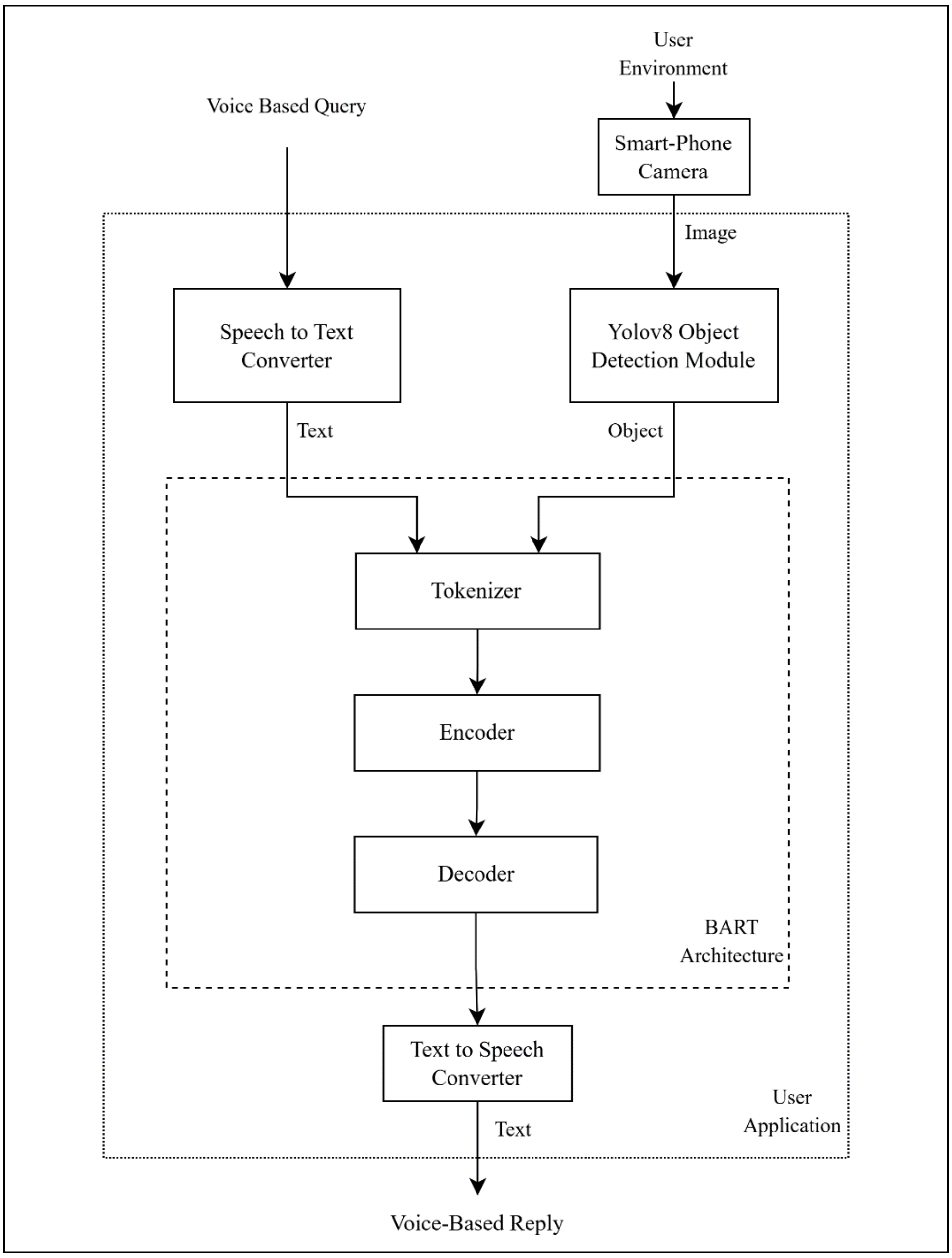


Figure 4-1: System Block Diagram

The system allows users to ask questions about cultural heritages and artifacts through voice commands. This involves several steps, including speech recognition of voice input, image capture, object detection, visual-question answering, and text-to-speech conversion.

4.2 Description of Working Principle

Initially, the user opens the app and captures the images of the cultural heritage or the artifact they want to know about. Then, the user asks a voice-based question about the captured image, voice input is then processed by speech-to-text functionality, which converts the spoken question into text form. The question is passed to the chatbot integrated with the app which holds a conversation with the user. The chatbot relies on a sophisticated backend model to generate informative and relevant responses.

The backend model comprises several key components. YOLOv8 is used to detect objects within the image and generate bounding boxes around them, effectively isolating the areas of interest and identifying them. The object name and the image post-bounding box generation are displayed to the user through the chat interface. The user inputs a query regarding the detected object. BART tokenizer receives the object name and query input by the user and converts them into manageable tokens. These text tokens are processed by the encoder in BART, converted into a sequence of embeddings and passed to the BART decoder, which generates an answer in a token-by-token manner for the given image and question pair.

Finally, the refined answer is converted into a voice-based response using text-to-speech functionality, providing the user with a clear and descriptive reply to their asked queries.

4.3 Data Pre-processing Pipeline

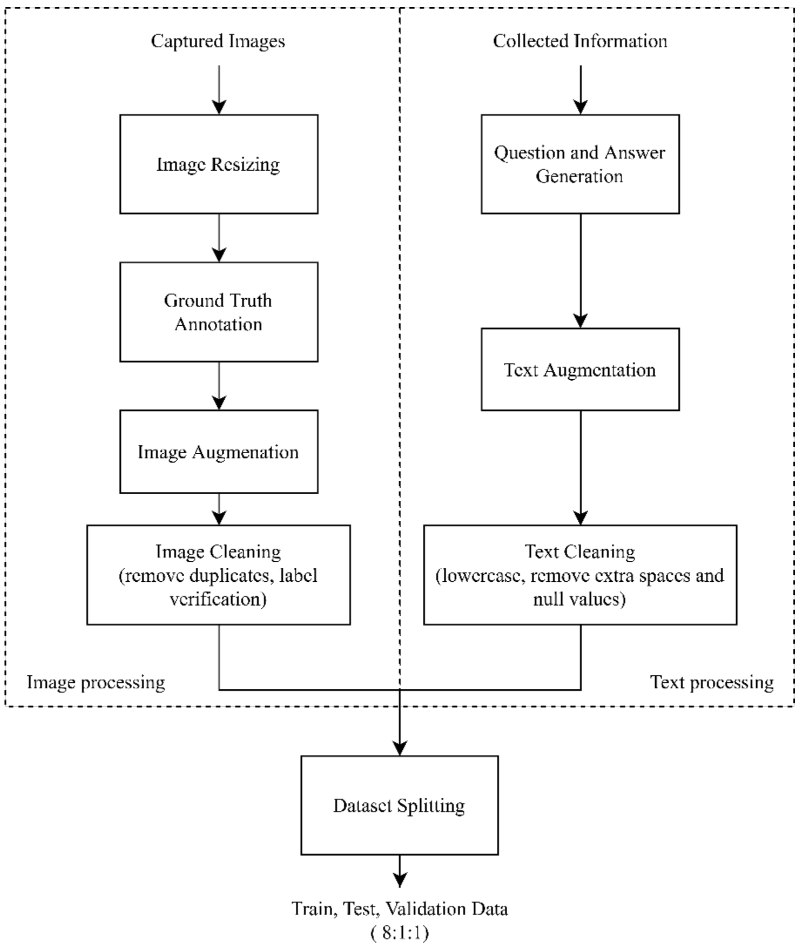

Captured Images
Image Resizing
Ground Truth Annotation
Image Augmenation
Image Cleaning (remove duplicates, label verification)
Image processing
Collected Information
Question and Answer Generation
Text Augmentation
Text Cleaning (lowercase, remove extra spaces and null values)
Text processing
Dataset Splitting
Train, Test, Validation Data (8:1:1)

Figure 4-2: Data Pre-processing Pipeline

The image processing pipeline begins with image resizing, where captured images are scaled to a consistent size, ensuring uniform input dimensions for subsequent tasks and model compatibility. Following this, ground truth annotation is performed, where the images are labeled with accurate information (e.g., bounding boxes, categories, or classes) to serve as reference data for training. Next, image augmentation introduces variations to the dataset by applying transformations like rotation, flipping, or brightness adjustment, which helps improve the model's robustness and generalizability. Finally, image cleaning is carried out to refine the dataset by removing duplicate entries and verifying labels for accuracy, ensuring that the data is clean, reliable, and free of inconsistencies.

In the text processing pipeline, the workflow starts with question and answer generation, where meaningful questions and corresponding answers are derived from the collected information, creating a foundation for the dataset. Then, text augmentation enriches the dataset by generating variations in wording or phrasing, increasing diversity and enabling the model to handle different textual contexts effectively. Subsequently, text cleaning ensures data quality by standardizing the text (e.g., converting it to lowercase), removing unnecessary white spaces, and handling null values to eliminate inconsistencies or noise. These steps collectively prepare the text data for training, ensuring its completeness and usability.

Once both the image and text processing pipelines are complete, the datasets are split into training, validation, and testing sets in an 8:1:1 ratio. This ensures a balanced distribution of data, enabling robust training, validation, and evaluation of the model.

4.4 System Flowchart

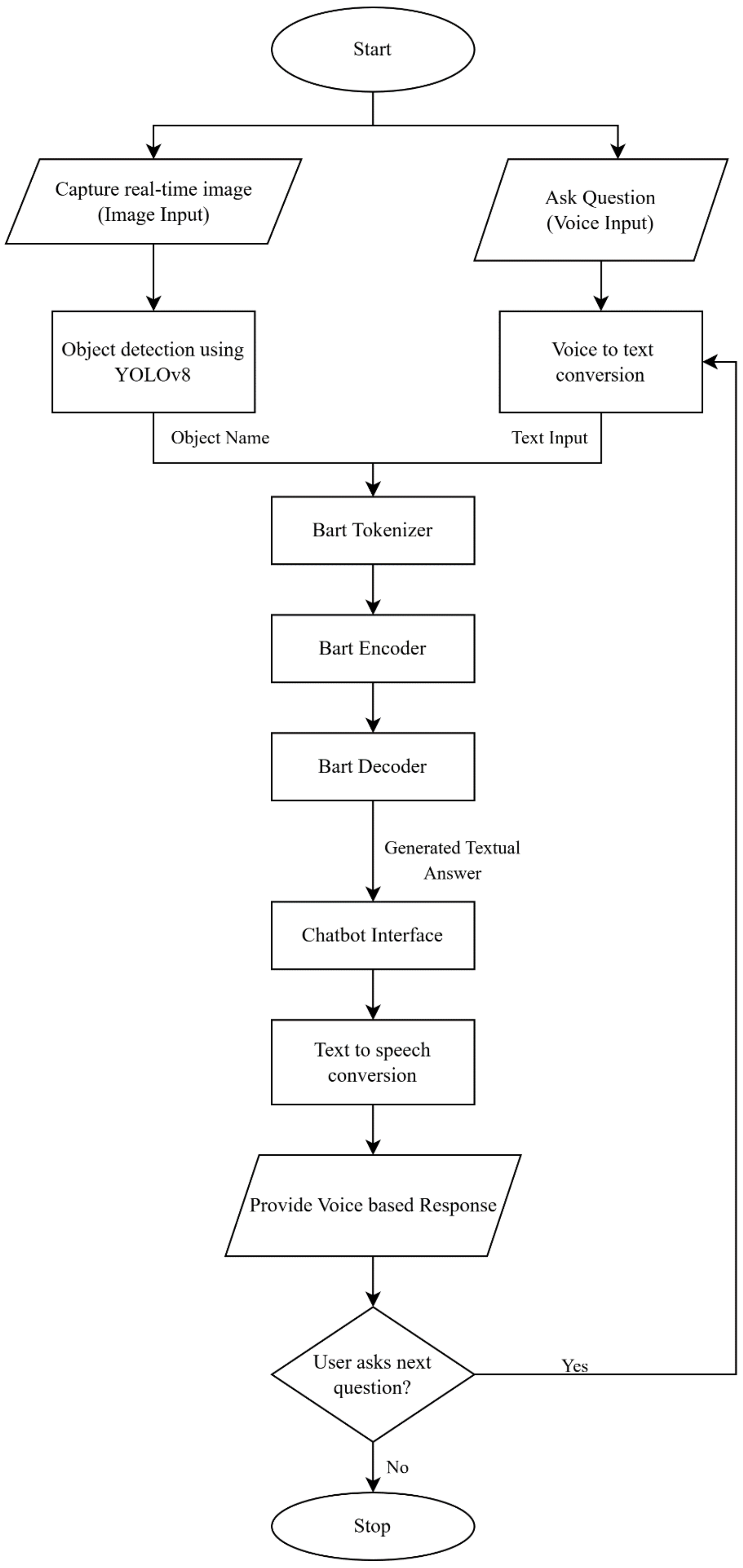


Figure 4-3: System Flowchart

This flowchart outlines the process for handling a single image and its corresponding questions using YOLOv8 for object detection and BART for vision-language understanding to handle real-time image and voice inputs seamlessly. Initially, the process captures a real-time image, which is processed through YOLOv8 for object detection. The object name identified by YOLOv8 is then extracted and used for further processing. Simultaneously, the user's voice input is converted to text using speech-to-text system, and the text is tokenized and embedded. Both the object name and the user query are fed into the BART encoder, which processes the combined input and generates embeddings. The BART decoder then generates a textual answer based on the combined information from the object name and the user query. The answer is delivered through a chat interface and converted to speech. The chatbot interface facilitates ongoing interaction, allowing the user to continuously ask questions about the identified object and receive answers until they decide to stop.

4.5 System Use Case Diagram

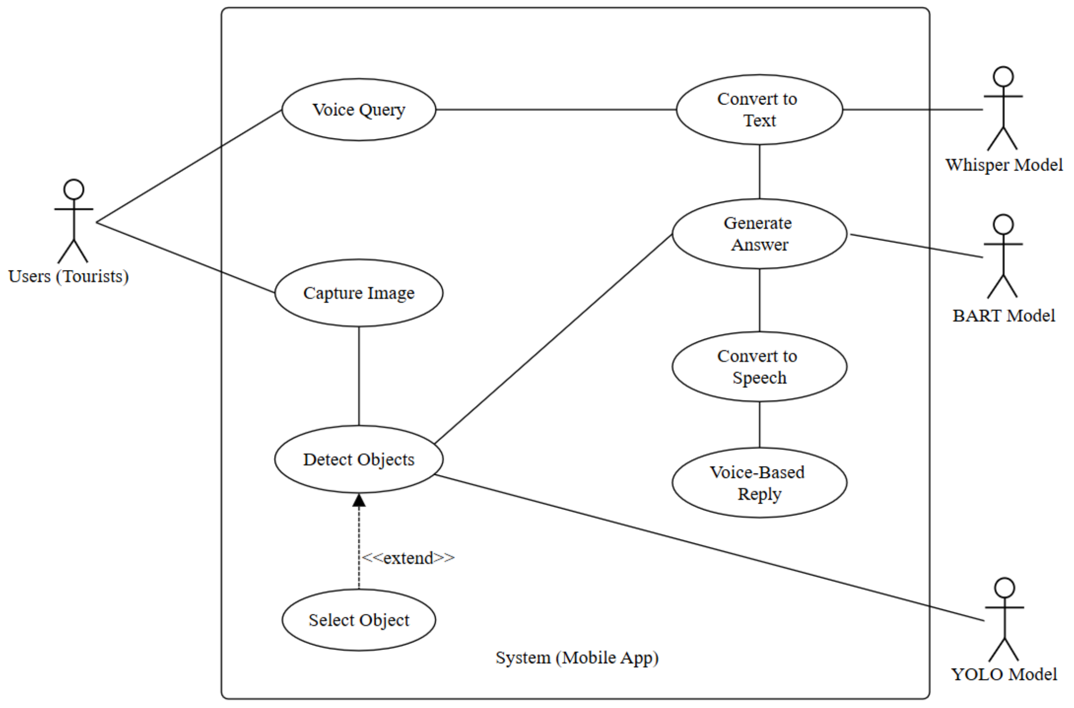


Figure 4-4: System Use Case Diagram

The use case diagram illustrates the interaction between the user and the mobile application system, supported by various machine learning models. The system enables users to either capture images or provide voice queries. Images are processed using the YOLO model for object detection, while voice queries are converted to text using the Whisper model. Both inputs are utilized to generate contextually accurate answers through the BART model, which are then conveyed back to the user via voice-based replies or displayed as text.

4.6 System State Diagram

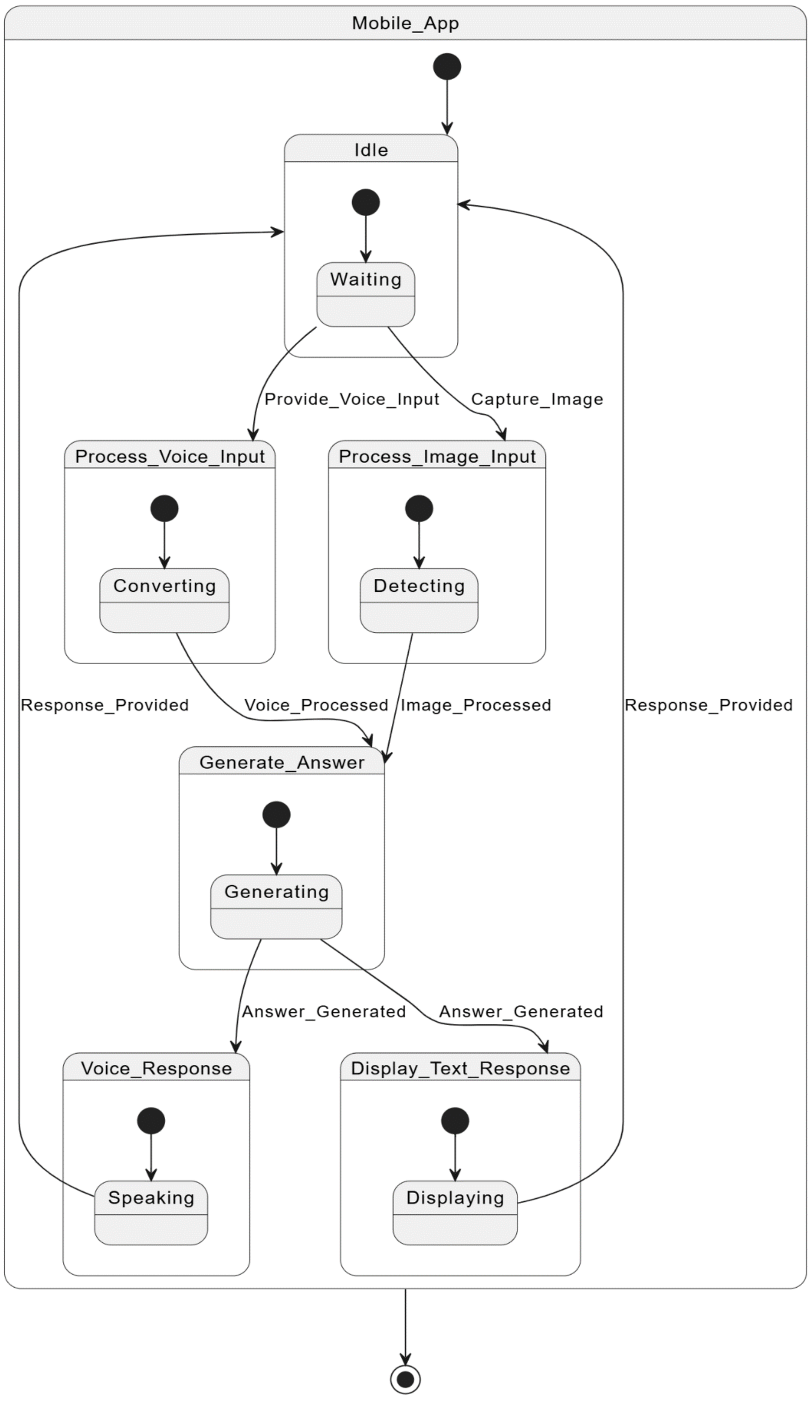


Figure 4-5: System State Diagram

Above state diagram illustrates the various states and transitions of a mobile application designed for voice and image-based input processing. The system begins in an idle state, awaiting user interaction. Upon receiving input, it transitions to either processing voice input or processing image input, depending on the user action. Voice inputs undergo a converting state, while image inputs proceed to a detecting state. Both pathways converge in the "Generate Answer" state, where the appropriate response is generated. The response is then provided either as a voice response (speaking) or a text response (displaying) before returning to the idle state.

4.7 YOLOv8 Architecture

YOLOv8, continues the tradition of single-stage object detectors that offer real-time performance and high accuracy. Designed to process an input image in one go, YOLOv8 predicts bounding boxes and class probabilities for multiple objects simultaneously. The overview of the architecture of YOLOv8 is given below:

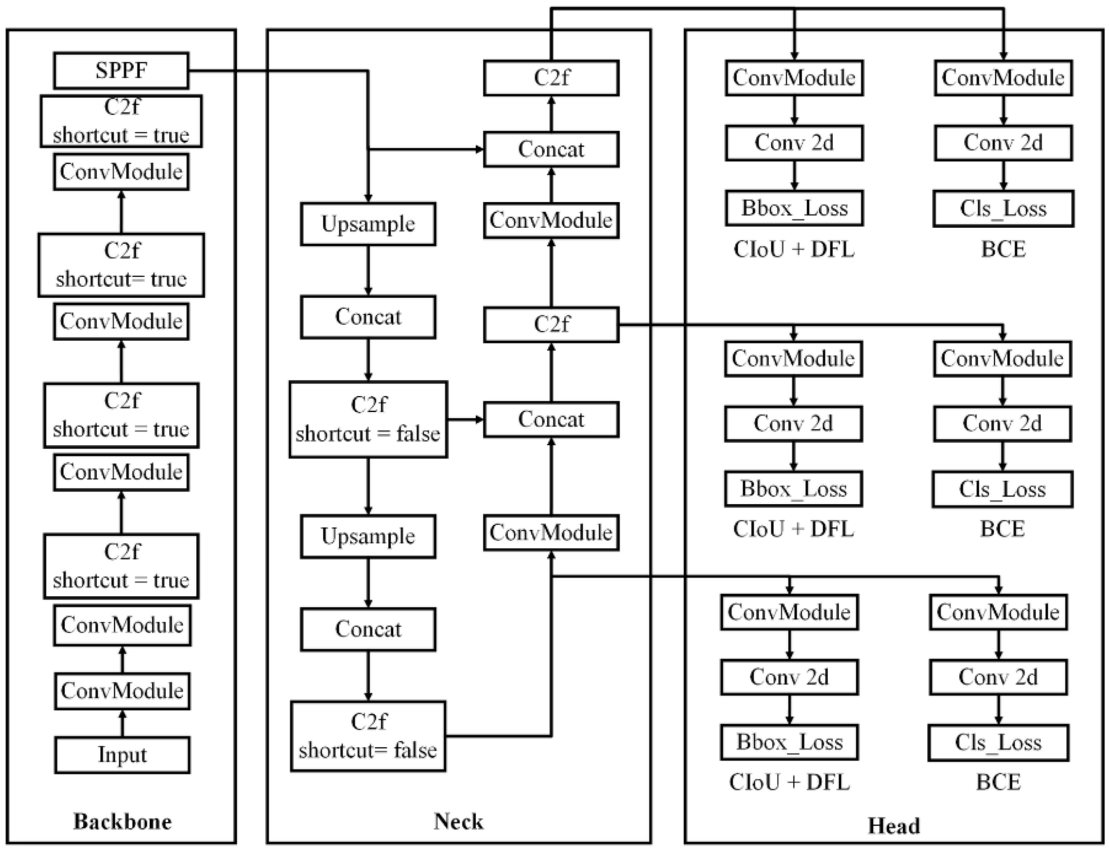


Figure 4-6: YOLOv8 Architecture

The YOLOv8 architecture is structured into three primary components: the backbone, the neck, and the head. The backbone, a pre-trained Convolutional Neural Network

(CNN), extracts feature maps at various levels of abstraction—low, medium, and high—from the input image. These feature maps are then integrated using the neck using path aggregation techniques. Finally, the head processes these combined features to classify objects and predict their bounding boxes.

4.7.1 YOLOv8 Blocks

The different blocks used in YOLOv8 architecture are explained below along with their internal architecture:

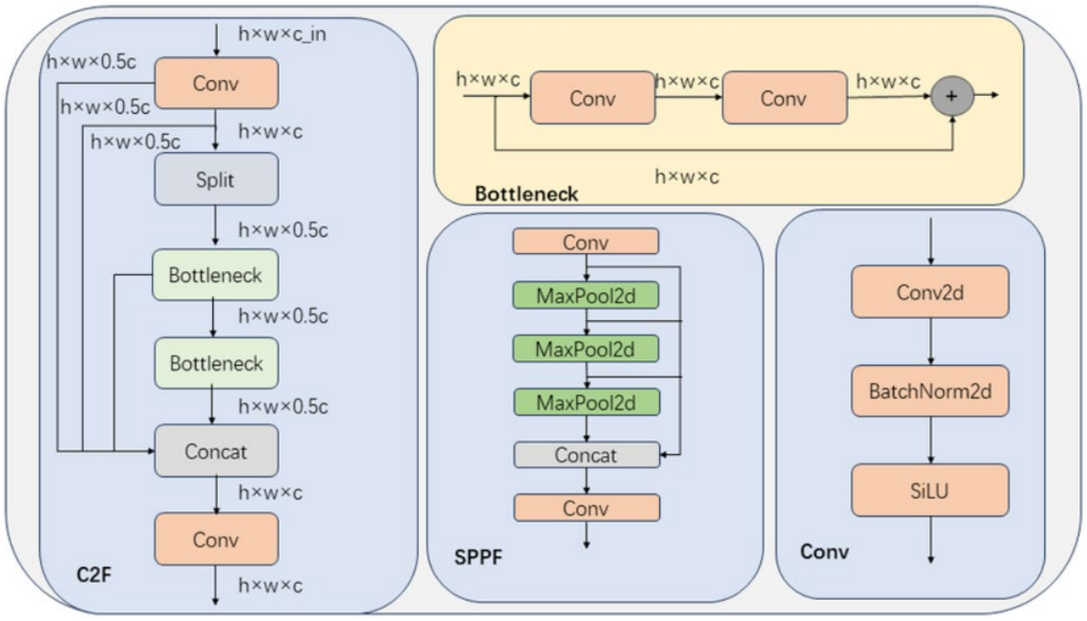


Figure 4-7: YOLOv8 Blocks

Convolutional Block

Convolution is a mathematical operation where a small matrix (kernel or filter) slides over input data, performing element-wise multiplication and summing the results to create a feature map. The "2D" in Conv2D indicates that this operation occurs in two spatial dimensions (height and width).

- k: Number of filters or kernels, determining the depth of the output volume, with each filter detecting different features.
- s: Stride, the step size for sliding the filter over the input. A larger stride reduces the spatial dimensions of the output.

- p: Padding, adding zeros to the input's borders to preserve spatial information and control output dimensions.
- c: Number of input channels, such as 3 for RGB images.

Batch Normalization (BatchNorm2d) normalizes 2D inputs from convolutional layers to stabilize and speed up training by ensuring values remain within a suitable range. The conv2D used SiLU activation function, which stand for Sigmoid Linear Unit. The SiLU activation function allows smoother gradients which helps with avoiding issues such as vanishing gradients during the training process.

Bottleneck Block

The Conv Block with a shortcut connection makes up the bottleneck block. A direct connection that skips one or more network layers is called a shortcut connection, sometimes referred to as a skip connection or residual connection. The shortcut is applied in the bottleneck block if the shortcut value is true; otherwise, the input is routed through two Conv Blocks in succession.

C2f Block

After performing a convolutional operation to create a feature map, the C2f block splits the feature map into two distinct routes. The feature map can be sent via one of two paths: either directly to a Concat block or via a bottleneck block. The depth_multiple parameter in the model determines how many bottleneck blocks are used in this operation. Ultimately, a final convolutional block receives the feature maps from the Bottleneck block and the split feature map after they have been concatenated together.

Feature map extraction is a critical process in the analysis and interpretation of convolutional neural networks (CNNs). Feature maps, also known as activation maps, are the outputs generated by convolutional layers as they process input data. Each feature map highlights different aspects or patterns detected in the input image, such as edges, textures, or more complex features like shapes and objects, depending on the depth of the network. By visualizing these feature maps, we can gain insights into how a CNN "sees" and processes data at different stages of its architecture.

The importance of feature map extraction lies in its ability to provide transparency into the otherwise black-box nature of deep learning models. By examining the feature maps, researchers and practitioners can better understand what specific features the network is focusing on at each layer. This can be crucial for diagnosing issues related to model performance, such as overfitting, underfitting, or the model's tendency to focus on irrelevant features. It also aids in model interpretability, making it easier to explain the network's decisions, especially in applications where understanding the reasoning behind predictions is essential, such as in medical diagnostics or autonomous driving.

Moreover, feature map visualization can guide model improvement. For example, if a model is found to focus too heavily on background noise rather than relevant objects, this insight can lead to adjustments in the network architecture, training data, or preprocessing techniques. Feature map analysis is also used in the development of attention mechanisms and other advanced techniques that help the network focus on the most relevant parts of the input. Overall, feature map extraction is a powerful tool for both understanding and improving deep learning models.

Spatial Pyramid Pooling Fast (SPFF) Block

Three MaxPool2d layers are layered after a convolutional block in the SPPF Block. To extract dominating characteristics and lower the computational cost of the network, pooling layers are employed to downsample the spatial dimensions of the input volume. A particular kind of pooling procedure known as "max pooling" discards all values other than the largest value for each region in the input tensor. At the conclusion, each feature map that the MaxPool2d layer produced is concatenated and put into a convolutional block. By capturing multi-scale information through pooling operations at distinct degrees of granularity, spatial pyramid pooling enables neural networks to interact with images of varying resolutions.

4.7.2 Non-Maximum Suppression (NMS)

In YOLOv8, Non-Maximum Suppression (NMS) is a vital post-processing step used to refine the object detection results by eliminating redundant and overlapping bounding boxes. After the model generates multiple bounding boxes with associated confidence scores, NMS sorts these boxes by their confidence levels and iteratively selects the box with the highest score as the reference. It then computes the Intersection over Union (IoU) with other boxes and suppresses those with IoU values exceeding a predefined threshold, thus ensuring that only the most confident bounding box remains for each detected object. This process improves the accuracy and clarity of the final detections by reducing duplicates and overlaps, resulting in a more precise and interpretable output.

4.7.3 Activation functions

YOLOv8 primarily uses the SiLU (Sigmoid Linear Unit) activation function throughout its network for hidden layers, providing smooth gradients and better performance. For the final layer, Sigmoid activation is used to produce probabilities for objectness, bounding box coordinates, and class scores. This combination enhances both the model's accuracy and its ability to handle complex object detection tasks.

SiLU activation function is given by:

$$\mathrm{SiLU(x)} = \mathrm{x} \cdot \sigma(\mathrm{x}) \tag{4-1}$$

where, $\sigma(\mathrm{x})$ is the sigmoid function, which is given by

$$\sigma(\mathrm{x}) = 1/(1 + \mathrm{e}^\wedge - \mathrm{x}) \tag{4-2}$$

4.7.4 Loss functions

YOLOv8 employs dedicated loss components for object detection: it uses CIoU (Complete Intersection over Union) loss for bounding box regression, which improves accuracy by considering center distance, aspect ratio, and overlap.

It is given by

$$\mathrm{CIoU} = 1 - \mathrm{IoU} + \frac{\rho^2(b, b^*)}{c^2} + \alpha \cdot v \tag{4-3}$$

where, IoU: Intersection over union

$\rho^2(b, b^*)$: squared distance between the center points of the predicted box b and the ground truth box b*

c^2 : squared diagonal length of the smallest box enclosing the predicted and ground truth box

α and v : parameters that balance the distance and aspect ratio terms

For classification, it applies Binary Cross-Entropy (BCE) loss to assess the correctness of class predictions. It is given by

$$\mathrm{BCE} = -\frac{1}{N}\sum_{i=1}^{N} \left[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)\right] \tag{4-4}$$

where,

y_i: true class label

p_i: predicted probability of object belonging to the true class label

N: number of samples in the dataset

Additionally, YOLOv8 incorporates Distribution Focal Loss (DFL) to handle class imbalance by focusing more on hard-to-classify examples and improving overall classification performance. It is given by

$$\mathrm{DFL} \;= -\textstyle\sum_{i=1}^{c} (1 - p_i^{z})\log(p_i) \tag{4-5}$$

where,

p_i is the predicted probability for the class i

γ is the focusing parameter, > 0, which adjusts the rate of down-weighting

4.8 Transformer

The Transformer architecture forms the backbone of the BART model. The Transformer model architecture, renowned for revolutionizing various natural language processing tasks such as machine translation, text generation, and question answering, employs a self-attention mechanism to capture relationships between tokens in a sequence, allowing for parallel processing and effectively modeling long-range dependencies. BART utilizes the Transformer decoder architecture to generate coherent and contextually relevant text based on the embeddings provided by the encoder. The decoder's role is to produce output sequences, token by token, ensuring the answers generated are accurate and natural. Together, these models use the Transformer's capabilities to effectively handle and relate image-text pairs and generate detailed responses.

A detailed explanation of the components used in transformer architecture is as follows:

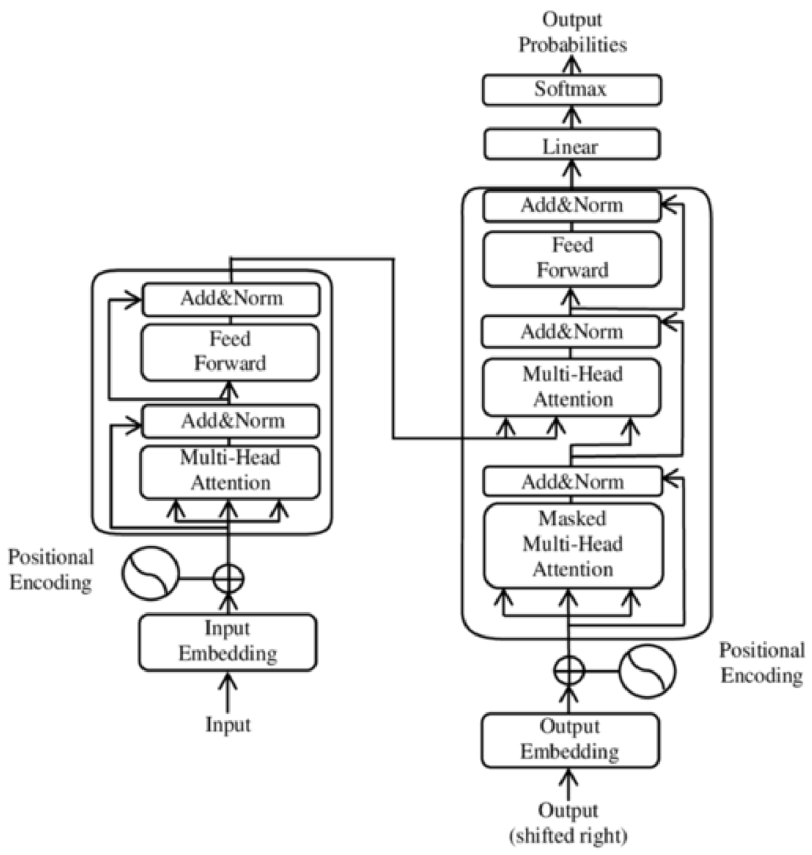


Figure 4-8: Transformer Architecture

4.8.1 Input Embedding

Input embedding in the Transformer architecture plays a critical role in converting tokens, such as words or subwords, from raw text into dense numerical vectors that the model can process. Each token in the input sequence is transformed into an embedding vector using an embedding matrix E. This matrix is initialized randomly or pretrained using methods like Word2Vec or GloVe to capture semantic relationships between tokens based on their contextual usage in the training data. These embeddings encode valuable information about the meaning and syntactic roles of words, allowing the model to understand and process language effectively.

4.8.2 Positional Encoding

Positional encoding is added to the input embeddings to provide information about the order of tokens in the sequence. This enables the model to distinguish between different positions and learn dependencies based on token order. To achieve positional encoding, sine, and cosine functions are used.

Formula:

$$P(k, 2i) = \sin\left[\frac{k}{n^{2i/d}}\right] \tag{4-6}$$

$$P(k, 2i+1) = \cos\left[\frac{k}{n^{2i/d}}\right] \tag{4-7}$$

Where,

k: position of the word in the input sequence

d: dimension of the embedded word

i: column index of the encoded value

n: user defined scalar, initially set to 10000

Here, equation (4-6) is used for even positions and equation (4-7) is used for odd positions.

4.8.3 Encoder Layer

The encoding component in a Transformer model consists of multiple encoder layers, each following a consistent structure. These layers take inputs from both the positional encoding and embedding layers. Within each encoder layer, two primary components are employed: a self-attention mechanism and a feed-forward neural network. Initially, the input vectors undergo processing through the self-attention layer, which enables the encoder to capture relationships among all tokens in the input sequence while encoding each token individually. The outputs from the self-attention layer are subsequently passed through a feed-forward neural network, which applies non-linear transformations to each token's representation, capturing complex dependencies and enhancing feature extraction. This uniform network structure is applied iteratively across all encoder layers, ensuring comprehensive processing of the input sequence before passing the transformed outputs to the subsequent encoder layer.

4.8.3.1 Self-Attention Mechanism

The self-attention mechanism in the Transformer model addresses the limitations of static embeddings done in embedding layer by allowing the model to dynamically focus on different parts of the input sequence when processing each token. Unlike traditional approaches where words are represented by fixed embeddings regardless of context, self-attention computes different weights for each token based on its relevance to other tokens in the sequence.

In self-attention, each token   in the input sequence is transformed into three vectors: Query (Q), Key (K), and Value (V). The meaning of behind these vectors can be explained as follows.

Query corresponds to the specific token of interest in the input sequence for which attention weights are computed. The Key represents all tokens in the sequence and is compared against the Query to determine their relevance or importance. Value vector provides the actual information or content associated with each token.

These vectors are derived from the token's embeddings through learned linear transformations as shown below:

$$\mathrm{Q}_i = \mathrm{W}_Q \mathrm{e}_{t_i} \tag{4-8}$$

$$\mathrm{K}_i = \mathrm{W}_K \mathrm{e}_{t_i} \tag{4-9}$$

$$\mathrm{V}_i = \mathrm{W}_V \mathrm{e}_{t_i} \tag{4-10}$$

where, e_{t_i} is the embedding vector of token t_i, and W_Q , W_K , W_V are learned weight matrices. These vectors   are then used to compute attention scores between token   and every other token in the sequence. The attention score is calculated as:

$$\mathrm{Attention}(t_i, t_j) = \frac{\mathrm{Q}_i \cdot \mathrm{K}_j^{\top}}{\sqrt{d_k}} \tag{4-11}$$

where, d_k is the dimensionality of K_i and K_j. These scores are scaled by $\sqrt{d_k}$ to to prevent gradients from becoming too small during training. The resulting attention scores are then used to compute a weighted sum of the Value vectors V_j, which is given by:

$$\mathrm{SelfAttention}(t_i) = \sum_j \mathrm{Softmax}\left(\mathrm{Attention}(t_i, t_j)\right) \cdot \mathrm{V}_j \tag{4-12}$$

where Softmax normalizes the attention scores across all tokens in the sequence to obtain a distribution over which tokens are most relevant for each token t_i.

4.8.3.2 Multi Head Attention

Multi-head attention involves computing multiple sets of attention weights, known as attention heads, in parallel. Each attention head independently learns to capture the different perspectives of input sequence. In "They saw her duck," multi-head attention handles ambiguity by focusing on both possible meanings of "duck" (action or bird), using different heads to analyze based on context like verb-object relationships.

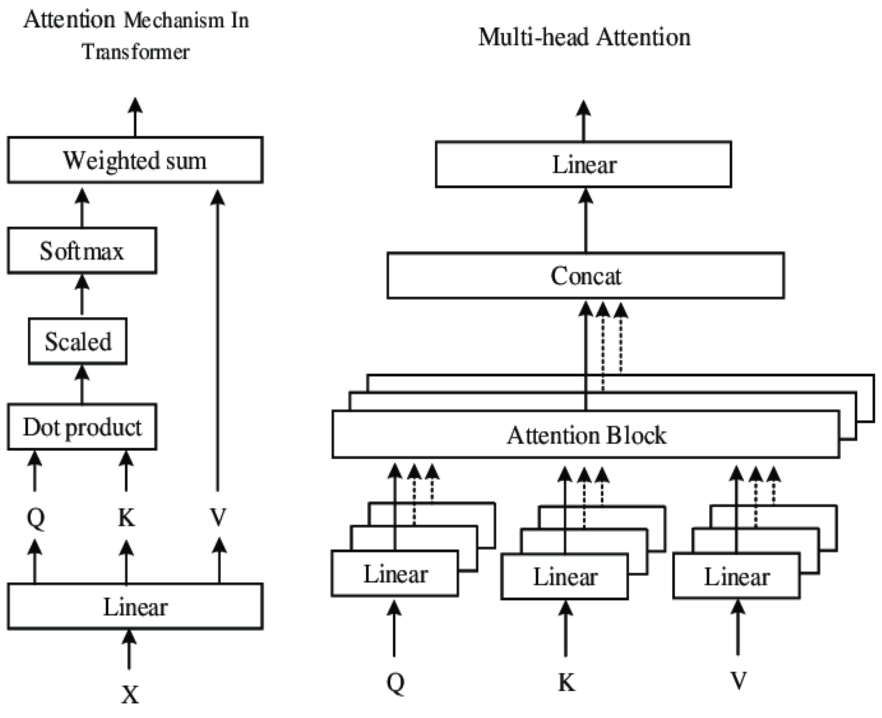


Figure 4-9: Attention Mechanism in Transformers

4.8.3.3 Feed Forward Neural Network

The self-attention mechanism's output undergoes processing through a feed-forward neural network, which introduces non-linearity to simulate complex interactions among words. This network consists of two linear transformations separated by an activation function, typically a Rectified Linear Unit (ReLU).

$$\mathrm{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{4-13}$$

where x is the output of multi-head attention layer, W1 and W2 are the learned weights.

4.8.3.4 Layer Normalization

Layer normalization is a technique used to standardize the output of each layer across the training examples by. Layer normalization is applied after both the self-attention and feed-forward network components within each Transformer encoder layer. This ensures that the activations across different tokens or units are normalized, increasing the convergence rate and making the model stable during training and inference.

4.8.4 Decoder Layer

The decoding component in a Transformer model consists of multiple decoder layers that mirror the encoder layers with some key differences. The first decoder layer receives inputs from both the positional encoding and embedding layers, similar to the encoder. In this case, the input to the decoder during training is the output sequence to be generated. Within each decoder layer, three primary components are utilized: a masked-multi-head attention mechanism, an encoder-decoder attention mechanism, and a feed-forward neural network. Initially, the input vectors pass through the masked multi-head-attention layer, allowing the decoder to focus on different parts of the output sequence generated so far while processing each token independently. The outputs from this layer are then fed into the encoder-decoder attention mechanism, which integrates information from the encoder's output. Finally, the processed information from these layers is passed through a feed-forward neural network, which applies non-linear transformations to enhance feature extraction and capture complex dependencies.

The components of the decoder layer that vary from the encoder layer are explained below:

4.8.4.1 Masked Multi-head Attention

Masked multi-head attention in the decoder differs from self-attention in the encoder by restricting the attention scope to only past and current tokens, preventing access to future tokens. This masking ensures that each token's generation relies solely on preceding tokens, preserving the sequence's temporal order. In contrast, self-attention in the encoder allows tokens to attend to all positions within the input sequence, providing a comprehensive understanding of relationships and dependencies across the entire input sequence.

4.8.4.2 Encoder Decoder Attention

Encoder-decoder attention is a mechanism in the decoder that allows the incorporation of information from the encoder's output. In this process, each token in the decoder generates Query vectors QD that interact with the Key vectors KE from the encoder. This interaction determines how much focus each decoder token should place on different parts of the encoder's output. The decoder also utilizes Value vectors VE from the encoder, weighted by the attention scores, to produce the final contextually relevant output. This setup enables the decoder to effectively capture the comprehensive context provided by the encoder, ensuring that the generated output is well-informed by the entire input sequence.

4.8.5 Output layer

The output layer of a Transformer model is responsible for generating the final predictions or tokens based on the processed representations from the decoder. After the last decoder layer processes the input sequence, the output embeddings are transformed into probabilities for each possible token in the vocabulary. This is achieved through a linear transformation followed by a softmax function.

The linear transformation is given by

$$z_i = \sum_j w_{i,j} x_j + b_i \tag{4-14}$$

where, x_j is the output embeddings from the final decoder layer. Then, z_i is passed to the softmax function given by

$$y = \text{softmax}(z_i) \frac{\exp(z_i)}{\sum_j \exp(z_i)} \tag{4-15}$$

The softmax function ensures that the output probabilities sum to one, making it possible to select the most likely token for each position in the sequence.

4.9 BART (Bidirectional and Auto-Regressive Transformer)

The BART model combines the benefits of both bidirectional and autoregressive architectures. The Bidirectional Encoder processes the input sequence in both forward and backward directions, enabling it to capture the context from both sides of any token. The Autoregressive Decoder, on the other hand, generates the output sequence one token at a time, predicting the next token based on the previous tokens generated and the encoder's output. This setup allows the decoder to generate coherent and contextually accurate sequences.

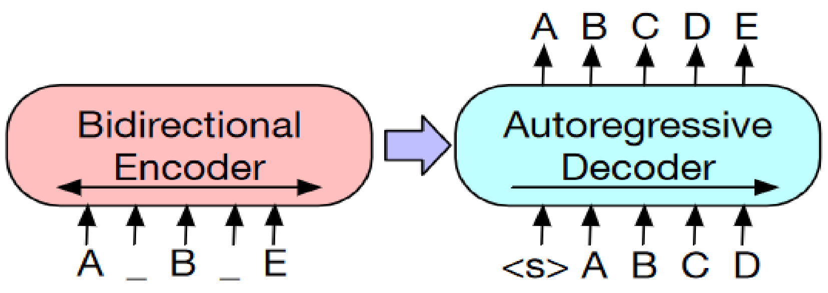


Figure 4-10: BART Model Architecture [16]

Encoder

The BART encoder processes the input text by converting it into a set of rich contextual embeddings that capture both the semantic and syntactic relationships within the sequence. It begins by tokenizing the input text into subword units and converting these tokens into dense vector representations (embeddings) of 768 dimensions. To preserve the order of tokens, positional encodings are added to these embeddings, guided by the max_position_embeddings parameter. The encoder then passes the embeddings through 6 encoder layers, each equipped with 12 attention heads, allowing it to focus on different parts of the input sequence simultaneously. Through its multi-head self-attention mechanism and feedforward networks, the encoder captures intricate dependencies between tokens, refining the embeddings at each layer. The output is a set of highly informative embeddings that encapsulate the meaning and context of the input text, ready to be used by the decoder for generating coherent and context-aware responses.

Decoder

The BART decoder has 6 layers, each with 12 attention heads, and uses the GeLU activation function. Additionally, it has a max_position_embeddings parameter (optional, defaulting to 1024), which represents the maximum sequence length that the model might ever be used with. This parameter is typically set to a large value just in case, such as 512, 1024, or even 2048.

The decoder typically uses an initial token, often represented as "<s>", to start the generation process. Each subsequent token is generated by considering both the encoder's context and the tokens generated so far. This autoregressive approach ensures that each token in the output sequence is dependent on the previous tokens, thereby maintaining coherence and fluency in the generated text.

This decoder receives input from the BART encoder, which provides embeddings of 768 dimensions. The BART encoder processes the input (object names with associated questions) and generates embeddings that encapsulate the multimodal information. These 768-dimensional embeddings are then fed into the BART decoder. The BART decoder uses these embeddings as context to generate the final output sequence, ensuring that the generated answers are relevant and informed by both visual and textual information from the input.

4.10 Speech Recognition model

For speech recognition, the Whisper model developed by OpenAI has been utilized—a state-of-the-art system designed to transcribe spoken language into text with exceptional accuracy. Whisper leverages a transformer-based architecture and has been trained on an extensive dataset of multilingual and multitask audio, which includes a wide variety of languages, accents, speaking styles, and background noise levels. This diverse training data allows Whisper to handle real-world audio complexities with remarkable robustness.

The model's architecture consists of an encoder-decoder structure, where the encoder takes raw audio signals as input and converts them into high-dimensional embeddings rich in contextual and linguistic information. These embeddings are then processed by the decoder, which generates the corresponding text transcription, ensuring coherence and semantic alignment with the spoken input.

One of Whisper's standout features is its multilingual training, making it highly versatile and capable of transcribing speech from numerous languages. This multilingual capability is particularly useful for recognizing and parsing Nepali words, as the model can seamlessly adapt to the phonetic and linguistic nuances of Nepali speech. Whether dealing with unique accents or local variations in pronunciation, Whisper demonstrates its strength in accurately transcribing Nepali along with many other languages, making it a powerful tool for diverse speech recognition tasks.

4.11 Evaluation Metrics

The evaluation metrics used for measuring the performance of the answers generated by BART as are follows:

4.11.1 BLEU Score

The BLEU (Bilingual Evaluation Understudy) score is a precision-based metric that measures the overlap between n-grams of the generated text and reference text. Its formula is given as:

$$\text{BLEU} = BP \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{4-16}$$

where,

BP: Brevity Penalty, which penalizes short generated sentences. It is calculated as:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{\left(1-\frac{r}{c}\right)} & \text{if } c \leq r \end{cases} \tag{4-17}$$

where, c is the length of the candidate (generated) sentence. And, r is the length of the reference sentence.

wn: weight of n-grams (typically takes the value of 1N

pn : precision for n-grams, which is given by

$$p_n = \frac{\sum_{n-gram-generated} Count_{Clip}(\text{n} - \text{gram})}{\sum_{n-gram-generated} \text{Count}(\text{n} - \text{gram})} \tag{4-18}$$

where, countclip(n gram) is the clipped count of the n-gram in the generated sentence, which means it is limited to the maximum number of times the n-gram appears in any single reference sentence. High BLEU scores indicate that the generated text is fluent and closely matches the reference text in terms of word sequences. However, BLEU may not capture semantic equivalence well and can be overly harsh on legitimate paraphrases that do not share exact n-grams with the reference texts.

4.11.2 ROUGE

ROUGE (Recall-Oriented Understudy for Gisting Evaluation) measures the overlap of n-grams and sequences between the generated and reference texts, focusing more on recall than precision. The common variants are ROUGE-1, ROUGE-2 and ROUGE-L

The formula for ROUGE-L is given by:

$$\text{ROUGE-L} = \frac{\text{LCS (generated,reference)}}{\text{length of reference}} \tag{4-19}$$

where, LCS is the the longest common subsequence between the generated text and reference text. High ROUGE scores indicate that the generated text effectively captures the essential content and information from the reference text. While ROUGE provides a good measure of content coverage, it may not emphasize fluency and precision as much as BLEU.

4.11.3 METEOR Score

The METEOR (Metric for Evaluation of Translation with Explicit ORdering) score considers synonyms, stemming, and paraphrasing, providing a more nuanced evaluation of the semantic content of the generated text. This allows METEOR to recognize that different words or phrases can convey the same meaning, thus improving the evaluation of semantic content. The METEOR score is calculated using the following formula:

$$\text{METEOR} = F_{mean}.(\text{1-penalty}) \tag{4-20}$$

Where F_{mean} is the harmonic mean of precision and recall, and the penalty is a fragmentation penalty that penalizes incorrect word order. High METEOR scores indicate that the generated text captures the meaning and semantic content of the reference text, making it a valuable complement to BLEU and ROUGE.

4.11.4 Semantic Similarity

Semantic similarity measures the closeness in meaning between two pieces of text, such as words, sentences, or documents. A common metric for this is Cosine Similarity, which computes the similarity between two vectors $\vec{A}$ and $\vec{B}$. The formula is given by:

$$\text{Cosine Similarity } (\vec{A}, \vec{B}) = \frac{\vec{A} \cdot \vec{B}}{||\vec{A}||||\vec{B}||} \tag{4-21}$$

Where:

$\vec{A} \cdot \vec{B}$ is the dot product of vectors $\vec{A}$ and $\vec{B}$,

$||\vec{A}||$ is the magnitude (Euclidean norm) of vector $\vec{A}$,

$||\vec{B}||$ is the magnitude of vector $\vec{B}$.

4.11.5 Exact Match Accuracy with Fuzzy Logic

Exact Match Accuracy is a metric that evaluates how frequently the predictions produced by a model precisely match the expected or ground truth results. This measure is particularly valuable in assessing the performance of models where exact alignment with the target is crucial. However, in many real-world applications, the output may not always match exactly due to variations in phrasing, synonyms, or paraphrased expressions that convey the same underlying meaning.

By integrating fuzzy logic with Exact Match Accuracy, partial matches are considered by allowing a similarity threshold accommodating cases where slight variations in wording or structure occur but the intended meaning remains the same, making it ideal for scenarios where the model needs to understand and evaluate semantic similarities, such as handling synonyms or paraphrased words.

$$\text{Exact Match } = \begin{cases} 1, & \text{if Ratio}(A, B) \geq T \\ 0, & \text{otherwise} \end{cases} \tag{4-22}$$

Here, T is the minimum similarity score (e.g., 90%) for a match to be considered "exact."

Here, Ratio function, which measures how similar two strings are using Levenshtein Distance (edit distance).

4.11.6 Word Error Rate (WER)

The **Word Error Rate (WER)** is a metric to evaluate the accuracy of speech recognition systems by comparing the predicted transcription to the reference transcription. It is calculated as follows:

$$\text{WER}= \frac{\text{S+D+I}}{\text{N}} \tag{4-23}$$

Where:

S = Number of substitutions (words incorrectly replaced).

D = Number of deletions (words missing in the predicted transcription compared to the reference).

I = Number of insertions (extra words added in the predicted transcription).

N = Total number of words in the reference transcription.

5. IMPLEMENTATON DETAILS

5.1 Dataset Analysis

5.1.1 Dataset Collection

The dataset was meticulously curated, incorporating images collected through site visits from Patan Durbar Square, Kathmandu Durbar Square, and Bhaktapur Durbar Square. To further increase the size of the dataset, web scraping from platforms like Shutterstock was also done, and relevant monument images were extracted from existing datasets. This comprehensive effort resulted in a final dataset comprising a total of 6812 images after applying augmentation techniques. The dataset consists of 12 classes, with 6 historical objects and 6 cultural monuments found within Kathmandu Valley. The histogram representing the classes of the dataset after augmentation is shown below.

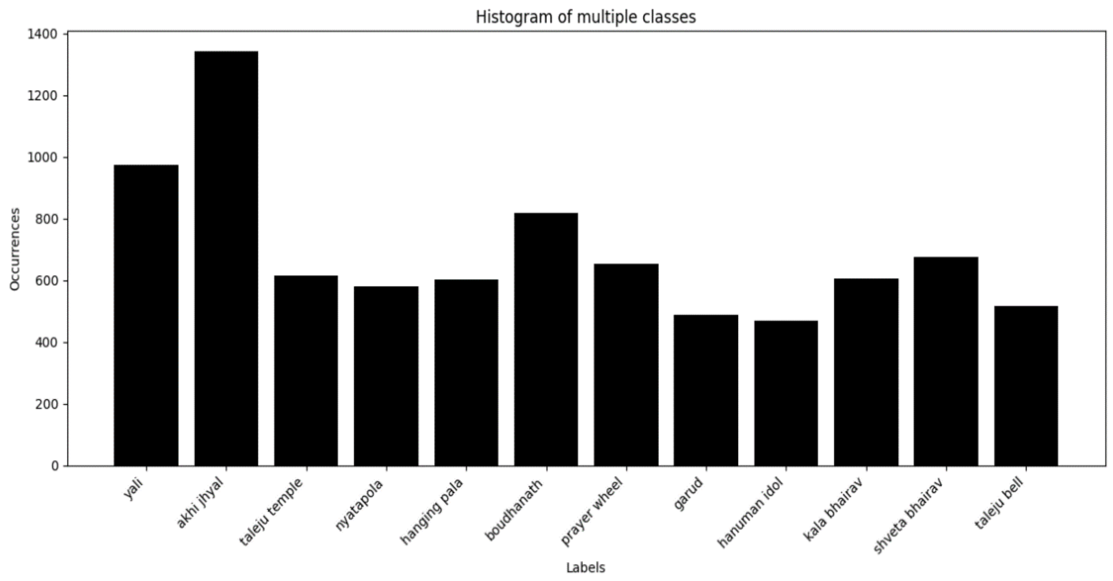


Figure 5-1: Histogram of Multiple Classes

5.1.2 Data Annotation

For data annotation we used Computer Vision Annotation Tool (CVAT) which is an open-source software application designed to facilitate the annotation of images and videos for computer vision tasks. Developed by Intel, CVAT provides a web-based interface that enables users to label objects within images and video frames efficiently. The tool supports a wide range of annotation types, including bounding boxes, polygons, polylines, and points, making it versatile for various use cases such as object detection, image segmentation, and keypoint tracking. CVAT's collaborative features allow multiple annotators to work on the same project simultaneously, streamlining the annotation process for large datasets. Additionally, it supports integration with deep learning frameworks, enabling users to automate parts of the annotation process using pre-trained models. CVAT's flexibility, user-friendly interface, and robust feature set make it a popular choice among researchers and developers in the field of computer vision..

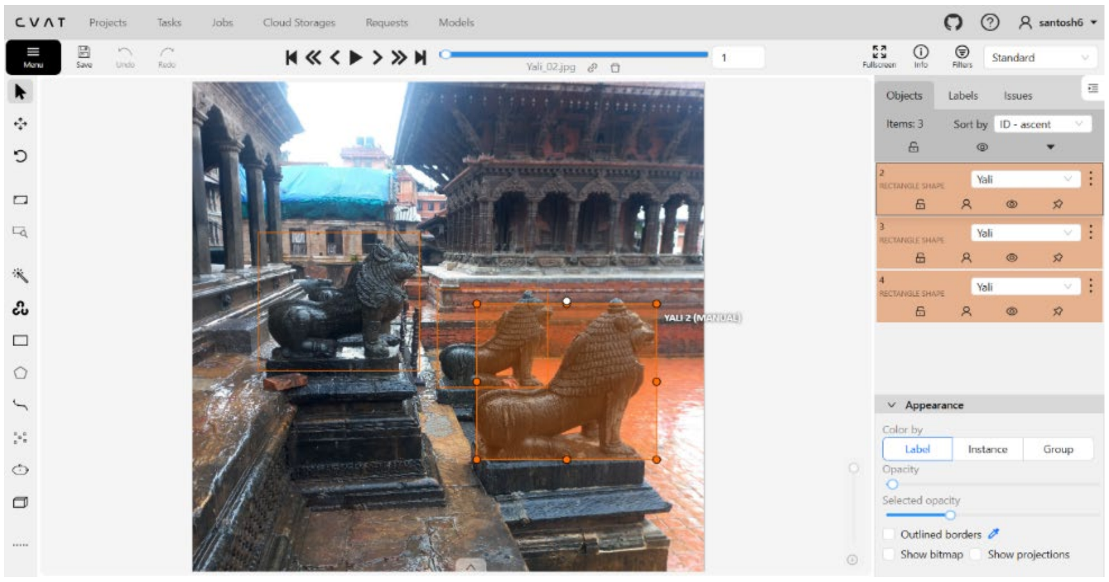


Figure 5-2: Annotation using CVAT

We used CVAT to annotate our dataset by creating rectangular bounding boxes around the objects of interest in each image. The tool's intuitive interface allowed us to efficiently label and organize the data for our specific project needs. Once the annotation process was complete, we exported the annotations in YOLO format, which is well-suited for training YOLO object detection models. This format includes a text file for each image, containing the class label and normalized coordinates of each bounding box, making it easy to integrate with our machine learning pipeline.

5.1.3 Data Augmentation for Images

To enhance the dataset, five unique augmented images were generated from each original image using the Albumentations library. The applied augmentation techniques included horizontal flipping, which mirrors the image to provide left-right variations; brightness and contrast adjustments, which alter the lighting and contrast to simulate different lighting conditions; gamma adjustments, which modify the overall luminance of the image; Gaussian noise, which introduces random noise to make the model more resilient to distortions; Gaussian blur, which smooths the image to reduce noise; and rotation, which rotates the image within a limit of ±15 degrees to provide orientation variability. Five images were generated using a random combination of the above-mentioned augmentation techniques. These transformations create a diverse and realistic set of images, improving the model's robustness and generalization capabilities.

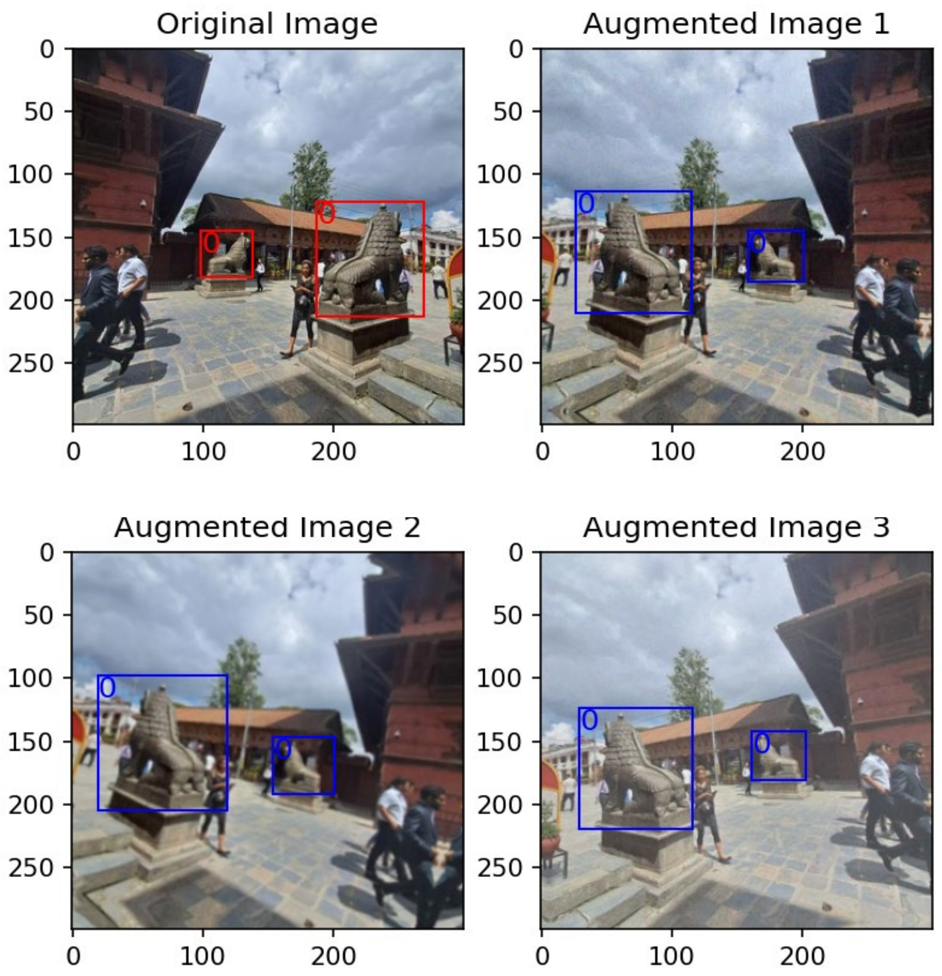


Figure 5-3: Image Augmentation

5.1.4 Question-Answer Pairs

We have created multiple question answer pairs for each class describing that particular object’s history, its significance, its uses and other relevant information. There are currently 8,898 question answer pairs in total for all classes meaning an average of 765 question answer pairs for each class. This question answer covers the important and relevant information about the classes that the tourists visiting the historical places might want to know about.

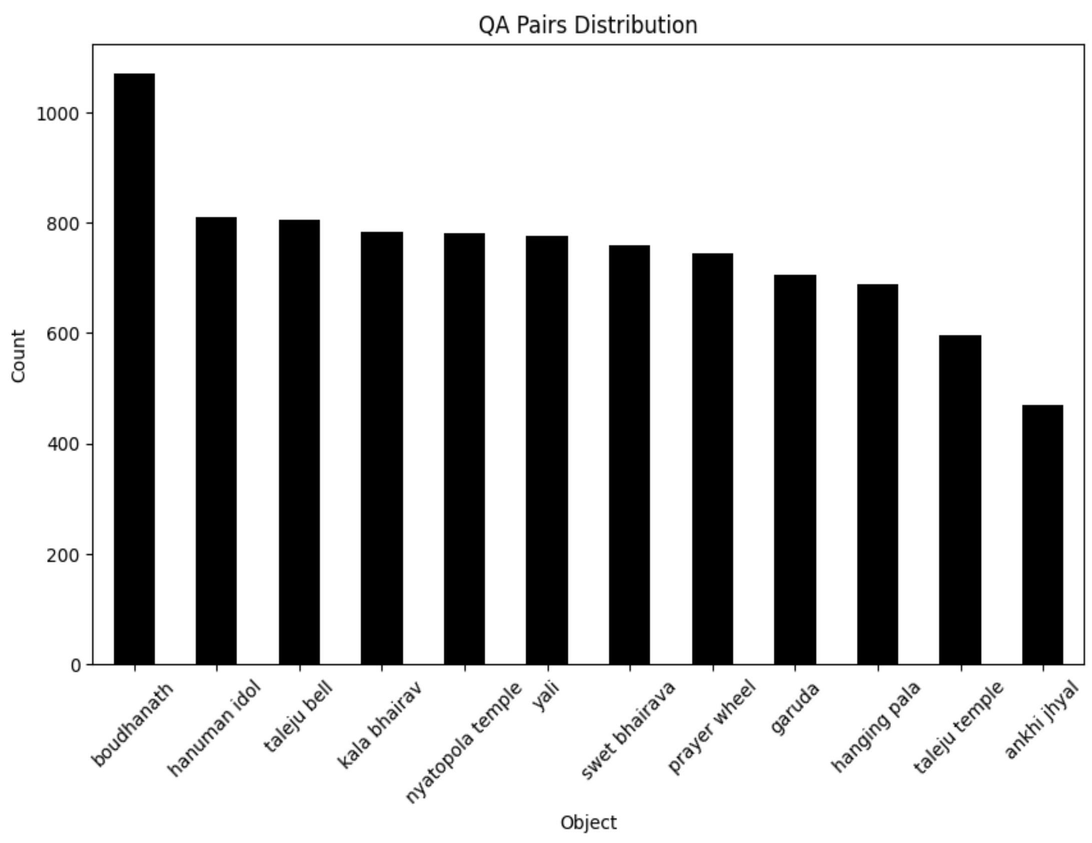


Figure 5-4: Distribution of Question-Answer Pairs

The figure shows a bar chart titled "Distribution of Question Answer pairs," which provides a detailed overview of the frequency of different objects in a question-answering dataset. The object "Boudhanath" stands out as the most frequently mentioned object, with a significantly higher frequency compared to others in the dataset. Objects such as "Nyatapola," "Taleju Temple," "kala bhairava," “garuda,” “swet bhairava,” “hanuman idol,” “taleju temple,” “taleju bell,” “boudhanath,” “akhi jhyal,” “hanging pala,” “yali,” also appear multiple times. The chart visually captures the disparity in object representation, with a steep decline in occurrences as we move from the most to the least mentioned objects, reflecting the dataset's underlying focus on specific themes or areas.

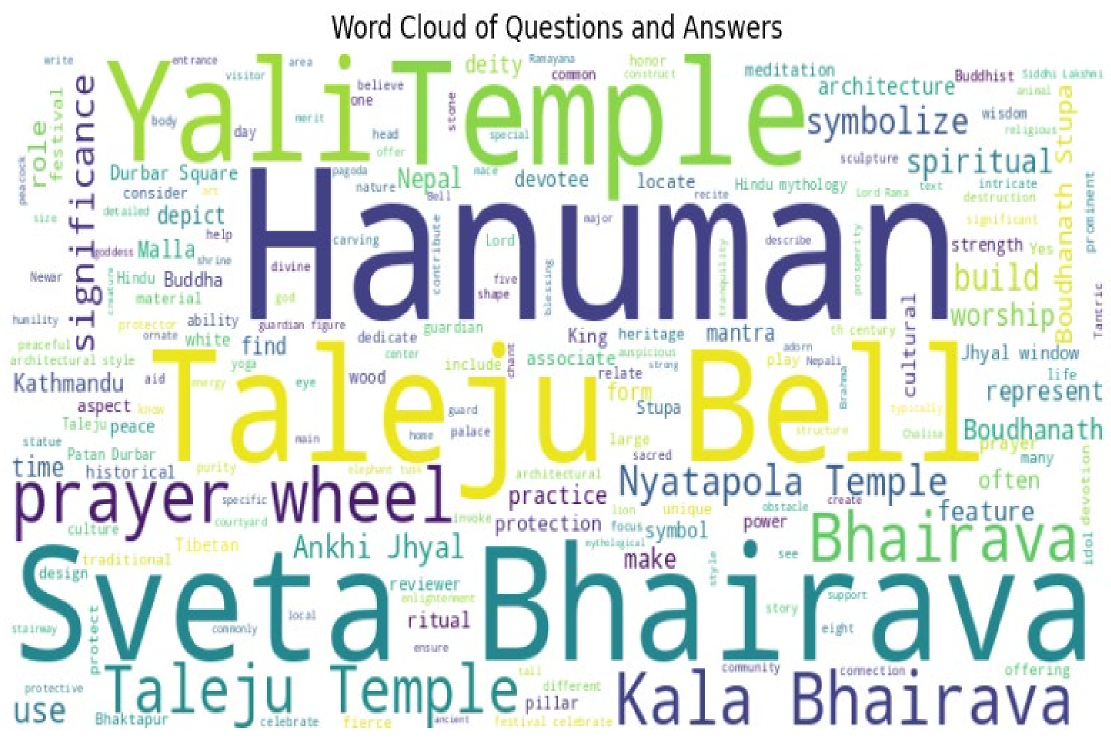


Figure 5-5: Word Cloud of questions and answers

A word cloud is a visual representation of text data where the size of each word indicates its frequency or importance in the dataset. In the figure titled "Word Cloud of Objects," the most prominent words, such as "Prayer Wheel", "Garuda", "Swet Bhairava", "Hanuman Idol", appear larger, indicating they are the most frequently mentioned objects in the dataset of questions and answers. Other words like "Yali", "taleju bell", "taleju temple", "Kala Bhairava", and "Nyatapola" are also significant but appear slightly lesser, showing they are less frequent but still notable. This word cloud highlights key terms related to religious and cultural topics, with a focus on deities, temples, and spiritual practices, suggesting that the dataset primarily revolves around these themes. The visual representation helps quickly identify the most common and relevant words, providing insights into the main subjects of the questions and answers.

5.1.5 Question-Answer Augmentation

To enhance the QA dataset and increase its robustness, four distinct text augmentation techniques were applied to each original question-answer pair. These methods were designed to introduce controlled variations, enriching the dataset with diverse linguistic structures and vocabulary. By creating variations in the text, these augmentations aim to simulate real-world scenarios, ensuring that the model can effectively handle different phrasings and unpredictable user inputs during deployment.

Embedding Augmentation

It modifies word embeddings by introducing subtle alterations. Word embeddings represent words in a vector space based on their semantic similarities, and slight adjustments in these embeddings result in small shifts in meaning without altering the overall context. This technique ensures that the dataset includes variations that teach the model to recognize and respond to semantically similar but syntactically different inputs, improving its generalization capabilities

Random Deletion

It involves removing random words from questions or answers. This augmentation method is designed to mimic scenarios where information is incomplete or missing, exposing the model to a variety of noisy inputs. By training the model on such altered data, this technique enhances its ability to manage incomplete or fragmented user queries effectively.

Translation-based Augmentation

It translates the text from one language to another and then back into the original language. This introduces natural linguistic variations and alternative sentence structures, reflecting real-world language usage. These transformations help the model adapt to regional language patterns and subtle phrasing differences that can arise in multilingual contexts.

WordNet-based Augmentation

It uses synonyms or related terms to replace certain words within the text. This approach enriches the dataset with diverse vocabulary and varied sentence structures, enabling the model to better understand and respond to different ways of expressing the same idea. By learning from these lexical variations, the model becomes more adept at handling diverse and nuanced inputs.

Table 5-1: Text Augmentation Techniques

Original Question	**Augmentation Technique**	**Augmented Question**
Which god is present in Taleju Temple?	Embedding Augmentation	Which deity is present in Taleju Temple?
Which god is present in Taleju Temple?	Random Deletion	Which god present in Taleju Temple?
Which god is present in Taleju Temple?	Translation-based Augmentation	ताजेलू मंदिर में कौन सा देवता है? (Translated from Hindi and back to English: Which *deity* is present in Tajelu Temple?)
Which god is present in Taleju Temple?	WordNet-based Augmentation	Which god is found in Taleju Temple?

5.1.6 Audio Dataset for Text to Speech testing

The dataset contains audio query files related to twelve unique objects: Yali, Akhi Jhyal, Nyatapola, Hanging Pala, Boudhanath, Prayer Wheel, Garud, Hanuman Idol, Kala Bhairav, Shveta Bhairav, and Taleju Bell. Each object is associated with audio queries recorded in five different accents: US, UK, Indian, Chinese, and Australian. For each object, there are a total of 100 audio files, with 20 questions recorded in each accent. This results in a uniform distribution of audio queries across objects and accents, creating a total of 1,200 audio files in the dataset. The structured organization of the dataset ensures that each object is equally represented across accents and question types, making it suitable for linguistic and acoustic analysis of the Whisper Model that incorporates Custom Vocabulary Injection used as the text to speech method.

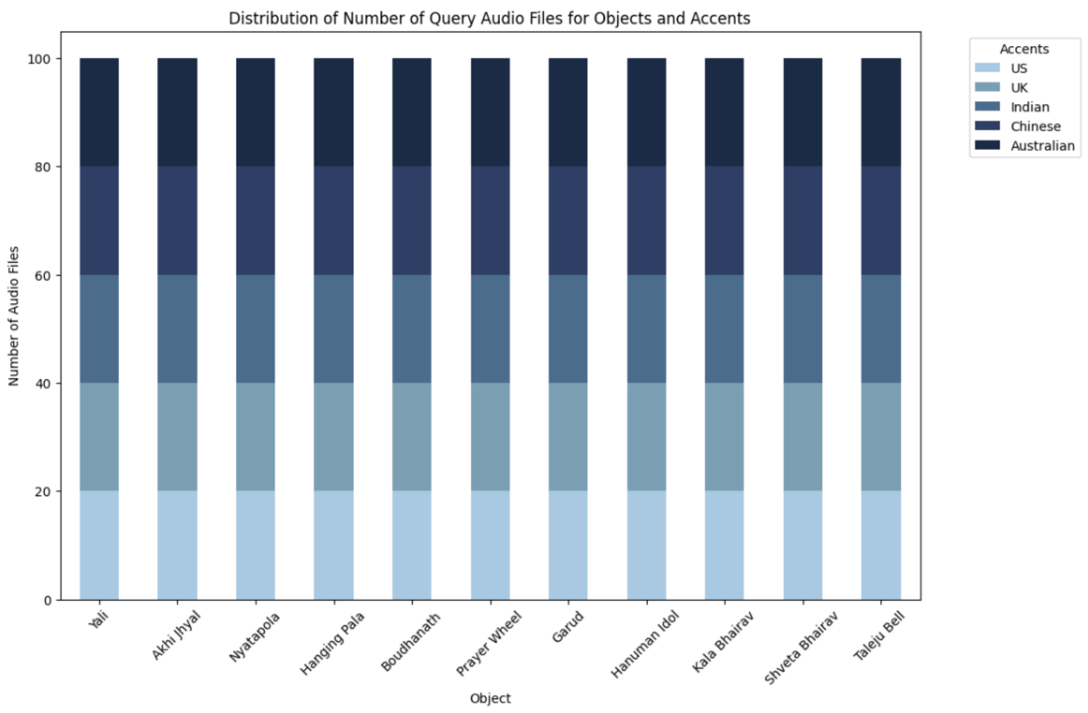


Figure 5-6: Audio Files Distribution

5.2 YOLO Model Training

We trained a YOLOv8n model by fine-tuning it on our custom dataset using its default parameter settings and achieved impressive results, which negated the need for any further adjustments. The training process was conducted for 100 epochs with a batch size of 16, utilizing the model's default configurations. This approach proved to be effective, as the initial outcomes met our performance expectations. The training was performed using Kaggle's free GPU, specifically a GPU P100. The hyperparameters that were used during the training process are shown in the table below.

Table 5-2: YOLO Training Hyperparameters

Hyperparameters	**Values**	**Description**
epochs	100	Total number of training epochs.
batch size	16	Number of samples per batch
image size	640	Input image dimensions in pixels
optimizer	AdamW (auto)	Adjusts a model's parameters to minimize the loss function
learning rate	0.01	Influences how rapidly model weights are updated.

freeze	null	None of the layers are frozen (to remain unchangeable during training)
dropout	0.0	Dropout (setting of fraction of neurons to zero) is not applied
IoU	0.7	Intersection over union threshold for non-max suppression
box	7.5	Weight for bounding box regression loss set to 7.5
cls	0.5	Weight for classification loss set to 0.5
dfl	1.5	Weight for distribution focal loss set to 1.5

5.3 Hyperparameter Tuning

Hyperparameter tuning is the process of finding the best combination of hyperparameters for a machine learning model to optimize its performance. The process used for hyperparamter tuning of the BART model was bayesian optimization.

Bayesian optimization

Bayesian Optimization is a probabilistic model-based optimization technique used to find the optimal hyperparameters for machine learning models. Unlike grid search or random search, which explore the hyperparameter space exhaustively or randomly, Bayesian optimization aims to intelligently search for the best combination of hyperparameters by learning from past evaluation results. The idea is to model the objective function (e.g., model performance) as a probabilistic function and use this model to select the most promising hyperparameters to evaluate next.To implement bayesian optimization, a popular python library Optuna was used.

Optuna is a popular Python library used for hyperparameter optimization that can implement Bayesian optimization. It provides an efficient and flexible way to search for the best hyperparameters in machine learning workflows. In Optuna, sampling and pruning are two key concepts that contribute to efficient hyperparameter optimization:

1. Sampling:

Sampling refers to the process of selecting hyperparameters to evaluate from the defined search space. Optuna uses intelligent sampling strategies to explore the hyperparameter space efficiently. Optuna's sampling process adapts over time by utilizing past trial results to explore hyperparameter combinations more intelligently and efficiently.

2. Pruning:

Pruning is the process of early stopping unpromising trials based on intermediate results. If a trial is unlikely to perform well, Optuna stops it before completing its execution, saving computational resources. After each trial's intermediate evaluation, Optuna uses the pruning algorithm (often based on the validation score) to decide whether to stop the trial early. The pruning method used is Median Pruner. It stops trials that perform worse than the median performance of all trials at the same step.

Algorithm for hyperparameter tuning

- Select Hyperparameters: Choose which hyperparameters to tune (e.g., learning rate, batch size, number of layers).
- Define Search Space: Define the range or set of values for each hyperparameter.
- Choose a Search Method: Select a method to explore the hyperparameter space, such as:
- Bayesian Optimization: Uses past evaluation results to intelligently pick the next combination to test.
- Evaluate Models: Train and evaluate models with different hyperparameter combinations
- Select the Best Model: Choose the combination of hyperparameters that gives the best performance based on evaluation metrics.

Hyperparameter tuning settings:

A total of 20 trials were conducted, each testing a unique combination of hyperparameters to optimize the model's performance. For every trial, the selected hyperparameters were evaluated over 5 epochs, using fuzzy match accuracy as the objective function to measure how well the model aligned with the desired outputs. After completing all 20 trials, the combination of hyperparameters that achieved the highest fuzzy match accuracy was identified and chosen as the optimal configuration for the task. This systematic exploration ensured a comprehensive search for the best-performing parameters while balancing computational efficiency.

Table 5-3: Hyperparameters Tuning Settings

Parameter	**Value**
No of Trials	20
No of Epochs	5 per trial
Objective Function	Fuzzy Match Accuracy
Evaluation Metric	Highest Fuzzy Match Accuracy (%) across trials

5.4 BART Model Training

The BART-base model was trained on a custom dataset of QA pairs, specifically curated for the task. Several hyperparameter configurations were explored using the Optuna library, and the optimal settings were identified through extensive experimentation. Among other BART variants, such as BART-large and BART-large-CNN, the BART-base model achieved the best results, demonstrating its suitability for this dataset and task. The training was performed using Google Colab's GPU, T4 GPU. The hyperparameters that were used during the training process are shown in the table below.

Table 5-4: BART Training Hyperparameters

Hyperparameters	**Values**	**Description**
epochs	14	total number of training epochs.
batch size	8	number of samples per batch
optimizer	AdamW (auto)	Adjusts a model's parameters to minimize the loss function
learning rate	0.00021	Influences how rapidly model weights are updated.
warmup steps	784	Number of steps to gradually increase the learning rate from zero to maximum
num beams	8	number of candidates the model considers at each decoding step during beam search

dropout	0.200	sets the fraction of neurons to zero to prevent overfitting
attention dropout	0.320	sets a fraction of attention weights to zero to prevent overfitting
weight decay	0.000010	adds a penalty to the loss function based on the size of the model's weights

5.5 Speech-to-Text Conversion

The system utilizes OpenAI's Whisper model for transcribing user queries from speech to text. To ensure optimal performance, query audio is resampled to 16,000 Hz, matching the training configuration of the Whisper model. This resampling step helps maintain consistency in input data and enhances transcription quality. Additionally, custom vocabulary injection is implemented to improve accuracy for domain-specific terms such as object names. This ensures precise recognition and transcription of critical terminology. The transcribed text query is then passed, along with detected object names, to the BART model for answer generation.

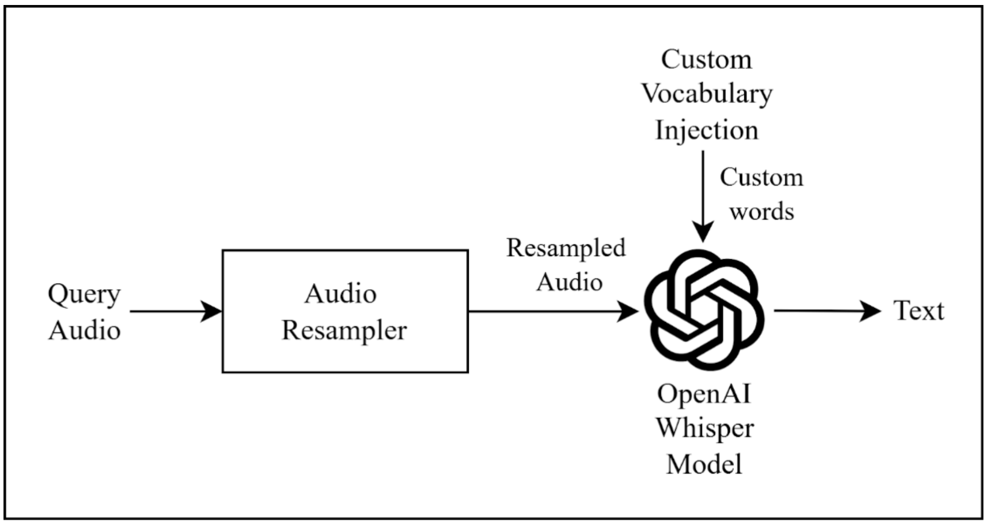


Figure 5-7: Speech-to-Text Conversion

Whisper Model

The Whisper model is a state-of-the-art automatic speech recognition (ASR) system developed by OpenAI, designed for transcription, language identification, and translation. It is trained on 680,000 hours of multilingual and noisy audio, enabling it to handle diverse languages and varying audio quality effectively. Whisper utilizes a

Transformer-based encoder-decoder architecture, where the encoder extracts features from log-Mel spectrograms of audio inputs, and the decoder generates text outputs, including multitask predictions like transcription and translation. For our project, the Whisper model plays a critical role in transcribing audio inputs while addressing domain-specific challenges.

To improve transcription accuracy for specialized terminology, such as object names and historical artifact identifiers in local Nepali languages, we implement custom vocabulary injection. This technique biases the model's predictions toward a predefined vocabulary, ensuring that key terms like ‘Yali’, ‘Akhi Jhyal’, ‘Taleju Temple’ are recognized and transcribed correctly. By integrating custom vocabulary, the Whisper model becomes more reliable in the specific context of our system, particularly for processing culturally significant terms that are essential for our Visual Question Answering (VQA) application.

5.6 Web Application

Currently in this project, a Web application is created using Django. All the used models, hosted in the server, directly communicate with the Web application. It initially takes input from the user in the form of images and text. This image is later on sent to the server as an input parameter. After an image is detected, the user can ask questions related to that object. BART model interprets the question and finally sends a meaningful descriptive answer as output to the user on the UI.

Communication Mechanism

Initially, an image is uploaded to the backend server, where the YOLOv8 model processes it to detect objects. The server then sends a basic response to the UI, including the names of the detected objects. Once the objects are identified and displayed, the user can enter text as a second input. The user interacts with the BART model to ask questions related to the detected objects, while the backend facilitates the conversation, displaying both the questions and answers on the interface. If no objects are detected or if a question is asked before an image is uploaded, a warning message will be displayed, ensuring a seamless and informative user experience.

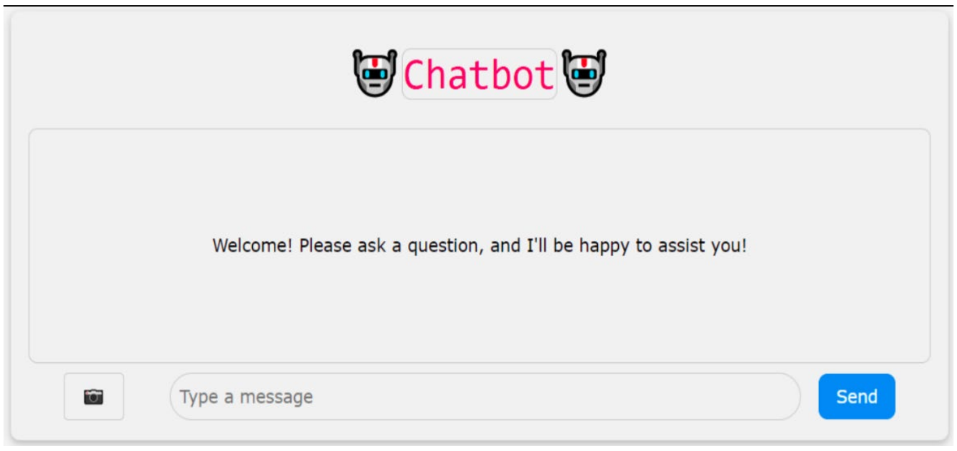

Chatbot
Welcome! Please ask a question, and I'll be happy to assist you!
Type a message
Send

Figure 5-8: Chatbot Interface

5.7 Software Implementation

5.7.1 Building Restful API

Building a RESTful API has been an essential aspect of our project. A RESTful API, or Representational State Transfer API, has been designed to facilitate communication between the client and server by adhering to REST principles such as simplicity, scalability, and statelessness.

In this project, the API has been developed using Django and Django REST Framework (DRF) to enable cross platform development, meaning using this API, information can be displayed in both Android application as well as Web Application.

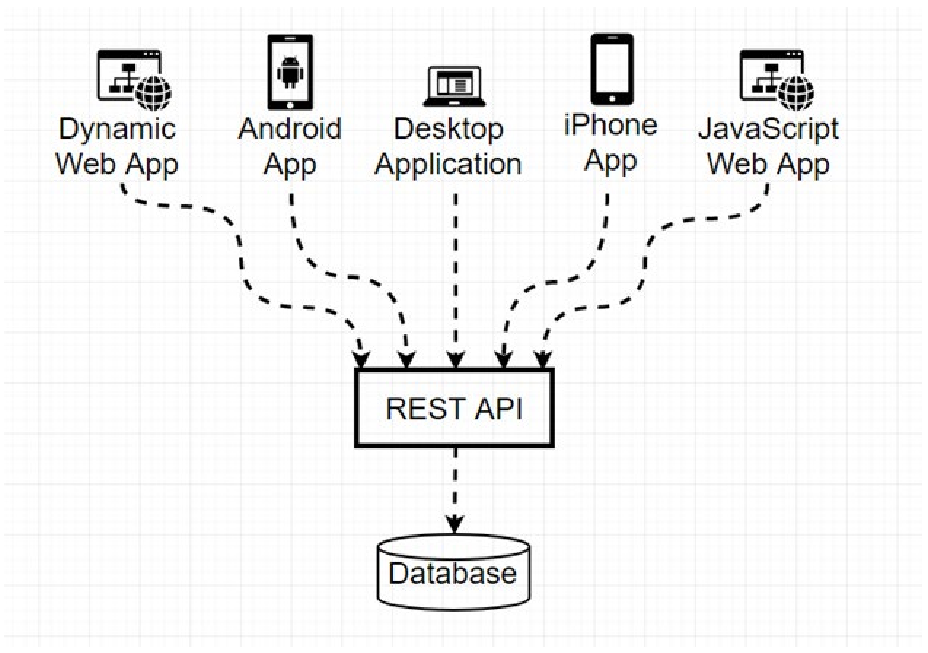


Figure 5-9: Cross-Platform Development

Understanding Restful API

A RESTful API has been implemented as a web service enabling clients to interact with resources stored on the server using standard HTTP methods. These resources, such as chat history or upload functionality, have been uniquely identified by URIs (Uniform Resource Identifiers).

The API architecture has adhered to the following principles:

Statelessness: Between each request the server does not store any client-specific data between requests. Each request must contain all the information needed to process it.

Resource-Based Design: Resources have been exposed through logical endpoints, such as `/`upload`/` or `/generate_audio/`.

Standard HTTP Methods: GET method is used to fetch the response from the model. Similarly, the POST method is used to post the image and audio to the application.

5.7.2 Various Endpoint with its feature

This project encompasses the development of a suite of APIs for multimedia handling and natural language processing. Key features include image and audio upload and retrieval, speech-to-text and text-to-speech conversion, a gibberish test to assess the quality of text, object detection in images, and chat history management with clearance capabilities. These APIs offer a versatile set of tools for users.

Image Upload

- Endpoint: upload/
- View Function: views.upload_image
- Method: POST
- Purpose: Handles image upload, performs object detection, and stores results for further interaction.

Audio Upload

- Endpoint: upload_audio/
- View Function: views.upload_audio
- Method: POST

- Purpose: Handles audio uploads, performs speech-to-text conversion, synthesizes text-to-speech (optional), and stores results for further interaction.

Speech to text

- Endpoint: stt/
- View Function: views.handle_audio_input
- Method: POST
- Purpose**:** Samples the uploaded speech in 16000hz then transcribe the generated audio to fetch the text from that speech.

Bart Text Generation

- Endpoint: ask_question/
- View Function: views.ask_question
- Method: POST
- Purpose: Generates natural language responses or answers to user queries using the BART model, enabling question-answering or text completion tasks.

Text to speech

- Endpoint: tts/
- View Function: views.generate_audio
- Method: GET
- Purpose: Converts input text into synthesized speech, generates an audio file.

Chat History Clearance

- Endpoint: tts/
- View Function: views.ClearChatHistory
- Method: GET
- Purpose: Clears the session's chat history, removing all previous interactions to reset the conversation state.

5.7.3 Exception Condition

Exception conditions are designed to handle scenarios where the user input or uploaded content does not meet the expected format or cannot be processed as intended. These conditions help ensure that the system behaves predictably and provides meaningful feedback to the user, even in the case of errors or unexpected inputs. By managing these edge cases effectively, the system can maintain a smooth and user-friendly experience.

Gibberish Test

In cases where the user submits a question or input that appears nonsensical or incomprehensible, such as "sdkfjnsdjfbs," the system performs a gibberish test to assess the quality of the text. If the input fails the test, the system responds with a polite message: *"I'm sorry, I couldn't understand that. Could you rephrase your question?"* This ensures that the user is aware of the issue and can refine their query for better interaction, maintaining the quality and relevance of the chatbot's responses.

Handling Multiple Detected Objects

When an uploaded image contains multiple detectable objects, the system notifies the user and presents a list of all detected objects. The user is then prompted to select one object from the list for further interaction. Once the user selects their desired object, the system initiates the question-answering process, tailoring its responses to the selected object. This approach ensures precise and context-specific interactions, even in scenarios involving complex images.

Object Not Found

If the user uploads an image in which no recognizable objects are detected, the system gracefully handles the situation by sending a message: *"Object not found."* This ensures the user is informed about the issue and can upload a different image or adjust their input, maintaining a smooth and user-friendly experience.

5.7.4 Session for Storage

Session storage in the project maintains temporary data between the client (browser) and the server for the duration of an active session. The session is identified using a unique session ID stored as a cookie on the client side. This ID links the client to session data stored on the server, often in a database. Key session variables include `chat_history` (conversation context), `yolo_result` (object detection results from YOLO), `bart_result` (NLP results from BART), and `detected_objects` (list of identified objects).

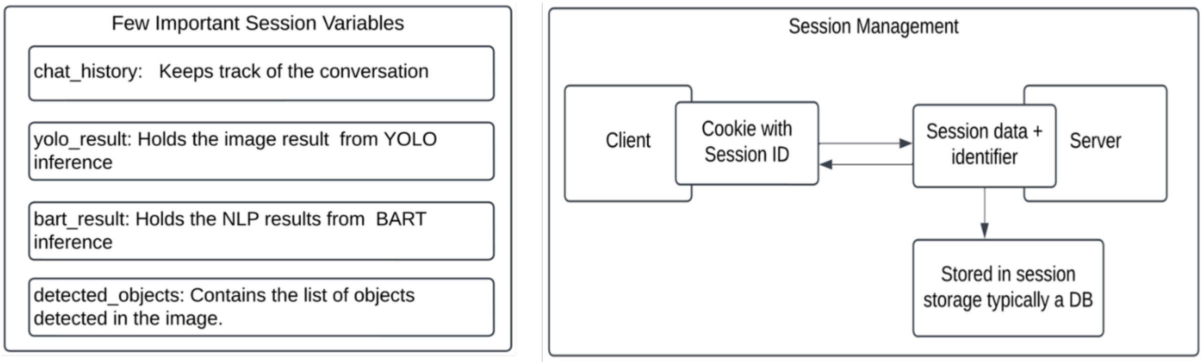

Few Important Session Variables
chat_history: Keeps track of the conversation
yolo_result: Holds the image result from YOLO inference
bart_result: Holds the NLP results from BART inference
detected_objects: Contains the list of objects detected in the image.
Session Management
Client
Cookie with Session ID
Session data + identifier
Server
Stored in session storage typically a DB

Figure 5-10: Session Management

5.7.5 Software Work Flow Diagram

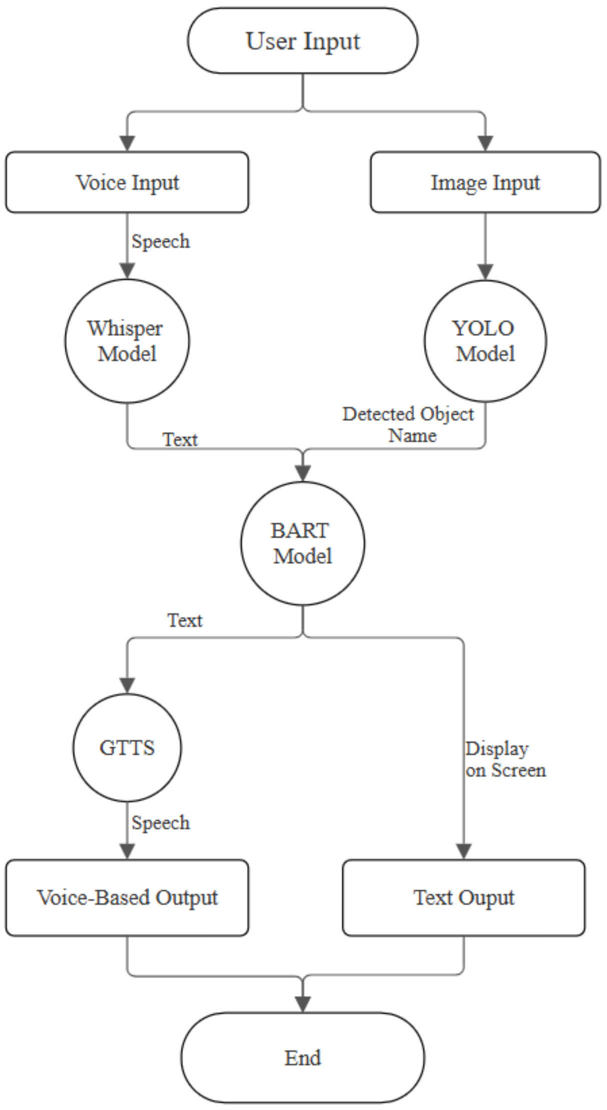


Figure 5-11: Software Work Flow Diagram

The work flow diagram depicts the flow of data in the web application system for processing user inputs and providing outputs. The system accepts two types of inputs: voice input and image input. Voice input is processed through the Whisper model to convert speech to text, while image input is analyzed using the YOLO model to detect objects and extract object names. Both data streams are forwarded to the BART model, which generates a contextually appropriate response in text form. The response is either converted to speech using the GTTS (Google Text-to-Speech) library for voice-based output or displayed on the screen as text output. The flow concludes with the delivery of the output to the user.

5.7.6 Time Complexity

The performance of the system was evaluated by measuring the time taken for both YOLO Inference (image processing) and BART Text Generation (question answering) using 10 images and 10 questions. The results provided insights into the time complexity and efficiency of these operations.

5.7.6.1 YOLO Inference Time Complexity

YOLO (You Only Look Once) is an efficient object detection model that performs inference on an image in a single pass. The time for processing each image was recorded, and the average time for image inference across 10 images was calculated.

Average Image Inference Time: 0.1519 seconds

The time taken for each image inference varies depending on the complexity of the image, but on average, it is quite fast, making YOLO well-suited for real-time object detection tasks. The time complexity of YOLO inference can be considered **O(n)**, where **n** is the number of pixels in the image. However, since YOLO processes the image as a whole, the time taken per image remains relatively constant, with the inference time not being heavily influenced by the number of objects or specific content in the image.

5.7.6.2 BART Text Generation Time Complexity

BART (Bidirectional and Auto-Regressive Transformers) is used for generating responses to questions. The time for text generation was measured for 10 different questions, and the average time taken for generating a response was computed.

Average Question Generation Time: 1.3622 seconds

Text generation with BART generally takes longer than image inference due to the complexity of processing natural language and generating relevant responses. The time complexity of BART text generation can be described as $O(m * d)$, where m is the length of the input (the question) and d is the depth of the transformer layers in the model. Longer questions tend to require more time for the model to generate a response.

5.7.6.3 Measurement Methodology

The time for each operation was measured using Python's `time.time()` function, which allows precise tracking of start and end times for both the image inference and text generation steps. Specifically:

For YOLO Inference: The time was measured by recording the start and end time for the object detection process on each image.

For BART Text Generation: The time was recorded by measuring the time taken for the entire process of question answering, including preparing the input and generating the output.

The average times for both operations demonstrate that YOLO inference and BART text generation are both efficient for handling image and question processing tasks:

Average Image Inference Time (YOLO): 0.1519 seconds

Average Question Generation Time (BART): 1.3622 seconds

These results reflect the balance between fast object detection and more computationally intensive natural language processing. The system performs well within expected timeframes, making it suitable for real-time interactions when handling both images and textual data.

6. RESULT AND ANALYSIS

6.1 YOLOv8 Training Results

YOLOv8 model was trained on the custom dataset using Kaggle's free GPU and good results were obtained. The following section shows the loss curves, precision, recall, and confusion matrix for the trained model.

6.1.1 Feature Maps

The feature maps presented in following figure provide insights into the hierarchical feature extraction mechanism employed by the YOLOv8 model. Each row corresponds to a distinct layer of the neural network, highlighting the progressive transformation of visual information as it flows through the model. The first row illustrates feature maps from Layer 1, while the second row shows feature maps from Layer 20, revealing the model's capability to extract and refine information for object detection.

In Layer 1, the feature maps capture low-level features such as edges, textures, and simple patterns. These initial feature maps exhibit dense activations, with sharp contrasts highlighting boundaries and textures present in the input image. For example, the first map in this row emphasizes structural details like edges, while the second and third maps show how the network detects simple variations in brightness and contrast. These low-level features form the foundational building blocks necessary for understanding the spatial layout and textural properties of objects in the image.

As we move to Layer 20, the feature maps exhibit a dramatic shift toward high-level, abstract representations. These deeper layers focus on identifying specific regions or features relevant to the detection of objects, with sparser activations isolating task-specific patterns. The feature maps in this layer are more diffuse, reflecting the model's abstraction of complex visual information into representations that are critical for classification and localization. For instance, the activations in Layer 20 appear as clusters, pinpointing areas of interest that correspond to distinct parts of an object or an entire object itself.

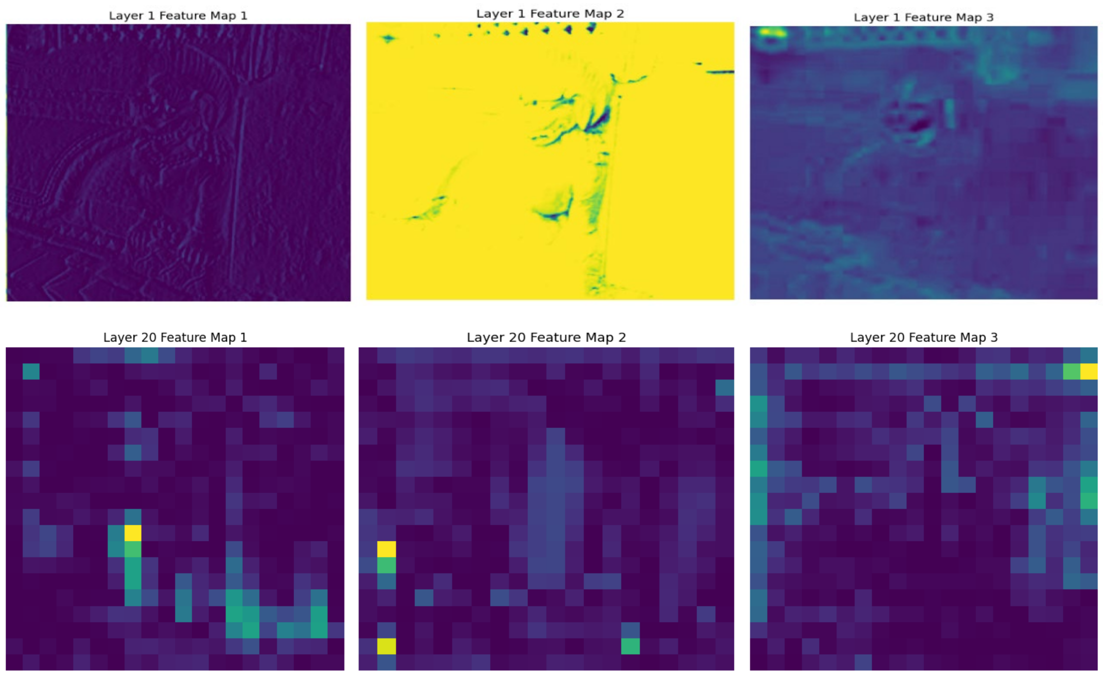


Figure 6-1: Feature Maps

6.1.2 Training and Validation Loss Curves

- **Train/Val Box Loss:**

The train/val box loss measures how well the model predicts object bounding boxes. A declining train/val box loss indicates that the model is improving its accuracy in predicting object bounding boxes, reflecting better localization performance.

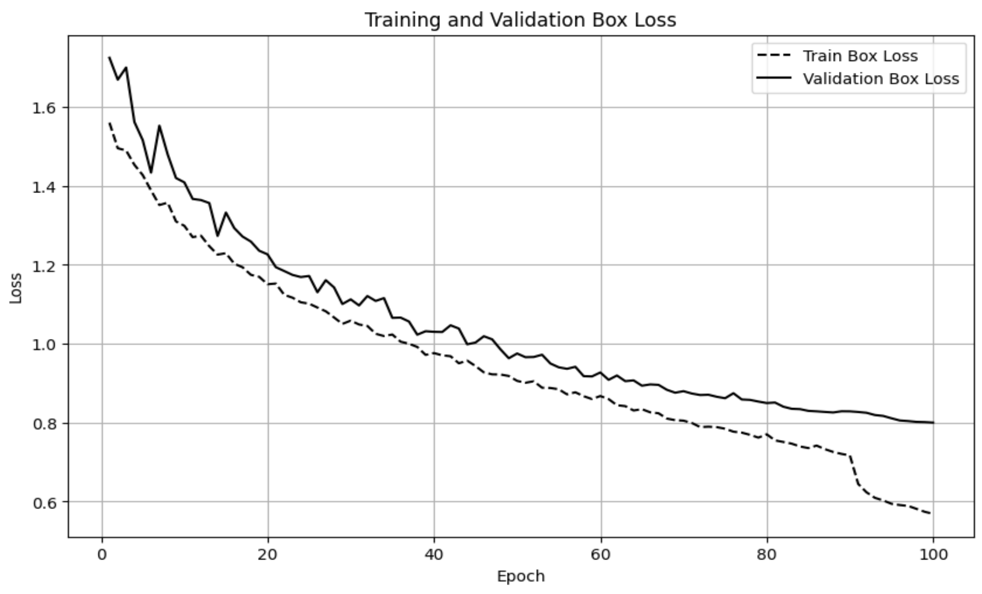


Figure 6-2: Training and Validation Box Loss

- **Train/Val Class Loss:**

The train/val class loss represents the model's performance in classifying objects correctly in the validation dataset. The class loss measures the discrepancy between the predicted class probabilities and the true class labels. A declining train/val class loss indicates that the model is improving its classification accuracy over time, showing effective learning and better generalization.

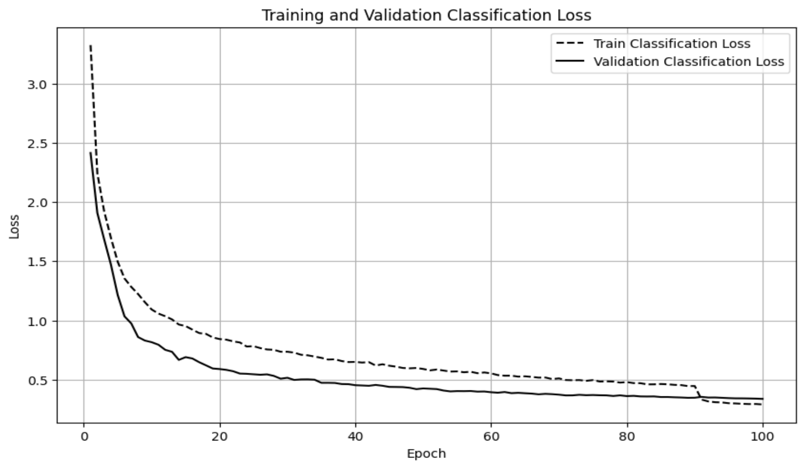


Figure 6-3: Training and Validation Class Loss

- **Train/Val Dfl Loss:**

DFL (Distribution Focal Loss) loss is designed to improve the accuracy of object detection by focusing on hard-to-classify examples. A declining DFL loss indicates improved accuracy in predicting detection scores, reflecting better overall object detection performance.

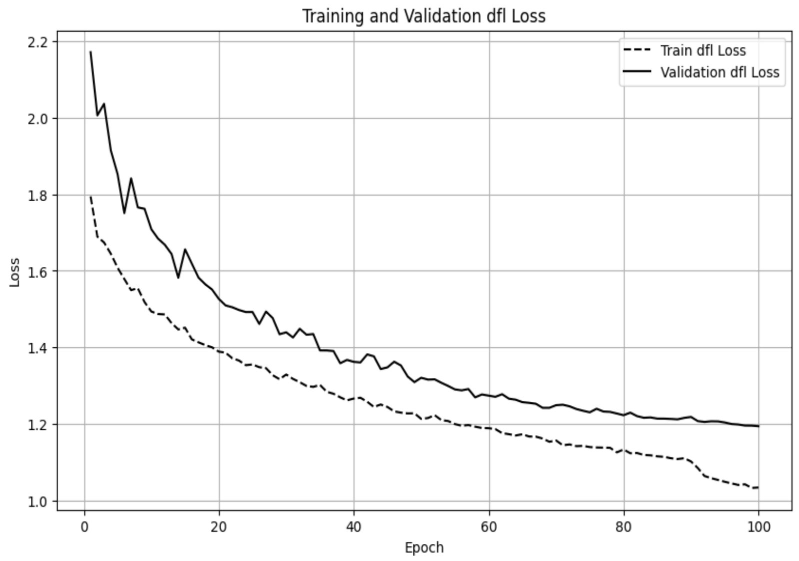


Figure 6-4: Train and Validation dfl Loss

6.1.3 Confusion Matrix

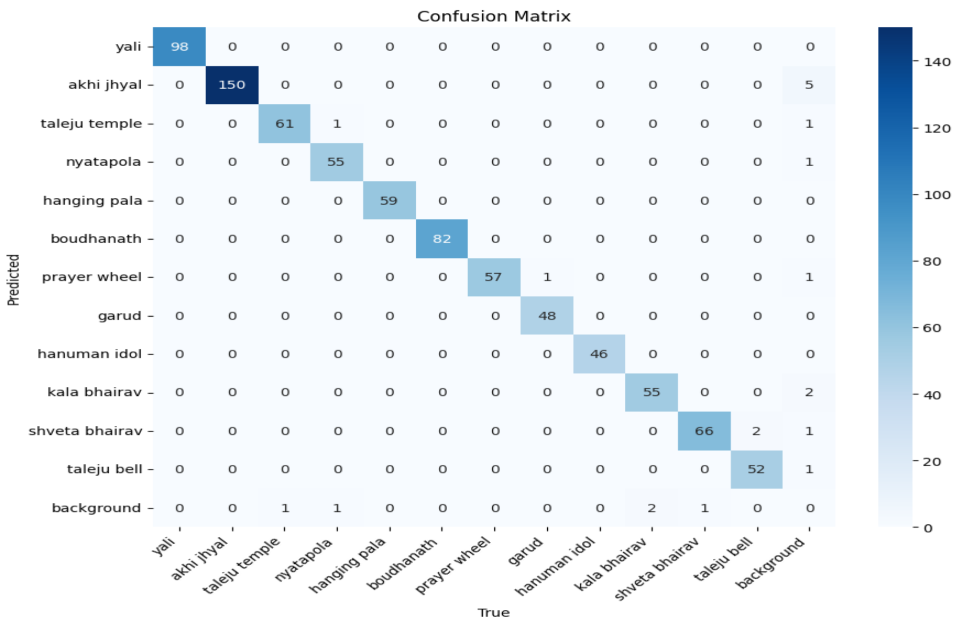


Figure 6-5: Confusion Matrix

The confusion matrix evaluates the classification model's performance across different classes. The confusion matrix indicates that all classes are classified with high accuracy and very few errors, except for the 'background' class. The 'background' class, intended to capture non-relevant instances or just the background, which is misclassified multiple times across different categories, including 5 times as 'akhi jhyal' and once into classes such as ‘taleju temple’, ‘nyatapola’, ‘prayer wheel’, ‘kala bhairav’ etc. This indicates that the model slightly struggles to correctly identify instances that should be categorized as 'background,' leading to some misclassifications. Despite these minor misclassifications with the 'background' class, the overall performance of the model remains accurate.

6.1.4 Precision Recall Curve

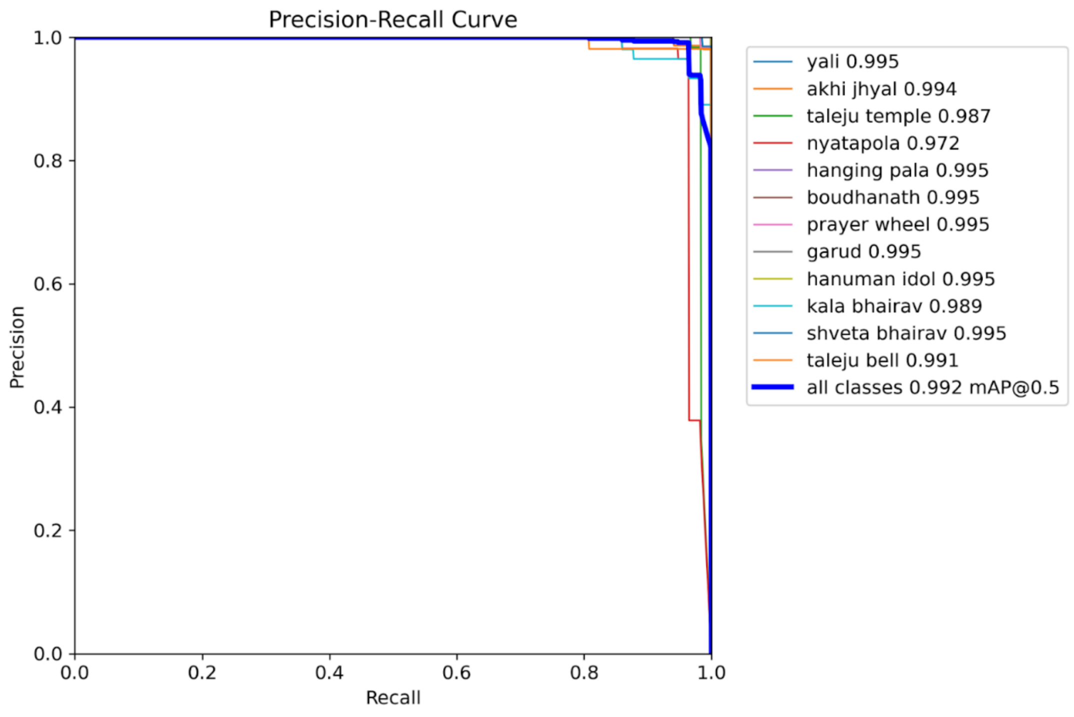


Figure 6-6: Precision Recall Curve

The precision-recall curve illustrates the trade-off between precision (the accuracy of positive predictions) and recall (the ability to find all positive instances). The curve shows that the model performs exceptionally well across most classes, with precision and recall values close to 1.0. Classes like 'yali', 'akhi jhyal', 'hanging pala', 'prayer wheel', 'garud', 'hanuman idol', and 'shveta bhairav' exhibit very high precision and recall, indicating accurate and comprehensive identification of positive instances. While the 'nyatapola' class has a slightly lower precision of 0.972. Overall, the model demonstrates strong performance across all classes, with an average precision (mAP@0.5) of 0.992, indicating robust and reliable classification.

6.1.5 F1 Curve

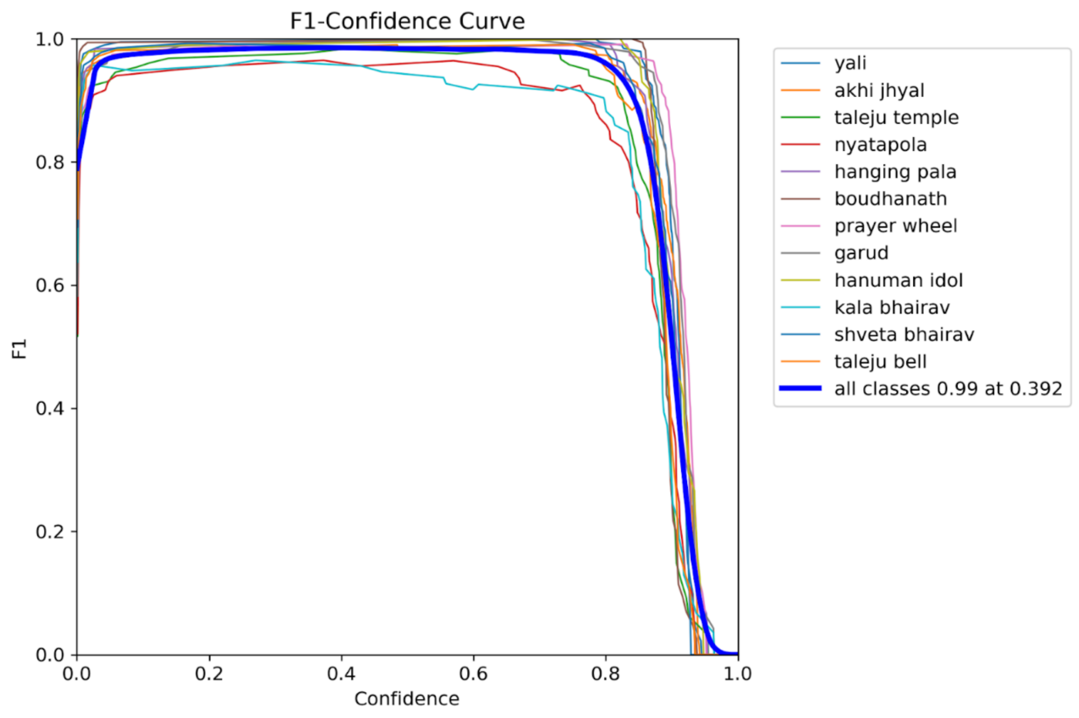


Figure 6-7: F1 Curve

The F1-confidence curve you provided illustrates the relationship between the confidence of predictions and their corresponding F1 scores across different classes in a classification problem. The x-axis represents the confidence level of the model's predictions, ranging from 0 to 1, while the y-axis shows the F1 score, a balance of precision and recall. Each curve represents a different class, with the legend on the right identifying these classes, such as 'yali', 'akhi jhyal', and 'taleju temple'. The thick blue curve labeled "all classes" represents the overall performance, showing an average F1 score of 0.99 at a confidence level of 0.392. This means that even when the model is only moderately confident in its predictions (confidence around 39.2%), it still achieves very high accuracy, which is an impressive performance.

6.1.6 mAP50 and mAP(50-95)

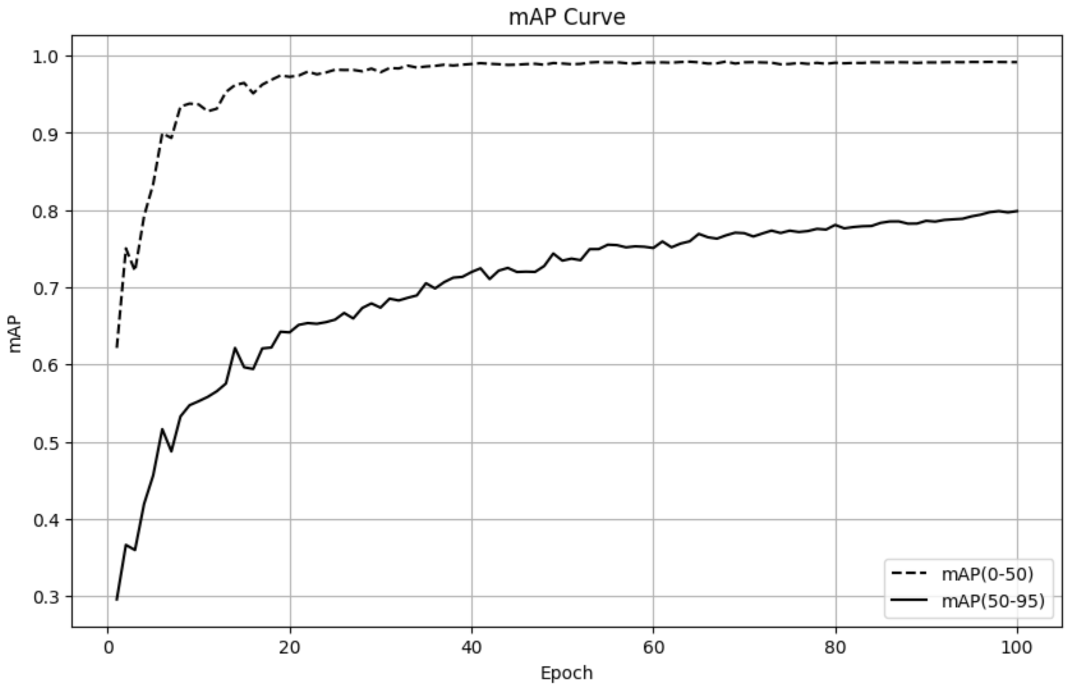


Figure 6-8: mAP50 and mAP(50-95)

The curve has two metrics: mAP50 and mAP(50-95). mAP50, represented by the dashed line, measures mean Average Precision at an IoU threshold of 0.5, showing how well the model detects objects with at least 50% overlap. The curve shows a rapid initial increase, indicating quick learning, and plateaus early, suggesting the model achieves near-optimal performance quickly. The solid line represents mAP(50-95), which averages precision across IoU thresholds from 0.5 to 0.95, providing a more stringent evaluation. This curve starts lower and rises more gradually, reflecting the greater challenge of achieving high precision with stricter overlap requirements.

6.2 Hyperparameter Tuning Results

The graphs obtained for hyperparameter tuning represent the performance of various parameter combinations across the trials. Each graph is assigned a unique name and color, making it easy to distinguish between different hyperparameter configurations.

6.2.1 Fuzzy Match Accuracy

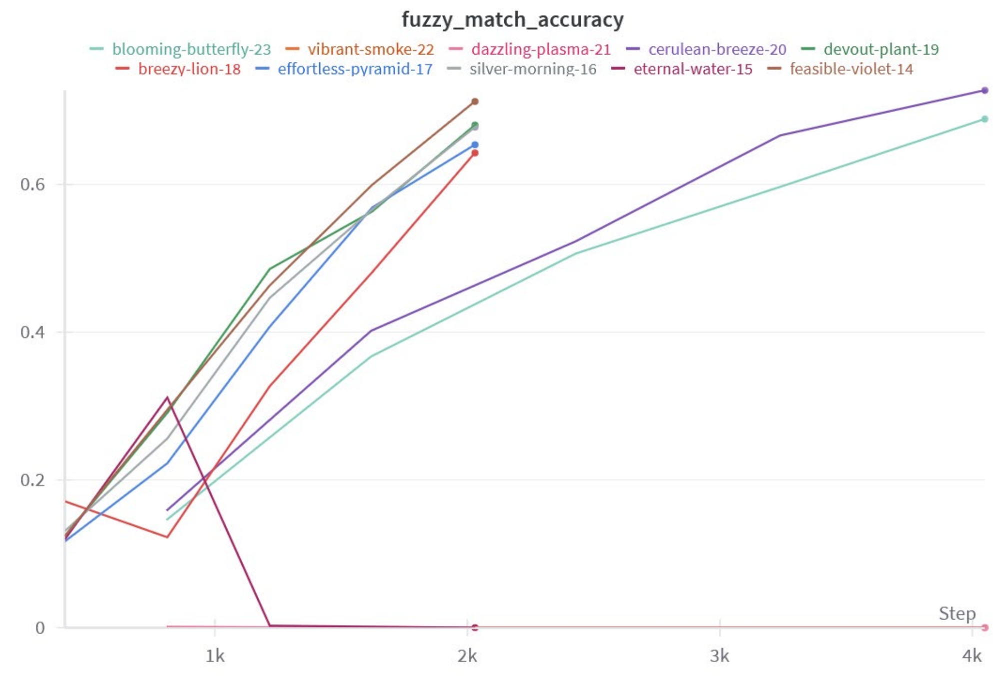


Figure 6-9: Fuzzy Match Accuracy

Over 4,000 steps, corresponding to 5 epochs in this case, different combinations of hyperparameters resulted in varying accuracies as measured by the fuzzy match accuracy metric. Among these combinations, the highest accuracy achieved was approximately 65% within the 5-epoch evaluation window. This demonstrates how the choice of hyperparameters significantly influenced the model's performance during the tuning process.

6.2.2 Meteor Score

Over 4,000 steps, corresponding to 5 epochs, different combinations of hyperparameters yielded varying METEOR scores, reflecting the diversity in performance across trials. While the highest METEOR score achieved was approximately 65%, some combinations performed significantly worse, with scores falling well below 30%. These results underscore the impact of hyperparameter selection on the model's ability to align with the objective function and highlight the importance of systematic tuning to identify the optimal configuration.

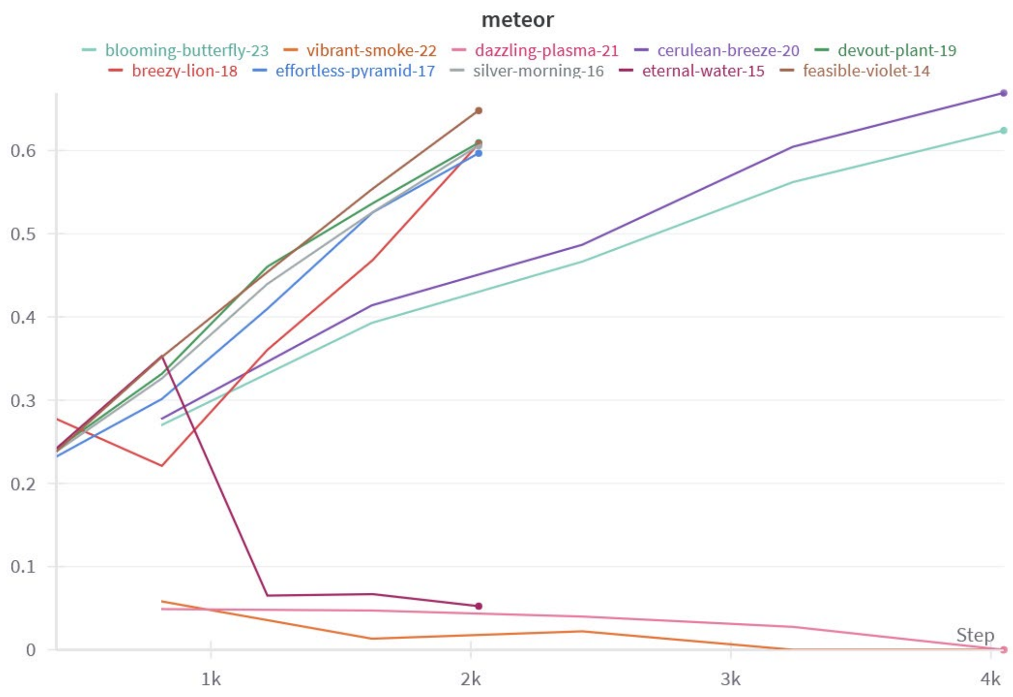


Figure 6-10: Meteor Score

6.2.3 Semantic Similarity

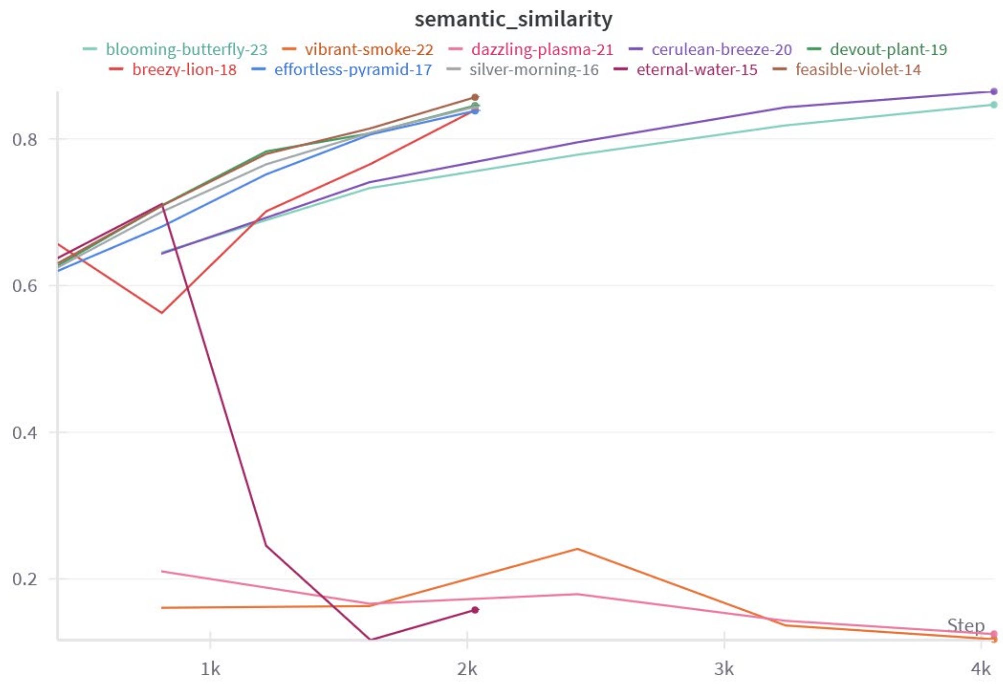


Figure 6-11: Semantic Similarity

Over 4,000 steps and 5 epochs, different combinations of hyperparameters resulted in varying semantic similarity scores. The highest semantic similarity score achieved was around 85%, indicating strong alignment with the desired outputs. However, some combinations performed poorly, with scores dropping below 20%.

6.2.4 Validation Loss

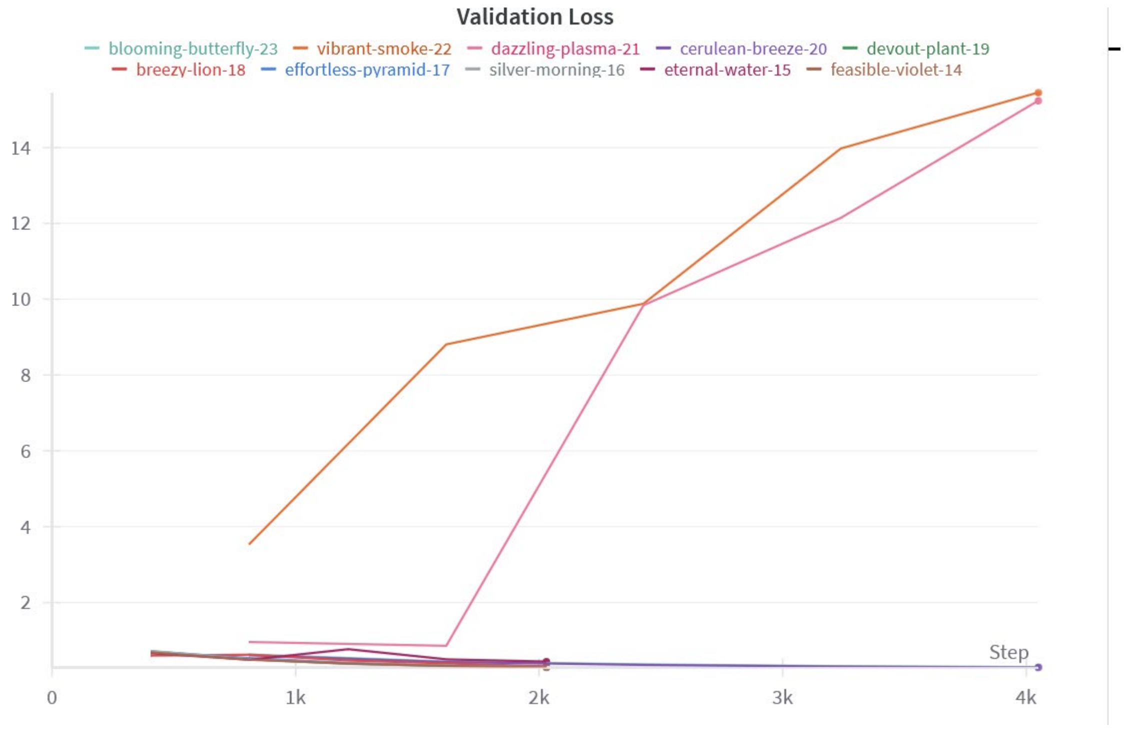


Figure 6-12: Validation Loss

Some hyperparameter combinations exhibited very high validation losses, reaching up to 14%, others achieved significantly lower values. The hyperparameter combinations with the smallest validation loss were prioritized and chosen, as they indicated better generalization and model performance on unseen data.

6.3 BART Training Results

The results obtained after fine-tuning the BART-base model on the custom dataset were promising. The following section presents these outcomes, including metrics such as ROUGE, BLEU, loss, and semantic similarity, demonstrating the effectiveness of the fine-tuning process.

6.3.1 Tokenization

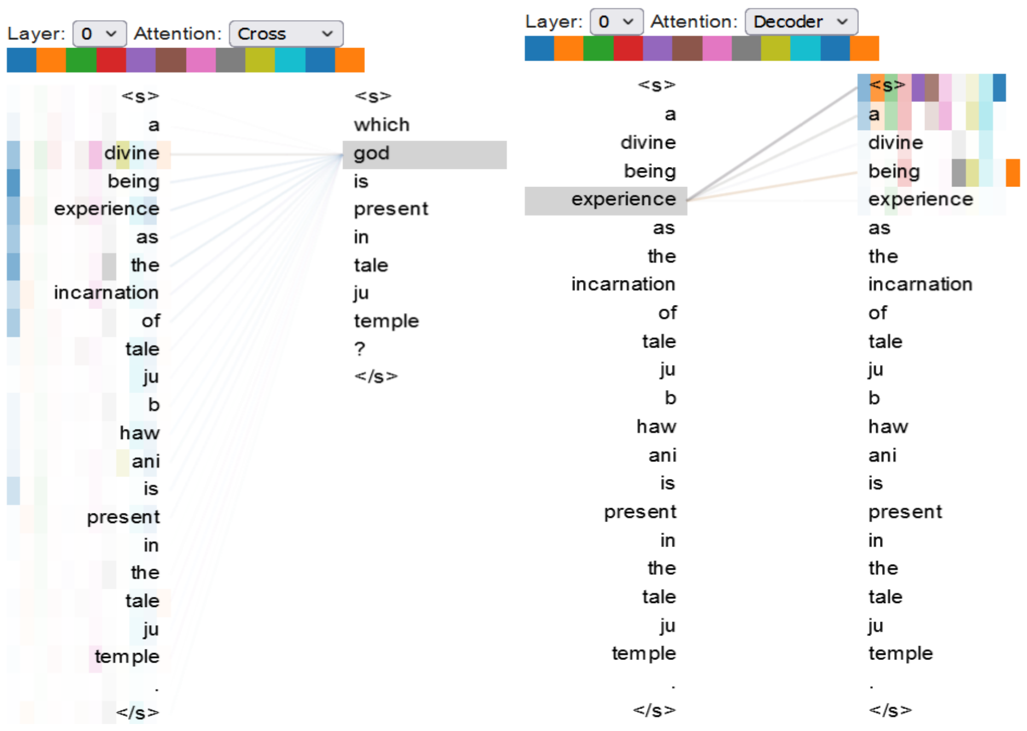
Byte Pair Encoding (BPE) is a subword tokenization method used in BART to break down words into smaller units, called tokens, by iteratively merging the most frequent pairs of characters or subwords. This approach allows BART to handle rare or unseen words by splitting them into smaller, known tokens, which improves its ability to generalize across different languages and domains.

For example, the sentence "Which god is present in Taleju temple?" is tokenized as:

['<s>', 'which', 'Ġgod', 'Ġis', 'Ġpresent', 'Ġin', 'Ġtale', 'ju', 'Ġtemple', '?', '</s>']

Here, "Taleju" is split into "tale" and "ju" because they are more frequent subwords compared to the full word "Taleju," which may not have been seen during training. The "Ġ" symbol before some tokens indicates a space at the start of those words.

6.3.2 Attention Maps


Layer: 0 Attention: Cross

Layer: 0 Attention: Decoder

<s> a divine being experience as the incarnation of tale ju b haw ani is present in the tale ju temple . </s>
<s> which god is present in tale ju temple ? </s>
<s> a divine being experience as the incarnation of tale ju b haw ani is present in the tale ju temple . </s>
<s> a divine being experience as the incarnation of tale ju b haw ani is present in the tale ju temple . </s>

Figure 6-13: Attention Maps

The images provide a detailed visualization of the attention mechanisms in a BART model, illustrating how the model processes both the input question and the generated answer. In the first image, which depicts cross-attention, the colorful patches represent attention heads that align parts of the input with specific parts of the output. The lines indicate the strength of the attention between words in the question and the generated text. Notably, the word "God" in the question shows a slightly higher attention score when aligned with the word "divine" in the input sequence. This demonstrates the model's capacity to recognize and associate semantic relationships, such as linking "God" with the concept of divinity. Moreover, different attention heads capture varying aspects of the context, suggesting that the model leverages multiple perspectives to form a nuanced understanding of the word "God" within the given question.

In the second image, the attention relationships within the generated text are visualized, highlighting how each token depends on both the input context and previously generated tokens. The word "experience," for instance, shows strong attention connections to earlier generated tokens, reflecting the autoregressive nature of the BART model. This autoregressive property ensures that each generated word is contextually informed, not only by the input sequence but also by the sequence of words generated so far. As a result, the model can maintain coherence and relevance throughout the output.

6.3.3 Training and Validation Loss Curves

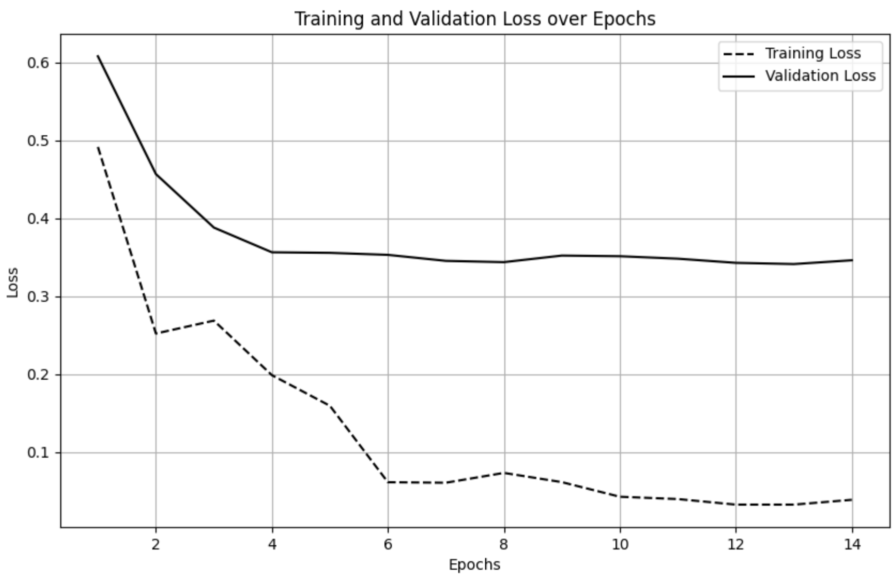


Figure 6-14: Training and Validation Loss Curves

The training loss decreases consistently, indicating effective learning. The validation loss initially decreases but plateaus around epoch 7 and slightly increases afterward, suggesting overfitting begins beyond this point. Observing the performance until epoch 14, the best model was selected based on the epoch with the lowest validation loss to ensure a balance between learning and generalization without overfitting.

6.3.4 Fuzzy Match Accuracy

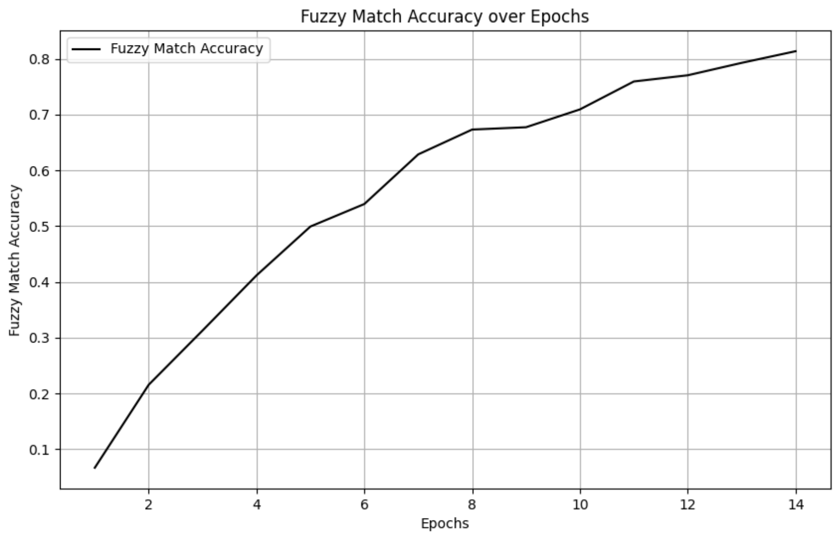


Figure 6-15: Fuzzy Match Accuracy

The accuracy demonstrates a steady and consistent increase throughout the training process, ultimately reaching approximately 80%, which highlights the model's growing ability to predict answers that closely align with the ground truth. During the initial epochs, the accuracy curve exhibits a sharp rise, signifying the rapid learning phase where the model captures fundamental patterns and relationships in the dataset. This steep improvement is indicative of the model effectively leveraging the training data to reduce error and align its predictions with expected outcomes.

6.3.5 Semantic Similarity

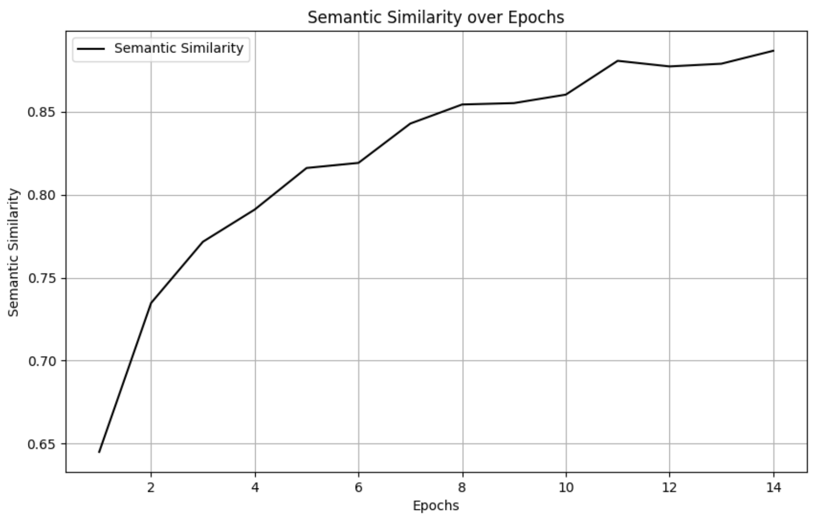


Figure 6-16: Semantic Similarity

The similarity score shows a steep increase in the initial epochs, followed by a more gradual improvement after epoch 6. The highest semantic similarity score was achieved at epoch 14, which is around 86% demonstrating the model's ability to generate responses semantically aligned with the ground truth.

6.3.6 ROUGE Score

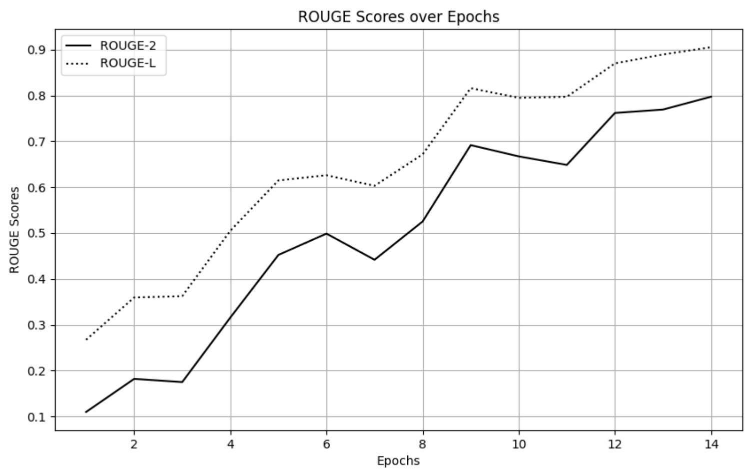


Figure 6-17: ROUGE Score

ROUGE-2 and ROUGE-L display a consistent upward trend, indicating improved alignment of generated answers with the ground truth. However, fluctuations are observed mid-training, which can be attributed to the metrics' limited ability to account for paraphrases and synonyms, leading to variability in the scores.

6.3.7 BLEU Score

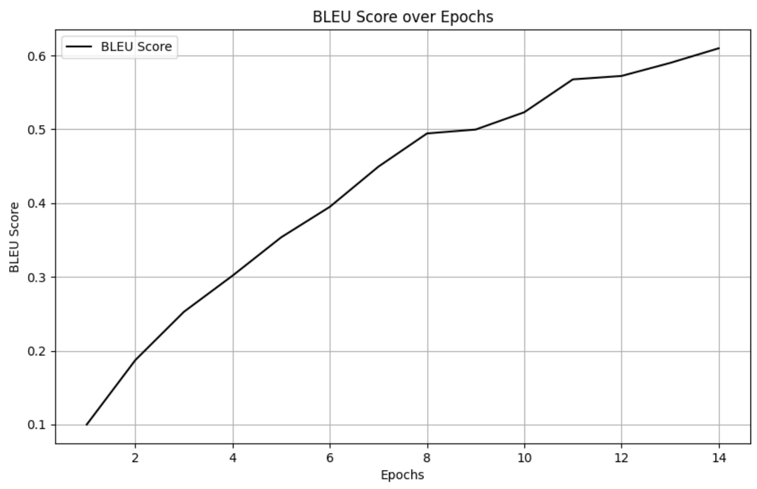


Figure 6-18: BLEU Score

BLEU score shows steady improvement, with significant gains in the earlier epochs and more gradual growth in later epochs. Similar to ROUGE, fluctuations in BLEU scores are observed due to its inability to fully consider paraphrases and synonyms, which impacts the consistency and slightly low of the metric.

6.3.8 METEOR Score

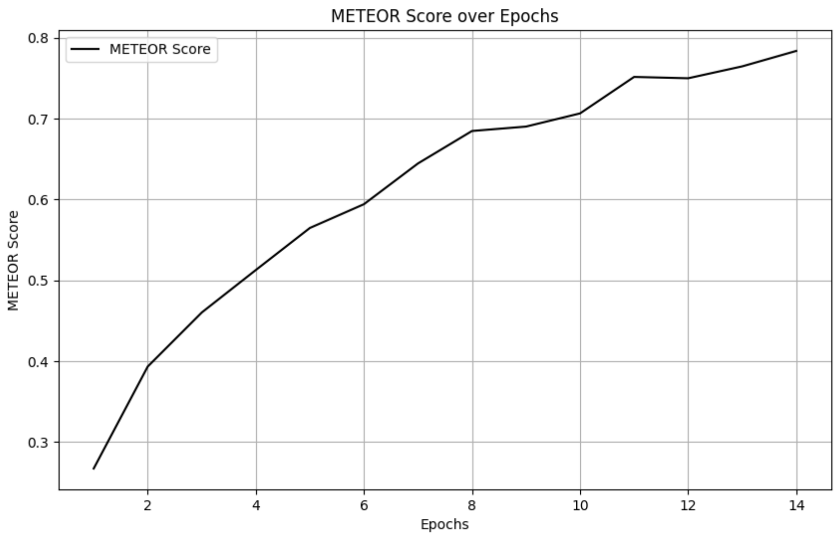


Figure 6-19: METEOR Score

The METEOR score of the BART-base model gradually increases throughout the 14 epochs, reaching up to 79% by the end of training. Unlike ROUGE and BLEU, METEOR accounts for paraphrasing and stemming, enabling it to better evaluate the semantic alignment between the generated and reference responses.

6.4 Comparison of Different NLP Models for Question-Answering

Different variants of models were evaluated to identify the best model for the question-answering task. Among the BART variants, models such as BART-large and BART-base were tested, along with other models like T5-small. After comparing their performance, BART-base emerged as the best-performing model, delivering the most optimal results for the task. This systematic evaluation ensured that the selected model was well-suited for achieving high accuracy and efficiency in question answering.

Evaluation Metrics	T5 Model (values)	Bart-large (values)	Bart-base (values)
BLEU Score	0.371	0.273	0.609
ROUGE-L Score	0.72	0.392	0.905
Fuzzy Match Accuracy	0.620	0.254	0.813
Validation Loss	0.403	0.732	0.345
Train Loss	0.053	0.187	0.030

Among the three selected models, BART-base performed the best for the question-answering task. While BART-large, being a larger model with more parameters, showed comparatively lower performance, this may have been due to the training dataset not being sufficient to fully leverage its larger capacity. On the other hand, T5-small achieved comparatively better results than BART-large. However, BART-base still outperformed both BART-large and T5-small, making it the optimal choice for the task. This highlights the balance between model size, available data, and task-specific performance.

Comparison Graphs

- **Fuzzy Match Accuracy**

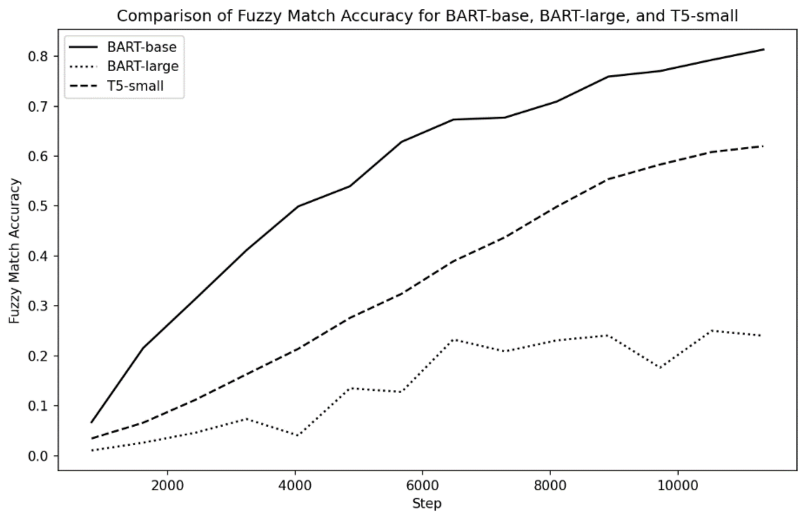


Figure 6-20: Fuzzy Match Accuracy

The comparison of the models reveals that BART-base achieved the best accuracy, with a stable and higher accuracy of approximately 80%. T5-small demonstrated comparatively smoother accuracy improvements, reaching around 60%. On the other hand, BART-large exhibited fluctuating accuracy, with a significantly lower value of around 20%. These results highlight the superior performance of BART-base for this task, while T5-small showed consistent but less impressive results, and BART-large struggled with instability and poor performance.

- **Validation Loss**

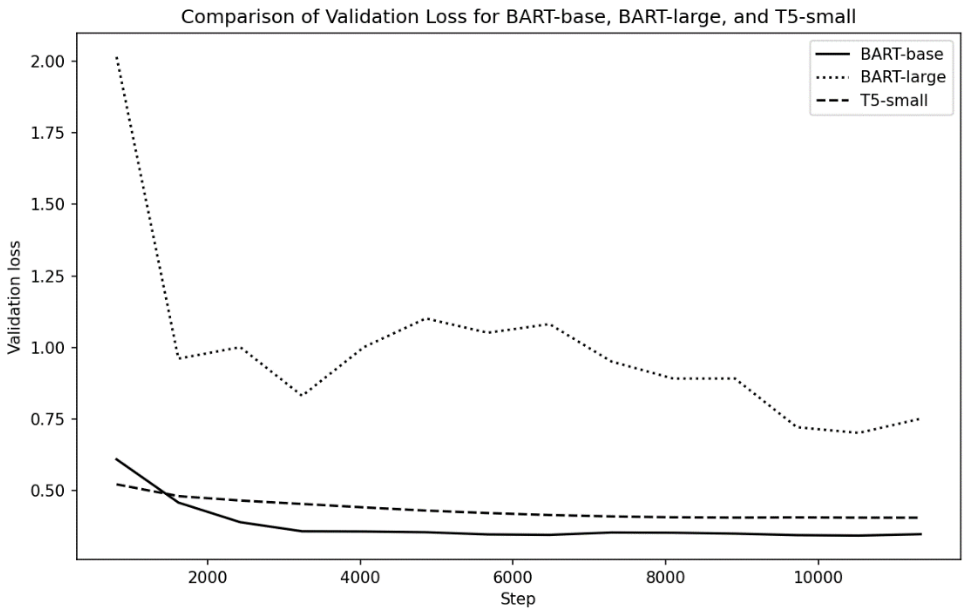


Figure 6-21: Validation Loss

The validation loss for BART-large shows significant fluctuations throughout the training process, indicating that the model struggles to stabilize its performance. In contrast, both BART-base and T5-small demonstrate relatively stable validation loss curves, suggesting that these models are more consistent in their performance across training steps. Given this behavior, BART-large may not be the ideal choice for this task, as its unstable validation loss could lead to less reliable performance.

6.5 Inference Results

The following results showcase the performance of the YOLOv8 object detection model, which has demonstrated remarkable accuracy in identifying and localizing objects. Each detected object is enclosed within a bounding box, with a floating-point number displayed in the top-right corner of the box. This number represents the confidence score, indicating the model's certainty about the detection. The high confidence scores highlight the model's reliability and its ability to make precise predictions, underscoring its strong performance in object detection tasks.

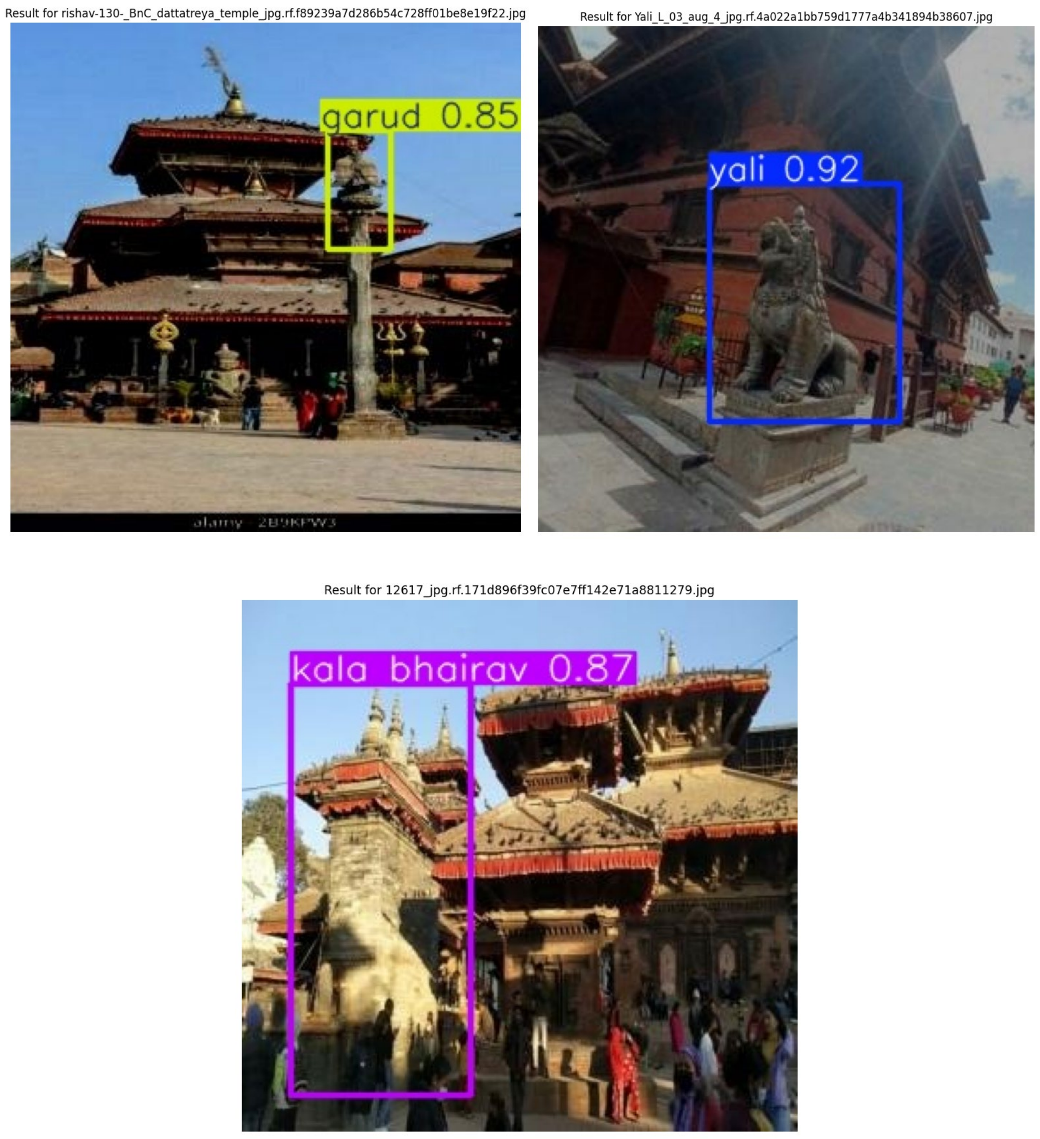

Result for rishav-130-_BnC_dattatreya_temple_jpg.rf.f89239a7d286b54c728ff01be8e19f22.jpg
garud 0.85
Result for Yali_L_03_aug_4_jpg.rf.4a022a1bb759d1777a4b341894b38607.jpg
yali 0.92
Result for 12617_jpg.rf.171d896f39fc07e7ff142e71a8811279.jpg
kala bhairav 0.87

Figure 6-22: YOLOv8 Inference Results

In the below image, the results generated by the BART model are presented. These results are semantically accurate and demonstrate the model's ability to comprehend and respond appropriately to the context of the given questions. This highlights the model's strong generative capabilities and its effectiveness in producing meaningful and contextually relevant outputs.

Table 6-1: BART Inference Results

Question	Original Answer	Generated Answer
what does the phrase 'turning the wheel of dharma' describe ?	it describes the way in which the buddha taught, and the prayer wheel is a physical manifestation of this concept.	it describes the way in which the buddha taught, and the prayer wheeled is a physical manifestation of this notions.
what is the alternative term used by newar inhabitants outside of bhaktapur when referring to the temple ?	people of newar descent living away from bhaktapur commonly refer to the temple as 'nyātāpau.'	outside bhaktapur, the newar people use the term 'nyātāpau' to refer to the temple.
what direction do pilgrims stroll when turning prayer wheels at monasteries ?	pilgrims stroll in a clockwise direction when turning prayer wheels.	pilgrims stroll in a clockwise direction when turning prayer wheels.
what story involves garuda planning to kill and eat a snake ?	the story involves garuda planning to kill and eat the sumukha snake, where indra intervenes.	the story involves garuda planning to kill and eat the sumukha snake, where indra intervenes.
did rama reply to hanuman's plea ?	despite the odds, decided to hanuman's by him the gift of immortality.	despite the odds, rama opted to fulfill hanuman's plea by granting him the gift of immortality.
what advantage is believed to manifest insects that pass through the shadow of a wheel ?	legend has insects passing through the shadow of a prayer wheel may receive blessings.	legend has it that insects passing through the shadow of a prayer wheel may receive blessings.
how do yalis reflect the culture of the newar community ?	they symbolize local beliefs of strength and protection, representing newar cultural heritage.	they symbolize local notion of strength and protection, representing newar cultural heritage.
whar does marjara yali represent ?	marjara yali symbolizes guile and intellect, depicted with cat-like feature.	marjara yali symbolizes guile and cunning, combining a crafty intellect with the strength of a protector.

6.6 Speech to Text

With Custom Vocabulary Injection Based on WER of Objects

The graph reveals insights on the Whisper model's performance, measured by the average Word Error Rate (WER) across various objects when custom vocabulary injection is applied. Among the listed objects, "Kala Bhairava" has the highest WER at

0.641, indicating significant challenges in accurate transcription, likely due to linguistic complexity or the uniqueness of the term. Similarly, "Swet Bhairava" and "Yali" also show high WERs at 0.500 and 0.485, respectively, suggesting that certain names remain difficult for the model to process even with custom vocabulary assistance.

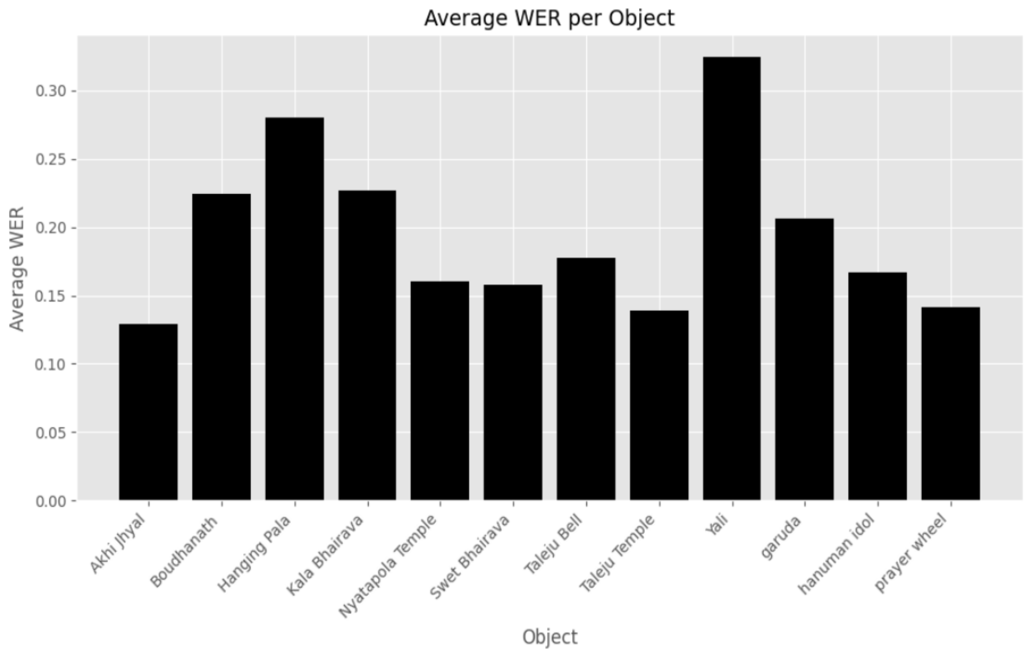


Figure 6-23: Average WER per Object (Custom Vocabulary)

In contrast, objects such as "Prayer Wheel" exhibit a much lower WER of 0.269, indicating that the model handles these terms more effectively, possibly due to their simpler phonetics or greater contextual familiarity. Moderately performing terms include "Akhi Jhyal," "Hanuman Idol," and "Taleju Temple," with WERs ranging between 0.377 and 0.458, showing some variability in transcription accuracy. Interestingly, widely recognized terms such as "Boudhanath" still present a relatively high WER of 0.440, highlighting areas where the model might benefit from further refinement in understanding culturally specific or phonetically diverse terms.

Based on WER in Accents

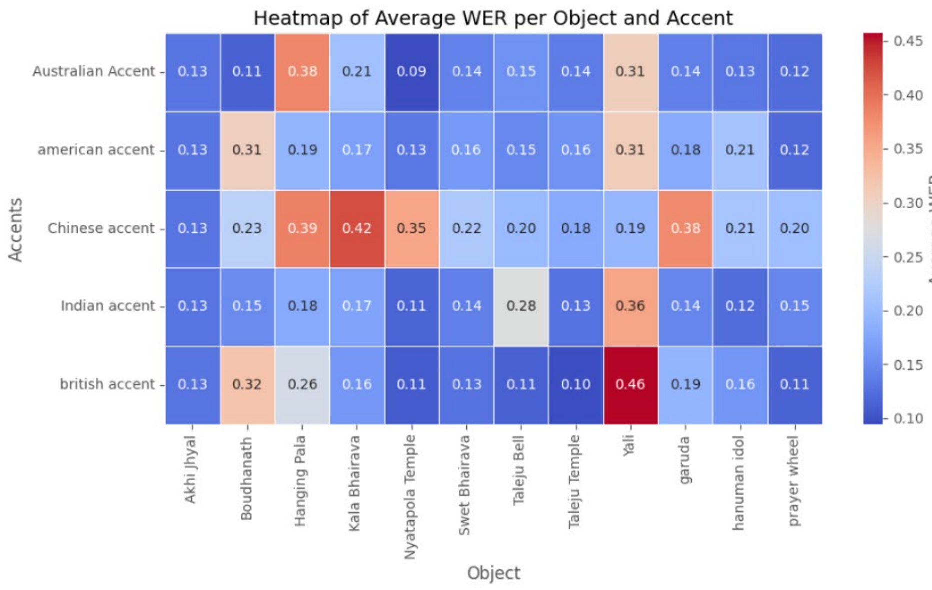


Figure 6-24: Average WER per Object and Accent (Custom Vocabulary)

The heatmap reveals that accents like the Australian and British accents generally achieve lower WERs, indicating stronger performance. For instance, the WER for "Prayer Wheel" is particularly low across all accents, such as 0.2308 for the Australian accent and 0.2647 for the British accent, suggesting that the model handles certain objects consistently well. However, objects like "Kala Bhairava" and "Yali" emerge as challenging across most accents, with "Kala Bhairava" showing a WER as high as 0.8117 for the Chinese accent and 0.5947 for the Australian accent. This pattern highlights a gap in the model's ability to recognize specific terms, likely due to their complexity or rarity in the training data. Similarly, accents like the American and Chinese accents exhibit greater variability in WER. The American accent struggles with objects like "Boudhanath" (WER 0.7186) and "Kala Bhairava" (0.6139), whereas the Chinese accent faces difficulties with "Kala Bhairava" and "Nyatapola Temple" (WERs of 0.8117 and 0.5269, respectively). This variability suggests that certain accent-object combinations pose greater challenges for the model, likely due to accent-specific phonetic nuances.

Without Custom Vocabulary Injection Based on WER of Objects

The bar graph reveals that "Kala Bhairava" stands out as the most challenging object for the model, with the highest average WER of 0.641. This consistently high error rate suggests a significant difficulty in accurately transcribing this term, likely due to its phonetic complexity or reduced representation in the model's training data. In contrast, the object "Prayer Wheel" has the lowest average WER of 0.269, showcasing the model's ability to handle simpler or more phonetically predictable terms with greater accuracy.

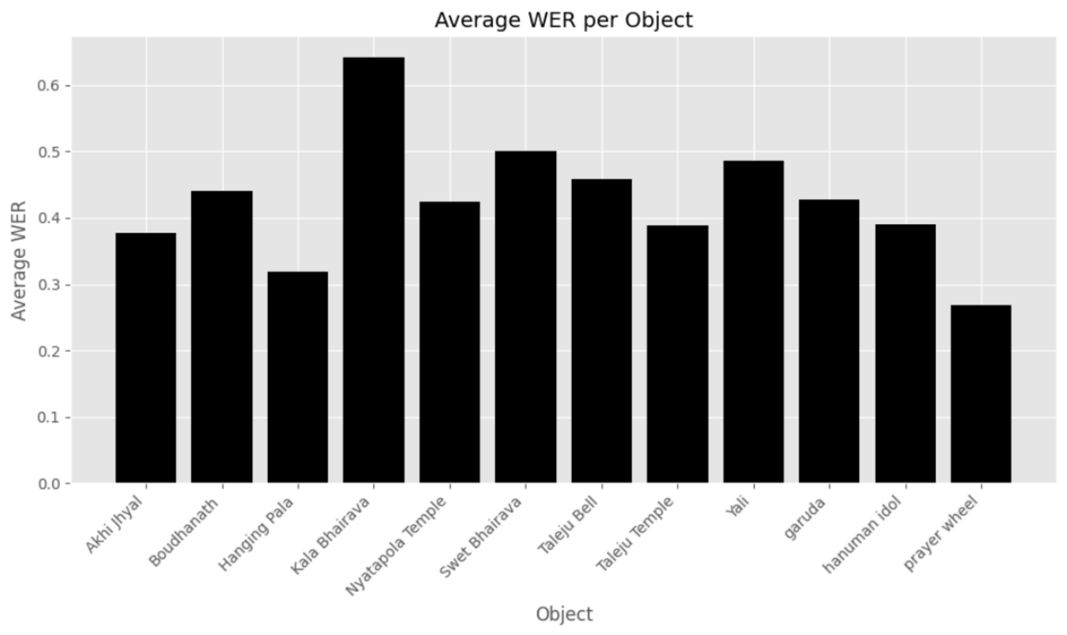


Figure 6-25: Average WER per Object

Intermediate levels of WER are observed for objects such as "Swet Bhairava" and "Yali," with average values of 0.500 and 0.485, respectively. These objects highlight moderate transcription challenges, possibly stemming from their linguistic characteristics or contextual ambiguity. Other objects like "Hanging Pala" and "Hanuman Idol" perform better, with average WER values of 0.319 and 0.391, respectively, indicating that the model encounters fewer difficulties with these terms. Terms such as "Boudhanath" and "Garuda" fall into a middle range, with WER values hovering around 0.441 and 0.427, suggesting a mix of moderate challenges and successful transcription attempts.

Based on WER in Accents

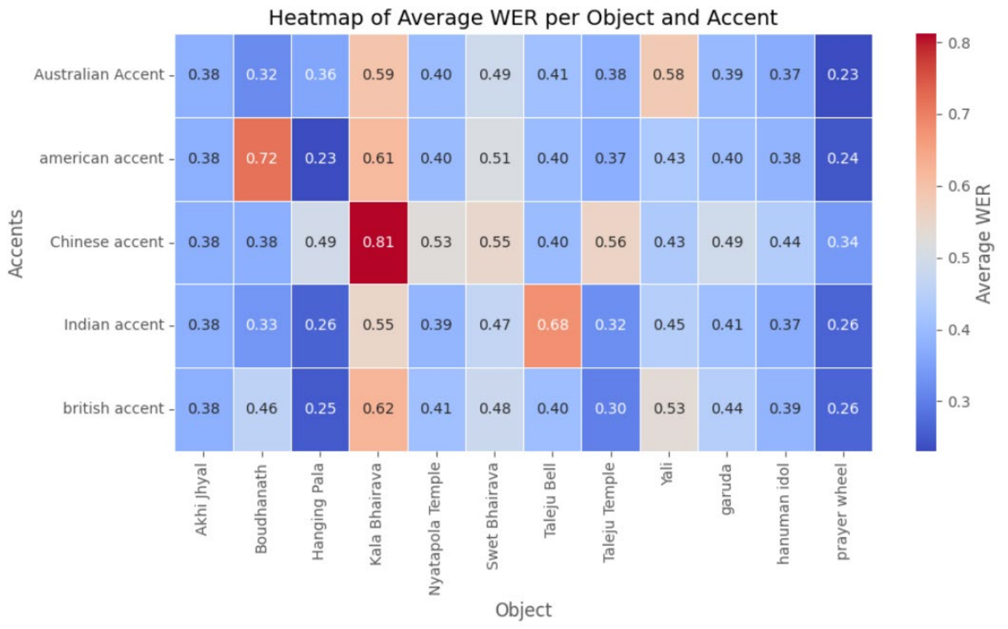
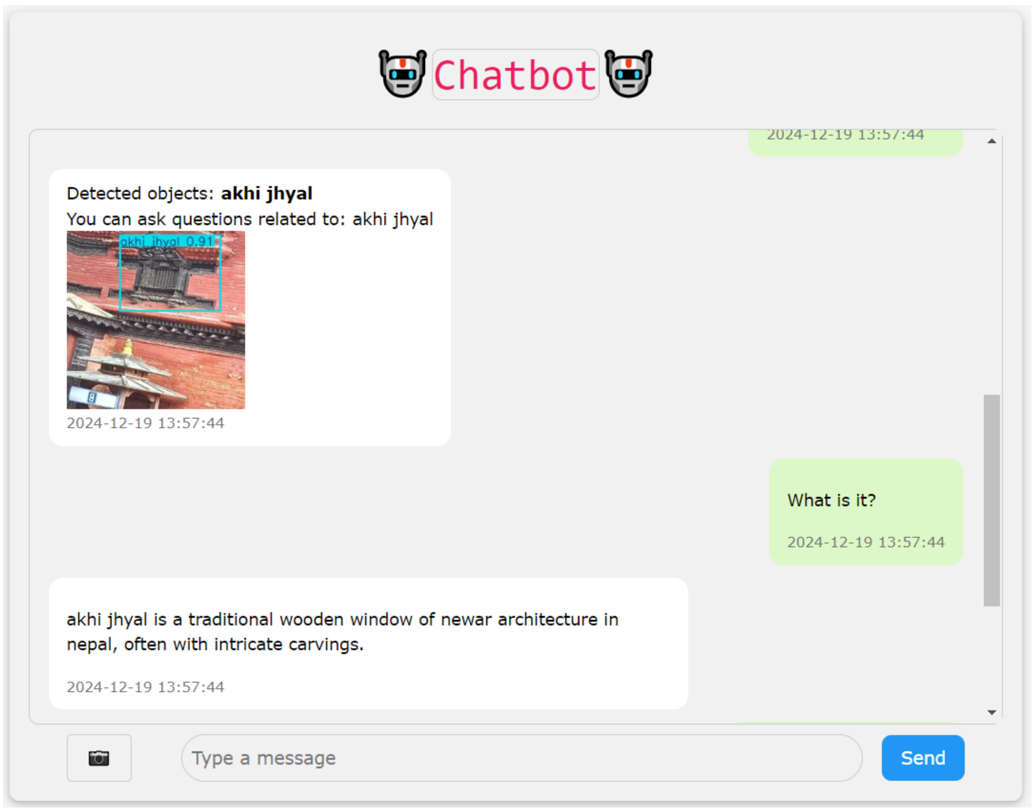


Figure 6-26: Average WER per Object and Accent

The analysis evaluates the performance of the Whisper model in transcribing audio questions about various objects spoken in different accents, without custom vocabulary injection. The results reveal a clear dependence on both accent and object, with significant variations in Word Error Rate (WER) across the dataset. The Australian and Indian accents generally exhibit lower WER values, indicating better transcription accuracy for these accents. In contrast, the Chinese accent consistently performs poorly, with the highest WER of 0.811 observed for the object "Kala Bhairava." This suggests that the model struggles to handle this accent, particularly for phonetically complex or less commonly encountered terms. Conversely, the lowest WER of 0.230 is achieved for the object "Prayer Wheel" in the Australian accent, highlighting this combination as the easiest for the model to transcribe accurately.

6.7 Web Application Result

When a user interacts with the web application, they are first prompted to upload an image for object detection and question-answering. The interface is designed to provide a seamless experience, displaying the uploaded image on the user's side of the chat window for better context. Once the image is uploaded, the YOLO-based object detection model processes it to identify objects within the image. If objects are detected, they are highlighted and labeled directly on the image, making it easy for the user to see the results. After object detection is complete, the BART-based question-answering (QA) model engages with the user, enabling them to ask detailed questions about the detected objects. For example, if the image contains a prayer wheel, the user might ask, "What is a prayer wheel?" The system then analyzes the detected object and provides precise answers based on the context. This interactive session ensures users get meaningful insights from their images, and the intuitive interface ensures that all results, including detected objects and the original image, are displayed side by side.



2024-12-19 13:57:44

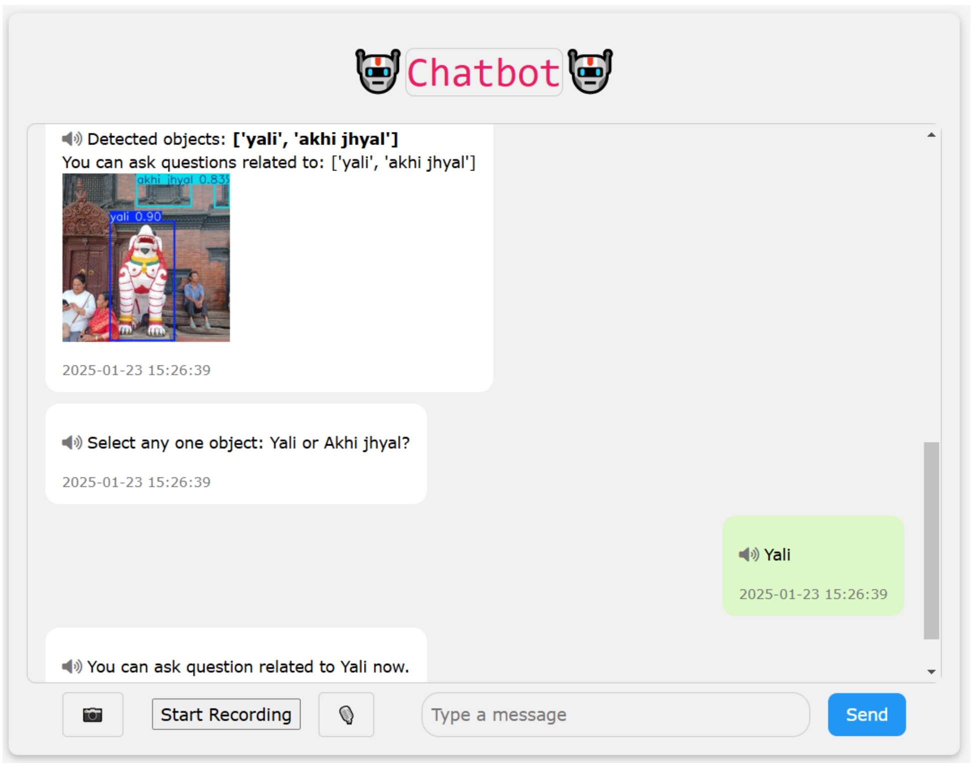
Detected objects: **akhi jhyal**
You can ask questions related to: akhi jhyal

2024-12-19 13:57:44

What is it?
2024-12-19 13:57:44

akhi jhyal is a traditional wooden window of newar architecture in nepal, often with intricate carvings.
2024-12-19 13:57:44

Figure 6-27: Single Object Detected Case (Success)


Chatbot

🔊 Detected objects: **['yali', 'akhi jhyal']**
You can ask questions related to: ['yali', 'akhi jhyal']

2025-01-23 15:26:39

🔊 Select any one object: Yali or Akhi jhyal?

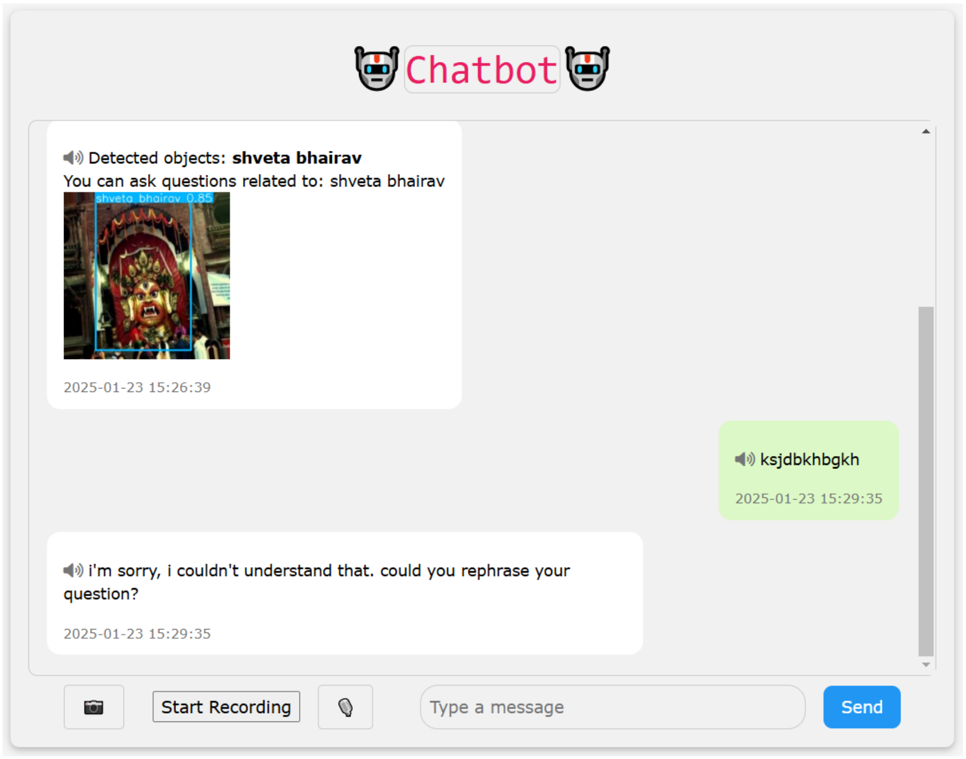
2025-01-23 15:26:39

🔊 Yali

2025-01-23 15:26:39

🔊 You can ask question related to Yali now.

Figure 6-28: Multiple Objects Detected Case


Chatbot

🔊 Detected objects: **shveta bhairav**
You can ask questions related to: shveta bhairav

2025-01-23 15:26:39

🔊 ksjdbkhbgkh

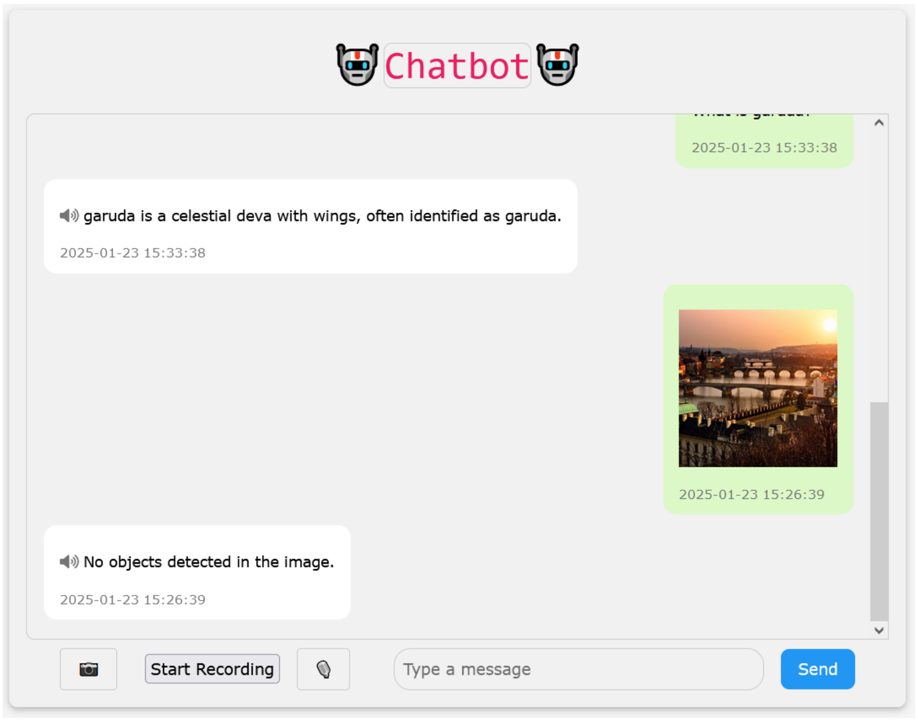
2025-01-23 15:29:35

🔊 i'm sorry, i couldn't understand that. could you rephrase your question?

2025-01-23 15:29:35

Figure 6-29: Web-Application Interface (Gibberish)

On the other hand, if a user attempts to ask a question before uploading any image, the application displays a warning message, reminding them to upload an image first. This functionality prevents unnecessary errors and ensures a logical workflow. Similarly, if the uploaded image contains no recognizable objects or includes objects outside the classes that the YOLO model was trained to detect, the system provides a clear warning message stating, “No objects detected in the image.” This message is accompanied by an appropriate placeholder image or graphic to visually communicate the issue. This approach avoids confusion and helps guide the user toward providing a valid image. By offering specific feedback in such cases, the application maintains a smooth and user-friendly experience while handling edge cases gracefully. Whether facilitating a meaningful Q&A session or ensuring proper image input, the system is built to provide a responsive and interactive platform for users.



2025-01-23 15:33:38

garuda is a celestial deva with wings, often identified as garuda.

2025-01-23 15:33:38

2025-01-23 15:26:39

No objects detected in the image.

2025-01-23 15:26:39

Figure 6-29: Web-Application Interface (Exception)

The application maintains the context of the detected object throughout a session, allowing users to ask multiple questions about it until a new image is uploaded. For instance, if a "prayer wheel" is detected, users can inquire about its features, and the system retains this context for consistent answers. When a new image is uploaded, the session resets, and the context switches to the objects in the new image. This ensures a seamless and organized experience, allowing users to interact with each image independently and effectively.

7. FUTURE ENHANCEMENTS

The mobile application leveraging AI for object detection and voice-based query processing has significant potential for future enhancements. Adding multilingual support would make the app accessible to users from diverse linguistic backgrounds by allowing voice queries and responses in multiple languages. Offline functionality, achieved by integrating lightweight models, would ensure uninterrupted service for tourists in areas with limited internet connectivity. Additionally, personalization features could enhance usability by adapting to user preferences, such as preferred languages, frequent queries, or objects of interest, and offering tailored recommendations.

Further improvements could focus on enhancing object detection by fine-tuning the YOLO model for diverse datasets and integrating other pre-trained models to identify more obscure or unique objects. Expanding compatibility with smart devices, such as wearables or AR glasses, could provide hands-free interaction and greater accessibility. These advancements would make the app more robust, versatile, and user-friendly, offering a powerful AI-driven solution for tourism and other applications.

8. CONCLUSION

This project showcases the potential of integrating advanced AI technologies, such as object detection and natural language processing, into a mobile application to assist tourists in exploring new places more effectively. By using models like YOLO for object detection, Whisper for voice-to-text conversion, and BART for generating meaningful responses, the application provides users with an intuitive way to interact with their surroundings. Its dual-input functionality, supporting both voice and image queries, offers flexibility and ensures accessibility for a wide range of users.

The system's design emphasizes user-centric functionality by allowing tourists to identify objects of interest and receive accurate and contextually relevant information through text or voice outputs. This eliminates language barriers and makes navigation and exploration easier, even in unfamiliar environments. The app's modular architecture ensures scalability, enabling future updates and enhancements to meet evolving user demands and technological advancements.

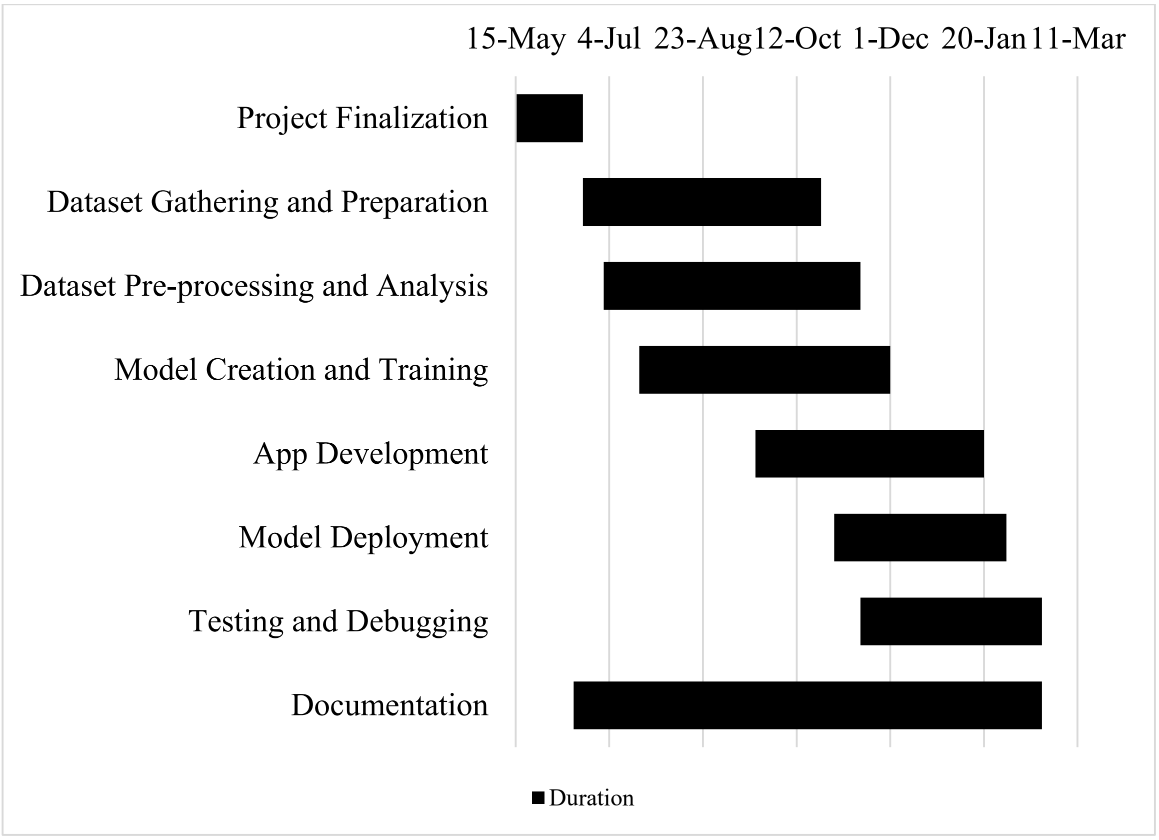
While the project achieves its core objectives of enhancing tourist experiences, it also highlights the importance of innovation in addressing real-world problems. The application not only provides valuable assistance to users but also demonstrates how AI can bridge gaps in communication, knowledge, and accessibility. This is particularly relevant in a globalized world where cultural and linguistic diversity often poses challenges to seamless exploration.

In conclusion, this project sets a foundation for developing intelligent and user-friendly systems that combine AI's power with practical applications. As the world moves toward greater digital transformation, solutions like this have the potential to revolutionize how people interact with their environment and access information. The project opens avenues for future research and innovation, paving the way for smarter, more accessible, and context-aware mobile applications.

9. APPENDICES

Appendix A: Project Timeline

Table 9-1: Project Timeline



Appendix B: Project Budget

Table 9-2: Project Budget

S.N.	Particulars	Quantity	Total (Rs)
1	Miscellaneous	-	10000/-
	Total		10000/-

References

[1] M. M. a. M. Fritz, "Towards a Visual Turing Challenge," 2015.

[2] S. A. e. al, "VQA: Visual Question Answering,," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Santiago, Chile, 2015.

[3] M. R. Mateusz Malinowski, "Ask Your Neurons: A Neural-based Approach to Answering Questions about Images," in *Conference: International conference on computer vision (ICCV)*, Santiago, 2015.

[4] R. K. a. R. Z. Mengye Ren, "Image Question Answering: A Visual Semantic Embedding Model and a New Dataset," in *Deep Learning Workshop at ICML 2015*, 2015.

[5] N. P. H. S. B. H. Hyeonwoo, "Image Question Answering Using Convolutional Neural Network with Dynamic Parameter Prediction," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[6] X. H. J. G. e. a. Z. Yang, "Stacked Attention Networks for Image Question Answering," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, 2016.

[7] X. H. Peter Anderson, "Bottom-Up and Top-Down Attention for Image Captioning and Visual Question Answering," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, 2018.

[8] D. Denis, "Using Deep Learning to Answer Visual Questions from Blind People," KTH Royal Institute of Technology, stockholm, 2019.

[9] A. B. P. A. A. P. Patil, "Speech Enabled Visual Question Answering using LSTM and CNN with Real Time Image Capturing for assisting the Visually Impaired," in *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, Kochi, 2019.

[10] Y. a. C. L. Xu, "Open-Ended Visual Question Answering by Multi-Modal Domain Adaptation," in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Association for Computational Linguistics, 2020, pp. 367--376.

[11] H.-T. N. Tung Le, "Multi Visual and Textual Embedding on Visual Question Answering for Blind People," *Neurocomputing 465,* 2021.

[12] H. H. Hu Yushi, "PromptCap: Prompt-Guided Image Captioning for VQA with GPT-3," in *International Conference on Computer Vision (ICCV)*, Paris, 2023.

[13] C. G. a. Y. Z. S. L. LV, "Multi-level Question Embedding Fusion for visual question answering," *Information Fusion,* vol. 102, p. 102000, 2024.

[14] A. S. Ruchita Sonawale, "Visual Mind: Visual Question Answering (VQA) with CLIP Model," *Ijraset Journal For Research in Applied Science and Engineering Technology,* 2024.

[15] B. S. a. I. K. Wonjae Kim, "ViLT: Vision-and-Language Transformer Without Convolution or Region Supervision," in *MACHINE LEARNING. INTERNATIONAL CONFERENCE*, 2021.

[16] Y. L. N. G. M. Lewis, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension," in *Annual Meeting of the Association for Computational Linguistics*, 2019.

[17] G. a. P. S. Lavanya, "Enhancing Real-time Object Detection with YOLO Algorithm," *EAI Endorsed Transactions on Internet of Things,* 2023.

[18] C.-Y. Wang, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," *Computer Science > Computer Vision and Pattern Recognition,* 2024.

[19] A. W. a. H. Chen, "YOLOv10: Real-Time End-to-End Object Detection," *Computer Science > Computer Vision and Pattern Recognition,* 2024.