TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Khwopa College Of Engineering
Libali, Bhaktapur
Department of Computer Engineering

A FINAL REPORT ON

AR Tour Companion: Enhancing Travel Experiences with Voice Assistance

Submitted in partial fulfillment of the requirements for the degree

BACHELOR OF COMPUTER ENGINEERING

Submitted by

Abhishek Hyangol	KCE076BCT003
Amit Shrestha	KCE076BCT004
Anuj Gaida	KCE076BCT007
Safal Raj Manandhar	KCE076BCT033

Under the Supervision of
Er. Milan Karki

Khwopa College Of Engineering
Libali, Bhaktapur
2023-24

Certificate of Approval

This is to certify that this major project work entitled **"AR Tour Companion: Enhancing Travel Experiences with Voice Assistance"** submitted by Abhishek Hyangol (KCE076BCT003), Amit Shrestha (KCE076BCT004), Anuj Gaida (KCE076BCT007) and Safal Raj Manandhar (KCE076BCT033) has been examined and accepted as the partial fulfillment of the requirements for the degree of Bachelor in Computer Engineering.

.....
Dr. Aman Shakya
External Examiner
Assistant Professor
Department of Electronics and Computer
IOE,Pulchowk

.....
Er. Milan Karki
Project Supervisor
Electronic and Computer Engineer
Build 360 Company

.....
Er. Dinesh Gothe
Head of Department
Department of Computer Engineering
Khwopa College of Engineering

Copyright

Acknowledgement

We would like to express our deep gratitude to our respected HOD of the computer department, Er. Dinesh Man Gothe sir for his advice, encouragement, and guidance for the completion of this project. We are also deeply indebted to Er. Niranjan Bekoju for his valuable suggestions and online training program. We would like to express our gratitude to our supervisor Er. Milan Karki for his supervision, encouragement, and advice during this project. Finally, we would like to acknowledge everyone who played a role directly or indirectly in completing this project.

Abhishek Hyangol	KCE076BCT003
Amit Shrestha	KCE076BCT004
Anuj Gaida	KCE076BCT007
Safal Raj Manandhar	KCE076BCT033

Abstract

This research introduces an innovative tour guide application that includes AR navigation and a Voice Assistant Chatbot integrated with a Monument Recognition Model that has the potential to revolutionize the tourism industry. AR Tour Companion with Voice Assistance is a practical application of Deep Learning (Computer Vision and Natural Language Processing) and Augmented Reality. It can be used to navigate to monuments, restaurants, hotels, and ATMs of Bhaktapur Durbar Square, detect and recognize monuments, and finally have an interactive and informative conversation about the recognized monument. In this paper, we have developed an Android application with an inferencing server that integrates a Monument Recognition Model which can detect a monument in the image using MobileNet-SSD v2 and recognize the detected monument using the CNNs-LSTM algorithm in real-time. The system includes pre-trained Whisper AI for speech-to-text conversion, a Transformer model as a chatbot for responding to users queries, and GTTS Library for text-to-speech conversion in real-time as well. Finally, the system also implements Mapbox Vision SDK, Navigation SDK, and Direction API for AR navigation. The working mechanism of our application involves four main stages, they are navigating to monuments through AR-rendered navigation signs on the live camera feed, detection and recognition of monuments, and audio-based query-response conversation of recognized monuments. The collected image datasets were annotated using Roboflow web app. The dataset for the NLP model was prepared from the collection of contexts about the monuments of Bhaktapur Durbar Square from published books, research papers, related articles, online blogs, and news portals.

Keywords: *Android App, Augmented Reality, CNNs-LSTM, Deep Learning, Mapbox, MobileNet-SSD v2, Natural Language Processing, Transformer, Whisper AI*

Contents

List of Tables

List of Figures

List of Symbols and Abbreviation

AIML	Artificial Intelligence Markup Language
API	Application Programming Interface
AR	Augmented Reality
BoW	Bag-of-Words
ATM	Automated Teller Machine
BERT	Bidirectional Encoder Representations from Transformers
BLEU	BiLingual Evaluation Understudy
BN	Batch Normalization
BoW	Bag-of-Words
CSV	comma-separated values
CV	Computer Vision
CDT	Context Dimension Tree
CNN	Convolutional Neural Network
ESB	Enterprise Service Bus
FAQ	Frequently Asked Questions
GRU	Gate Recurrent Unit
GPS	Global Positioning System
GTTS	Google Text-to-Speech
HOG	Histogram of Oriented Gradients
IMU	Inertial Measurement Unit
IOU	Intersection Over Union
JSON	JavaScript Object Notation
LDA	Latent Dirichlet Allocation
LSA	Latent Semantic Analysis
LSTM	Long Short-Term Memory Network
MOS	Mean Opinion Score
NLP	Natural Language Processing
NLTK	Natural Language Toolkit
PII	Personally Identifiable Information
SDK	Software Development Kit
SGD	Stochastic Gradient Descent
SSD	Single Shot Detector
VIO	Visual Inertial Odometer
VGG	Visual Geometry Group

Chapter 1

Introduction

1.1 Introduction

Augmented Reality technology and Natural Language Processing applications, Chatbot are in the spotlight of recent technological advancement. Using AR and NLP technology we have designed an AR Tour Companion with Voice Assistance to access a smartphone's GPS location and camera feed to navigate the user to the desired location by rendering AR content; AR Lane and AR Fence onto the live video feed of the device's camera. It can also detect and recognize the monument from images and allows for interactive query sessions about the recognized monument providing information regarding historical, architectural, religious, and cultural aspects of the recognized monument. In recent years, there have been various studies on monument and landmark recognition across the globe but none in the context of Nepal. Also, the AR Navigation system in the Google Maps application known as "Live View" was launched in August of 2019, but the feature is currently available only in six cities in different countries around the globe. Also, there are multiple LLM (Large Language Model) chatbot models under development as well as available for general-purpose use. However, their accuracy about a specific context is not always accurate. So this system is deployed as an android app with the features to provide AR navigation, monument detection, and chatbot with specific knowledge domain of monuments of Bhaktapur Durbar Square with the motive to promote tourism and attract attention to preserve cultural heritage. In this system, we have included four monuments of Bhaktapur Durbar Square for training our object detection and recognition models whereas we have included all the monuments within the core area of Bhaktapur for training our chatbot model. Since the system is deployed as a mobile application, anyone with a smartphone and internet connection has the privilege to use the system. Therefore, the AR Tour Companion with Voice Assistance system can be useful as a virtual tour guide to domestic as well as foreign tourists visiting Bhaktapur Durbar Square. Additional feature of navigating users to nearby local services like Newari cuisine, hotels, ATMs, etc makes the system a fully functional virtual guide.

1.2 Motivation

Nepal is a country with a flourishing tourism industry. But most of the country's people occasionally visit the cultural heritage sites and do not hire tourist guides. In the context of Nepal, applications related to virtual tour guides haven't been developed. This was one of the motivating factors to pursue this project. Besides famous monuments, people don't have much knowledge about other existing monuments.

1.3 Problem Statement

In the present context of Nepal's tourism, the tourists are provided with small booklets and brochures by the Tourism Authorities which contain very limited information about the heritages they have been visiting and since the booklets and brochures are limited in hard-copy format they get lost at some point. Similarly, the general public seems to have poor knowledge and declined interest in the monuments and heritages. So, to tackle this problem, there should be some means to make people accessible to historical information about these heritages and monuments easily and conveniently. Hence, we designed a system that renders AR navigation signs to guide tourists to heritage locations, detects and recognizes monuments as well as interacts with users about historical, religious, and cultural aspects of the heritage site. This system can be proven effective in a cultural heritage-rich country like Nepal by preserving the information about such heritages and monuments digitally and then making them easily accessible to the local people as well as the tourists. The system is deployed as a mobile application such that it is easily available and accessible to everyone.

1.4 Objective

The main objective of this project is to design a mobile application with an AR navigation component, Monument Recognition, and an audio Chatbot model that can function as a virtual tour guide.

1.5 Scope and Applications

The AR Tour Companion with Voice Assistance system can be used as a fully functional virtual tour guide for the Bhaktapur Durbar Square tourism site.

Chapter 2

Literature Review

AI-embedded conversational chatbots in the native language, industrial sectors like banking, emotion-based chatbots, etc are in the state of research in the present context of Nepal. However, no related research on the domain of the tourism industry on the heritage sites is to be found. In recent years, AR navigation systems have been developed on a commercial basis rather than a research-based approach with disclosed implementation methods.

2.1 AR Navigation

2.1.1 An ARCore-Based Augmented Reality Campus Navigation System

In this paper [2], a mobile application for indoor and outdoor navigation systems using ARCore to render 3D models in real scenes is developed with Unity platform. Visual inertial odometer(VIO) algorithm with an extended Kalman filter is used for real-time locating and map generating in mobile devices by integrating the inertial measurement unit(IMU) data with vision. Further, an Area learning module implemented with a boundary extraction method is used to extract and remember visual features of a region so that, the device can recognize the region. MOS is used as a performance metric to evaluate the AR navigation system developed.

2.1.2 Development of an Augmented Reality Tour Guide for a Cultural Heritage Site

This paper [3], describes design and development of AR based mobile application for tour guide of Hwaseong Fortress,South Korea as target site.The app stores pre-processed and binarized images of scanned targets. Scanning mode of the app takes continuous images at pre-defined rate, binarizes and compares with stored target images. Corresponding AR content designed using Unity and ARKit plugin is rendered on the scanned target once match is detected. The process of scan, recognition and visualization of rendering is repeated to deliver multiple AR objects.

2.1.3 Mobile Application Outdoor Navigation Using Location-Based Augmented Reality (AR)

The mobile application [4], is build using Unity AR GPS-location software plugin for deploying AR contents on desired location using latitude and longitude coordi-

nates. Furthermore, the app is exported to smart phones from the Unity platform as well.

2.1.4 Outdoor Navigation System by AR

The research paper [5] introduces a novel approach to outdoor navigation implementing Kalman filtering to estimate user position to improve the accuracy of AR objects displayed. The paper discusses the key components including GPS for location tracking, digital maps for spatial reference, and AR technology for rendering contextual information. The system is developed using Unity AR Foundation for rendering AR objects, Mapbox API services for route aquisition.

S.N	Title	Summary
1	An ARCore-Based Augmented Reality Campus Navigation System [2]	Navigation system built in Unity with ARCore, uses VIO algorithm and Area learning module for real-time locating and learning target region respectively.
2	Development of an Augmented Reality Tour Guide for a Cultural Heritage Site [3]	Mobile app that stores pre-processed target area images, compares the stored image with pre-processed camera-feed and renders AR content if match is found.
3	Mobile Application Outdoor Navigation Using Location-Based Augmented Reality (AR). [4]	Mobile app build using AR GPS-location software plugin in Unity platform.
4	Outdoor Navigation System by AR. [5]	Mobile app build using AR Foundation, Mapbox API in unity platform with implementation of Kalman filter to improve accuracy of rendered AR object.

Table 2.1: Review Matrix with Research Papers and summary of corresponding papers for AR Navigation.

2.2 Chatbot Model

2.2.1 A Chatbot for supporting users in Cultural Heritage contexts

In this paper [6], a prototype chatbot to help students study cultural heritage of archaeological park of Paestum is developed.The system's Inference Engine includes text analysis and context extraction.Context elements in text generated by chatbot are identified through Context Dimension Tree (CDT), Latent Dirichlet Allocation(LDA) model is implemented for this purpose. The system divides user interaction into clusters of simple sentences using appropriate Bayesian filters for keywords

2.2.2 Chatbot for University Related FAQs

Ranoliya, B.R., et al. at 2017 [7] purposed a University Related FAQs which uses the dataset of FAQs of the university. Here pre-processing and evalutation part are not explained. The model answers query based on datasets usig AIML and LSA(Latent Semantic Analysis). The unanswered questions by AIML are viewed as a reply by LSA. The operation consists of 3 steps, user post query on chatbot then processing is done on user query to match the predefined format, finally pattern matching between user entered query and knowledge is done.

2.2.3 A cultural heritage framework using a Deep Learning based Chatbot for supporting tourist journey

In this paper [8], web application of conversational agent based on seq2seq model using Multi-level Gate Recurrent Unit(GRU) cell is used for guiding tourists. Back-end design for supporting conversational interface consists of a micro-service architecture and an Enterprise Service Bus (ESB) has been designed for analyzing data from different sources. TensorFlow has been used to develop the chatbot engine and Negative Log-likelihood Loss function is used for evaluating its performance.

2.2.4 An application for Cultural Heritage using a Chatbot

In this paper [9], a Chatbot system is designed to provide contextual information through semantic analysis of an archeological park in Italy. System architecture contains various modules; Semantic Analyzer module to analyze and obtain topic from text using Latent Dirichlet Algorithm(LDA), Workflow Analyzer for intent recognition using appropriate Petri Network, Context Aware Module that contextualize answers based on information from Knowledge Base and External Services(websites,social media). The collection and pre-processing of datasets for chatbot are not mentioned.

2.2.5 Indonesian Chatbot of University Admission Using a Question Answering System Based on Sequence-to-Sequence Model

In this paper [10], a Chatbot system is designed using Sequence-to-Sequence model with and without attention mechanisms. Acoording to the comparative analysis, the model with attention mechanism and bidirectional LSTM encoder was somewhat better than the model without attention mechanism. BLEU was used to analyze the model.

S.N	Title	Summary
1	A Chatbot for supporting users in Cultural Heritage contexts [6]	Chatbot prototpye with text analysis and context extraction in its Inference Engine through use of LDA algorithm and divides user's text into clusters with Bayesian filters for keywords.
2	Chatbot for University Related FAQs [7]	Chatbot that answers queries from refernce of its knowledge base using AIML and LSA algorithms.
3	A cultural heritage framework using a Deep Learning based Chatbot for supporting tourist journey [8]	chatbot based on seq2seq model and Multi-level GRU with ESB for analyzing data from different sources.
4	An application for Cultural Heritage using a Chatbot [9]	chatbot system uses LDA to analyze contents in text, Petri Network for intent recognition and Context Aware module to contextualize respone from contents of Knowledge Base and External Services.
5	Indonesian Chatbot of University Admission Using a Question Answering System Based on Sequence-to-Sequence Model [10]	An application that has implemented Seq2Seq model with and without attention mechanisms. On reviewing output model with attention mechanism was slightly better.

Table 2.2: Review Matrix with Research Papers and summary of corresponding papers for Chatbot Model.

2.3 Monument Recognition Model

2.3.1 Landmark Recognition Using Machine Learning

In this paper [11], for detection of buildings in image, the image is cropped into multiple overlapping cells with identical aspect ratios and features are extracted from the cells using the HOG descriptor and classified using the SVM algorithm. The proposed model used dataset consisting of 193 images of various buildings collected from Google Images. To improve performance of model, each classifier was run 20 times varying the training and test sets by randomly permuting the dataset. The accuracy of the SVM classifier approaches 90%

2.3.2 Towards Deep Learning for Architecture: A Monument Recognition Mobile App

This project [12] uses data augmentation techniques on the dataset of roughly 50–100 images per monument to generate 500 training images per monument which is trained using CNN algorithm based on the MobileNet model implemented in Python using the open-source libraries TensorFlow and Keras. A commercial iOS app was developed with overall accuracy of the trained models estimated to be over 95%.

2.3.3 Monument Recognition using Deep Neural Networks

In this paper [13], the concept of Transfer learning has been used to prune the computational load, dataset of about 400 images per monument were used to retrain the final layer of the Inception model. The model is tested on a few arbitrary images and results with a training accuracy of 99.4% and corresponding testing accuracy ranging from 96-99% .

2.3.4 Cross Platform Web-based Smart Tourism Using Deep Monument Mining

The proposed platform [14] uses a deep neural network (VGGNet model; transfer learning technique is used) for feature extraction from the images captured on a mobile phone and a classification algorithm: SVM and Random Forest are used for identification of monuments in the image. image uploaded to the web server detects the monument in the image, extracts its information from the database, and sends the results to the device.

2.3.5 Mobile Travel Guide using Image Recognition and GPS/Geo Tagging A Smart Way to Travel

This mobile app [15] project uses Open-CV for image recognition and GPS navigation system with Google Maps API to point the location of that monument on the map. The information associated with monument is available in the JSON database. To identify the monument, the camera's captured image is compared to images stored in the database using Open-CV. The application compares two images using histogram comparison and feature detection using the ORB method.

S.N	Title	Summary
1	Landmark Recognition Using Machine Learning [11]	In this paper, feature extractor; HOG descriptor and classifier; SVM algorithm is used on 193 training images with model accuracy of 90%
2	Towards Deep Learning for Architecture: A Monument Recognition Mobile App [12]	This project uses CNN algorithm based on the MobileNet model trained on 500 images per monument to develop a commercial iOS app with overall accuracy of the trained models estimated to be over 95%.
3	Monument Recognition using Deep Neural Networks [13]	In this paper, Transfer learning has been used only retraining the final layer of the Inception model with 400 images per monument. The model is tested on a few arbitrary images and results with a training accuracy and testing accuracy of 99.4% and 96-99% respectively.
4	Cross Platform Web-based Smart Tourism Using Deep Monument Mining [14]	In this paper,feature extraction: (VGGNet model; transfer learning technique is used) and classification algorithm: SVM and Random Forest are used for identification of monuments in the image.
5	Mobile Travel Guide using Image Recognition and GPS/Geo Tagging A Smart Way to Travel [15]	This mobile app project uses Open-CV for image recognition and GPS navigation system with Google Maps API to point the location of that monument on the map. The application compares two images(camera's captured image and images stored in the database) using histogram comparison and feature detection using the ORB method.

Table 2.3: Review Matrix with Research Papers and summary of corresponding papers for Monument Recognition Model.

2.4 Theoritical Framework

2.4.1 Convolutional Neural Network

Convolutional Neural Network [16] also known as CNN or ConvNet is a class of artificial neural network which is mostly used for image analysis. It can detect patterns from images by using convolutional layers which helps in image analysis. A convolutional layer will accept some input and then perform some transformation which is a convolution operation using some filters and then output it to the next layer which may either be a convolutional layer or a non-colutional layer. The filters are generally small matrices that can be used for detecting patterns from the images which are also matrices and they can detect patterns like edges, corners, squares, etc. A digital image is the binary representation of data that contains a series of pixels arranged in a grid fashion that contains pixel values that will indicate the brightness and color of a pixel. In our project, we have used CNN-based models for the detection and classification of monuments.

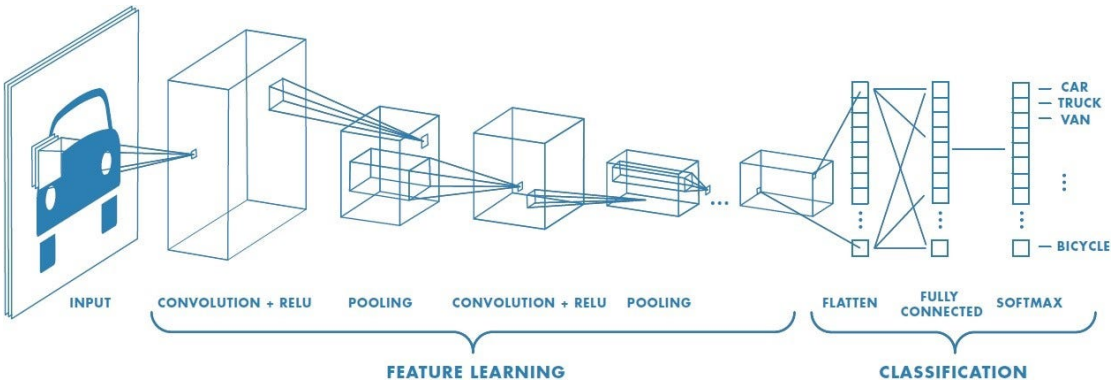


Figure 2.1: Architecture of CNN
(source: [16])

2.4.2 Transfer Learning

Transfer learning is one of the deep learning techniques that makes use of pre-trained models on large datasets trained for some specific task for the new, related task with a smaller dataset. By using transfer learning, a deep learning model can be trained more efficiently and with less data than training from scratch. This is because the pre-trained model has already learned general features that are useful for a wide range of tasks, and only needs to be fine-tuned on the specific dataset for the new task. Transfer learning is a good choice in situations when the amount and quality of the dataset are less and when training from scratch is time-consuming and expensive. By leveraging pre-trained models, transfer learning can significantly reduce the amount of data required for a specific task, while still achieving good performance. In our project, we have performed transfer learning on MobileNet-SSD v2, VGG-16 and VGG-19 models. Fine-tuning the pre-trained models on our monument image dataset.

2.4.3 Intersection Over Union

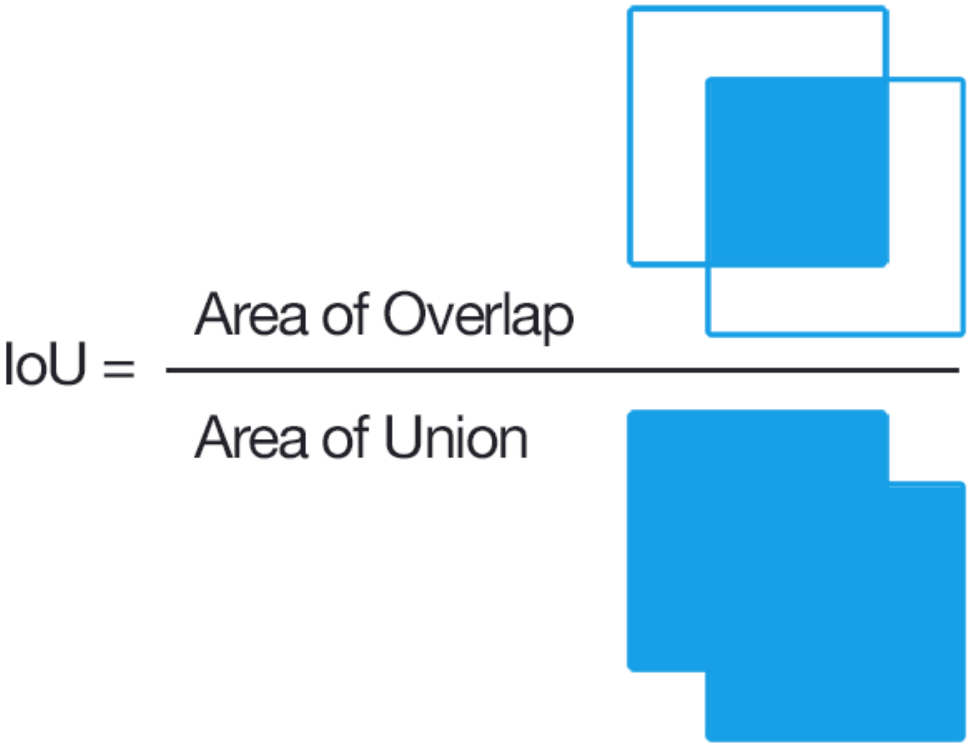


Figure 2.2: Intersection Over Union

Intersection over union(IOU) is a measure of overlap between two bounding boxes. In computer vision, it is used for correctly detecting an object. To know object detection first you have to know about object localization. Object localization refers to figuring out where is the object in the picture and showing it with a rectangular box. IOU is known to be a good metric for measuring overlap between two bounding boxes or masks. The following pictures show what is goodness measure of IOU

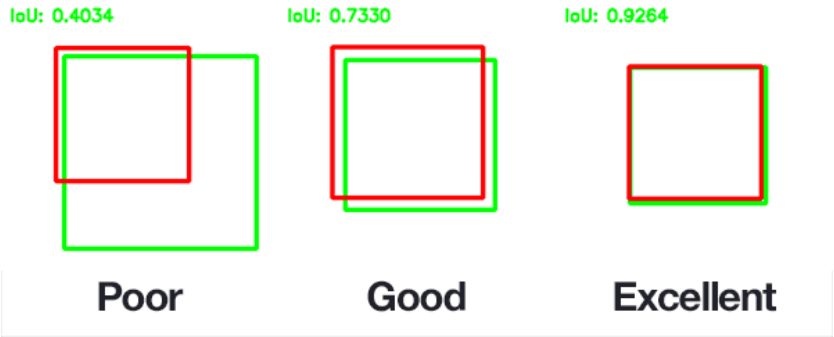


Figure 2.3: Goodness measure of IOU

Informally, IOU measures how equal two areas are. In terms of size and location of the area. If two areas are exactly equal, IOU will be 1. If two areas are far apart, even if their shape is the same, they will have IOU 0. And if two areas

lie at the same location but their size differs a lot, then also IOU will be a small value. [17]

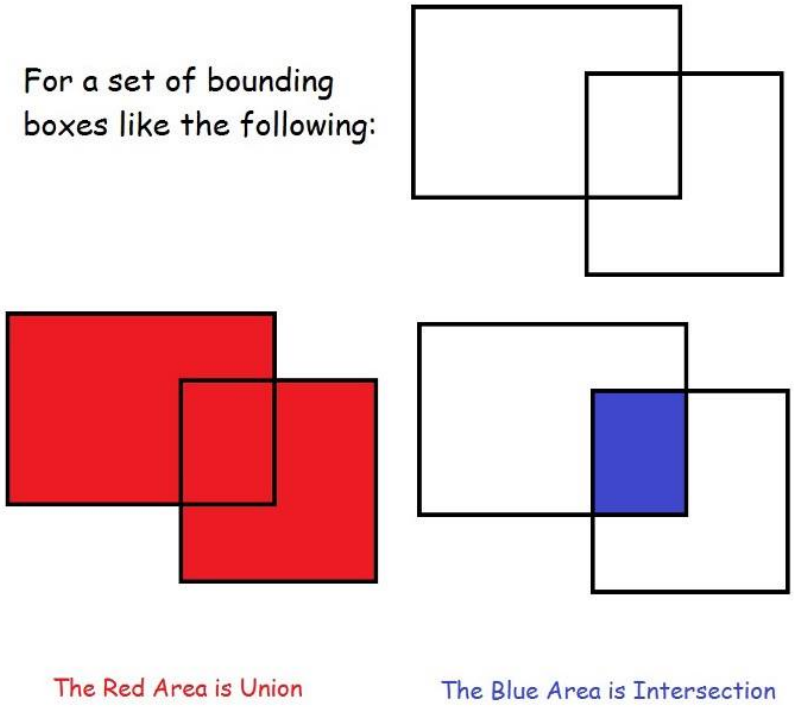


Figure 2.4: Bounding Box Union and Intersection

$$IOU(Box1, Box2) = \frac{Intersection_Size(Box1, Box2)}{Union_Size(Box1, Box2)} \tag{2.1}$$

2.4.4 Non-Maximum Suppression

Non-Maximum Suppression (NMS) is a technique used in many computer vision algorithms. It is a class of algorithms to select one entity (e.g. bounding boxes) out of many overlapping entities. The selection criteria can be chosen to arrive at particular results. Most commonly, the criteria are some form of probability number along with some form of overlap measure (e.g. IOU).

In object detection, there is a problem that an algorithm detects multiple bounding boxes for a single object. To solve this problem there is a technique called non-max suppression. Non-max suppression cleans up the multiple detections and ends with just one detection per object. For this it chooses the bounding box with the highest probability and suppresses all the other bounding boxes who's IOU with it is greater, so in last only one bounding box is left which is more accurate. [18]
Input: A list of Proposal boxes B, corresponding confidence scores S and overlap threshold N.
Output: A list of filtered proposals D.
Algorithm:

A. Select the proposal with the highest confidence score, remove it from B, and add it to the final proposal list D. (Initially D is empty).

B. Now compare this proposal with all the proposals — calculate the IOU (Intersection over Union) of this proposal with every other proposal. If the IOU is greater than the threshold N, remove that proposal from B.

C. Again take the proposal with the highest confidence from the remaining proposals in B and remove it from B and add it to D.

D. Once again calculate the IOU of this proposal with all the proposals in B and eliminate the boxes that have a higher IOU than the threshold.

E. This process is repeated until there are no more proposals left in B

Figure 2.5: Result before and after Non-max Suppression

2.4.5 Transformer

Transformers are neural network architecture introduced in the paper "Attention is all you need" [19] that revolutionized sequence-to-sequence modeling tasks that solved the limitations of recurrent neural networks by using attention-based mechanisms. The transformer architecture consists of an encoder and a decoder. The encoder processes the input sequence producing a fixed-length vector representation, which is then used by the decoder to generate the output sequence. The self-attention mechanism in the Transformer architecture allows the model to weigh the importance of different parts of the input sequence when making predictions. Specifically, the encoder and decoder use multi-head self-attention, which allows the model to jointly attend to information from different positions in the input sequence. Each attention head learns a different representation of the input sequence, which is then combined to produce the final output. The encoder is composed of a multi-head self-attention and a fully connected feed-forward network. The decoder is composed of masked-multi-head attention, attention with encoder output, and a feed-forward network. The self-attention mechanism allows the model to capture long-range dependencies and focus on important parts of the input sequence, while the residual connections and layer normalization help to stabilize training and improve performance. Transformer architecture has achieved state-of-the-art performance on a variety of natural language processing tasks. In our project, we have used a pre-trained transformer model Whisper AI for speech-to-text conversion. We have also built our custom transformer model trained on our dataset from scratch.

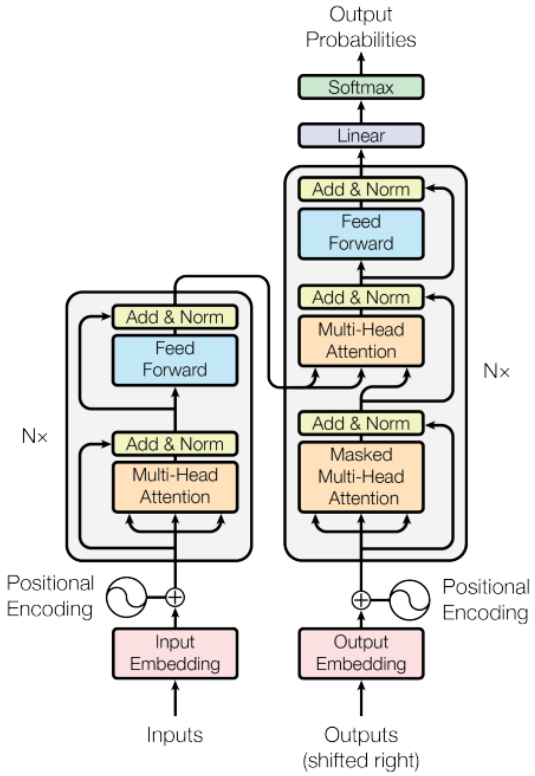


Figure 2.6: Architecture of Transformer (source: [19])

2.4.6 Augmented Reality

Augmented reality (AR) is an interactive technology that overlays graphical digital information over the physical world. Its main purpose is to combine real-world and virtual information in a manner that provides real-time interaction. AR can be defined as a system that incorporates three basic features: a combination of real and virtual worlds, real-time interaction, and accurate 3D registration of virtual and real objects. The overlaid sensory information can be constructive (i.e. additive to the natural environment), or destructive (i.e. masking of the natural environment). This experience is seamlessly interwoven with the physical world such that it is perceived as an immersive aspect of the real environment. In our project, we have used Mapbox Direction API, Vision, and Navigation SDK to provide an AR-based navigation system.

2.4.6.1 Mapbox Directions API

Mapbox Direction API provides routes and directions between two or more points on a map. It is frequently employed in applications that call for distance estimation, route planning, and navigation.

2.4.6.2 Mapbox Navigation SDK

The Mapbox Navigation SDK for Android embeds the power of Mapbox Directions API and provides the necessary tools to build a turn-by-turn navigation experience within an application. The SDK includes components to create a navigation app including accurate positioning on the map, displaying real-time progress along a route, and providing audio and visual guidance for maneuvers [20].

2.4.6.3 Mapbox Vision SDk

The Mapbox Vision SDK uses powerful neural networks to run on existing devices to understand the roadway in real-time. The Vision SDK is composed of the following frameworks:

a. Vision framework is the primary library to integrate Mapbox Vision. Its components enable camera configuration and display of interfaces.

b. Vision AR framework is an add-on module for Vision used to create augmented reality experiences. It uses Mapbox Directions API and Mapbox Navigation to allow configuration of the user's route visualization via rendering of AR signs (Lane and Fence).

Chapter 3

Requirement Analysis

3.1 SOFTWARE REQUIREMENT

Softwares tools required to prepare our system:

- Android Studio
- ClickUp
- FastAPI
- Figma
- GitHub
- Google Collaboratory
- Google Drive
- Kotlin
- Microsoft Azure
- Microsoft Teams
- OverLeaf
- Postman
- Python
- StarUML
- Visual Studio Code

3.2 HARDWARE REQUIREMENT

Hardware required to prepare our system are as follows:

- A computer system with the:
    - CPU: Intel Core I5 (7th generation or newer) or equivalent AMD Ryzen 5 (2000 series or newer)
    - Installed memory (RAM): 8 GB or above
    - Storage Device: SSD (500 GB or above)
    - GPU: CUDA enabled GPU (Graphics card by NVIDIA)

- An Android OS based mobile phone with the:
  - Octa core CPU or above
  - ROM: 2GB or above
  - RAM: 16GB or above
  - Camera: 8 Megapixels or above
  - Android Version 7.0 or higher
  - arm64-v8a or armeabi-v7a architecture with OpenCL support
- Google Colabs GPU '12 GB NVIDIA Tesla K80 GPU'
- 12GB Nvidia RTX 3080Ti

3.3 FUNCTIONAL REQUIREMENT

After completing this project, our system has achieved the following functional requirements, which describe the functionality of the system.

3.3.1 AR Navigation

The system can receive destination input from the user via a map interface or list of Landmarks and nearby services. The mobile app requests Mapbox Direction API for the navigation route to a selected destination and then renders AR Lane and AR Fence as navigation signs on the live stream of the device's built-in camera.

3.3.2 Monument Detection

The mobile app contains MobileNet-SSD v2 model which can detect monument in the device's camera feed in real time.

3.3.3 Monument Recognition

The system can recognize the detected monument using the CNNs-LSTM model deployed on the Cloud Server via POST request from the mobile application.

3.3.4 Interaction with Chatbot

The Chatbot model, also deployed on the Cloud server, receives speech input from users sent via POST request from a mobile application and responds to generated text as well as audio answers.

3.4 NON-FUNCTIONAL REQUIREMENT

These are essential for the better performance of the system. The points below focus on the non-functional requirement of the system prepared.

3.4.1 Reliability

Different test metrics are conducted to test the accuracy of the system.

3.4.2 Maintainability

The System is broken down into multiple sub-modules so that we can know where the problem is and maintain the sub-module.

3.4.3 Performance

The object detection requires only a single process and Mapbox SDK as well as API has optimized navigation. So, it will be able to provide better performance in real-time.

3.4.4 Usability

The system application is easy to use and simple to understand.

3.4.5 Availability

Users can detect and recognize monuments using an on-device detection model and send requests to an Azure virtual machine server which also contains a Chatbot model for interaction hosted all the time. so, it is available at any time the server is on.

Chapter 4

FEASIBILITY STUDY

4.1 Economic Feasibility

The main expenses for this project are the computer, cloud services, GPU, and computation power. As we used our computer for designing, training, and operating the model, the cost of the computer is preserved. Also, we used the free version of Google Colab and Kaggle for GPU resources and Google Drive for storage which is also free of cost. Our main expense was for hosting and deployment of the model in cloud services. We used Azure cloud services for hosting our models, which provided us with $100 credits for using educational mail, so this expense was managed. Finally, we used the Mapbox account's free tier, which provided 10 MAU (Monthly Active Users) for free. Considering all these factors, our project is economically feasible.

4.2 Technical Feasibility

Technically, this project is challenging as we worked on the monument dataset of Bhaktapur Durbar Square. However, the dataset needed for this project is free and easily available on different websites, published books, research papers, and articles related to Bhaktapur Durbar Square and we have used the monument image dataset collected manually by taking pictures from our mobile device camera our Minor Project Research. we developed the NLP models by using Python language. we also developed an API using the FastAPI framework, which was hosted in Azure cloud services and accessed by a mobile app developed using Android Studio. Models were trained in Google Colab GPU. Also, we had a ready-to-use Mobilenet-SSD v2 model trained on our custom Monument Image dataset and a ready-to-use CNNs-LSTM image classification model developed during our Minor Project Research. Finally, we have used GTTS Library and Wishper AI model for speech-to-text and text-to-speech conversion respectively. Considering all these facts, we can conclude that this project is technically feasible

4.3 Operational Feasibility

We have developed our system with interactive UI and UX, so our system is easy to operate. In this system, the user can select a destination directly on the map or explore the list of landmarks included in the UI. Users can input speech by speaking to which the system will return the response to the input query in audio and text format. We have also added various exceptional handling mechanisms to this project, considering various test cases. This system is operationally feasible.

4.4 Time Feasibility

The project was expected to be finished within 1 year time. A lot of time for collecting, preprocessing, and annotating the image dataset as well as training and building the computer vision model was saved by using the dataset and models from our Minor Project Research. Thus, we have completed all of our research and built all of the models on time. So, the project is time-feasible.

Chapter 5

Methodology

5.1 SOFTWARE DEVELOPMENT APPROACH

The Agile methodology is a method of managing a project by breaking it into several phases focusing on releasing products frequently and updating the project as per the requirement. The agile method cycle includes Planning, Requirement Analysis, Designing, Development, and Testing stages. Agile software development refers to a group of software development methodologies based on incremental and iterative approaches for evolution and adaptation welcoming changes at any time of the SDLC process. Scrum, Kanban, Extreme Programming, and Adaptive Project Framework are some of the development frameworks under the Agile project management methodology. For this project, we are going to use the Scrum framework as it encourages team collaboration and quality software development. Scrum projects are broken down into small and consistent time-boxed iterations of SDLC.

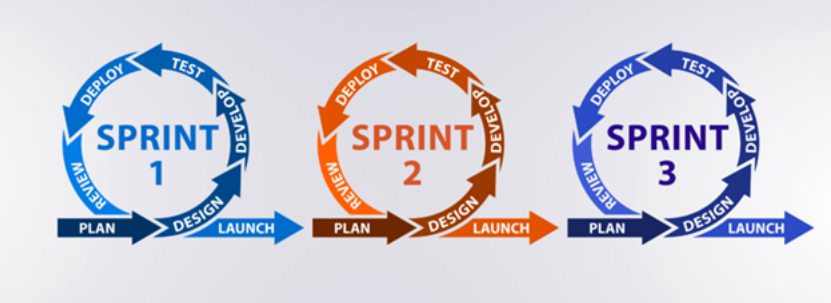


Figure 5.1: Agile Methodology, Scrum Model for Software Development [1]

5.2 ClickUp as Project Management Tool

We used ClickUp as our project Management tool to organize, distribute, manage, and track all the works and to assign tasks to each member of our team as shown in figure A.1. ClickUp is a project management software that allows users to manage tasks, projects, and workflows. It offers a variety of features that can help teams collaborate more effectively and increase productivity. First, we created a project plan then sprints and completed the tasks from the project plan in these sprints.

5.2.1 Product Backlog

Product Backlog consists of new features, bug fixes, and activities of changes during the development phase of the project. Currently, as we have completed four sprints, our product backlog includes the following features:

1. Studying research papers and blogs
2. Research and Experimentation of AR Navigation
3. Research on Word2vec Model
4. Research on Transformer Model
5. Dataset Collection for Chatbot Model
6. Dataset Preprocessing and preparation for BERT Model
7. Dataset preparation for pre-trained Transformer model
8. Training of BERT Model on custom Dataset
9. Training pre-trained Transformer model on custom Dataset
10. Training CNNs-LSTM Model on custom Dataset
11. Testing BERT Model
12. Testing pre-trained Transformer model
13. Evaluation of CNNs-LSTM Model
14. Custom Transformer Model Development
15. Dataset preparation for custom Transformer model
16. Training custom Transformer Model
17. Evaluation of custom Transformer model
18. UI design for Mobile App
19. Frontend Development using Android Studio
20. Deployment of MobileNet-SSD v2 in Mobile app
21. VM setup on Server
22. Backend Development using Fast API
23. Deployment of custom Transformer model on Server
24. Deployment of CNNs-LSTM Model on Server
25. Evaluation of AR Navigation
26. Documentation

Different tasks we have done in different sprints during project development are as follows:

5.2.2 Sprint 1

After doing some research and proper planning, we started our first sprint. The Sprint 1 consists of the following tasks:

1. Studying research papers and blogs for AR Navigation and Chatbot model
2. Dataset Collection for NLP Model
3. Dataset preprocessing and preparation for BERT Model
4. Research and Experimentation of AR Navigation
5. UI design for mobile app
6. Frontend development using Android Studio
7. Training BERT Model
8. Testing BERT Model

At the end of the first sprint, the required data was collected from various sources, analyzed, and processed. We prepared a dataset suitable for training the BERT Model. We trained and evaluated the BERT Model on our custom Dataset and also developed a tested basic prototype of an Android application implementing the Mapbox SDK and API for AR Navigation.

5.2.3 Sprint 2

We started the second sprint by analyzing the result of the BERT Model and AR Navigation Component from sprint 1, while exploring solutions we researched and experimented with the pre-trained question answer-based Transformer model. We prepared a dataset suitable for training the Model.We trained our CNNs-LSTM Model on the monument image dataset and evaluated the model. The sprint 2 consists of the following tasks from our Product Backlog:

1. Research on Transformer model
2. Dataset preprocessing and preparation for pre-trained Transformer model
3. Training pre-trained Transformer model
4. Testing pre-trained Transformer model
5. Training of CNNs-LSTM model
6. Evaluation of CNNs-LSTM model
7. Update UI design for mobile app

At the end of this second sprint, we tested a pre-trained Transformer model that can take input text as a query and respond to the text-based response, the CNNs-LSTM model that can classify monuments in the input image. We also updated the UI design of our prototype app for an easy-to-use interface and fixed some bugs.

5.2.4 Sprint 3

In the third sprint, we researched on Word2vec model for building chatbot and we deployed the monument detection and recognition models and developed the working framework of the system on the server. In the third sprint following tasks were completed:

1. Research on Word2vec
2. Backend Development using Fast API
3. VM setup on Server
4. Backend Development using Fast API
5. Deployment of CNNs-LSTM Model on Server
6. Deployment of MobileNet-SSD v2 Model in Mobile app
7. Frontend Development using Android Studio

At the end of the third sprint, the CNNs-LSTM model was deployed on the server after comparing its performance against CNNs-LSTM. Finally, the MobileNet-SSD v2 Model was deployed in the mobile app with a UI interface for the Chatbot Model.

5.2.5 Sprint 4

In the fourth sprint, we researched and developed a custom Transformer model, collected additional datasets, processed and prepared a dataset for training our custom Transformer model. We deployed a custom Transformer model on a server with the Wishper AI model and GTTS Library to complete the structure of the chatbot model.

1. Custom Transformer model development
2. Additional Dataset Collection for Custom Transformer Model
3. Dataset preparation for custom Transformer model
4. Training custom Transformer Model
5. Evaluation of custom Transformer Model
6. Deployment of custom Transformer model on the server
7. Evaluation of AR Navigation
8. Documentation

At the end of the fourth sprint, the final system with all the features was obtained. The chatbot system was deployed in the cloud. The documentation and report of the project was also completed.

Chapter 6
System Design and Architecture

6.1 Use Case Diagram

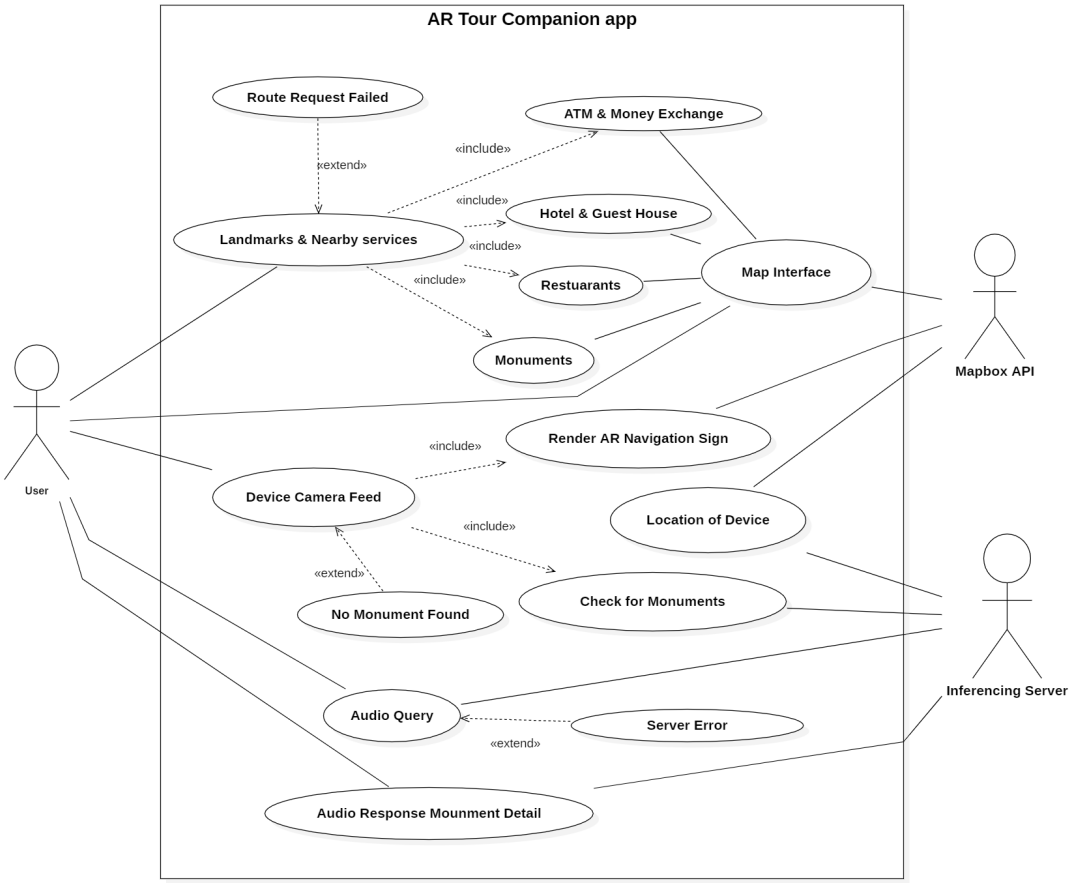


Figure 6.1: System Use Case Diagram

6.2 System Flowchart Diagram

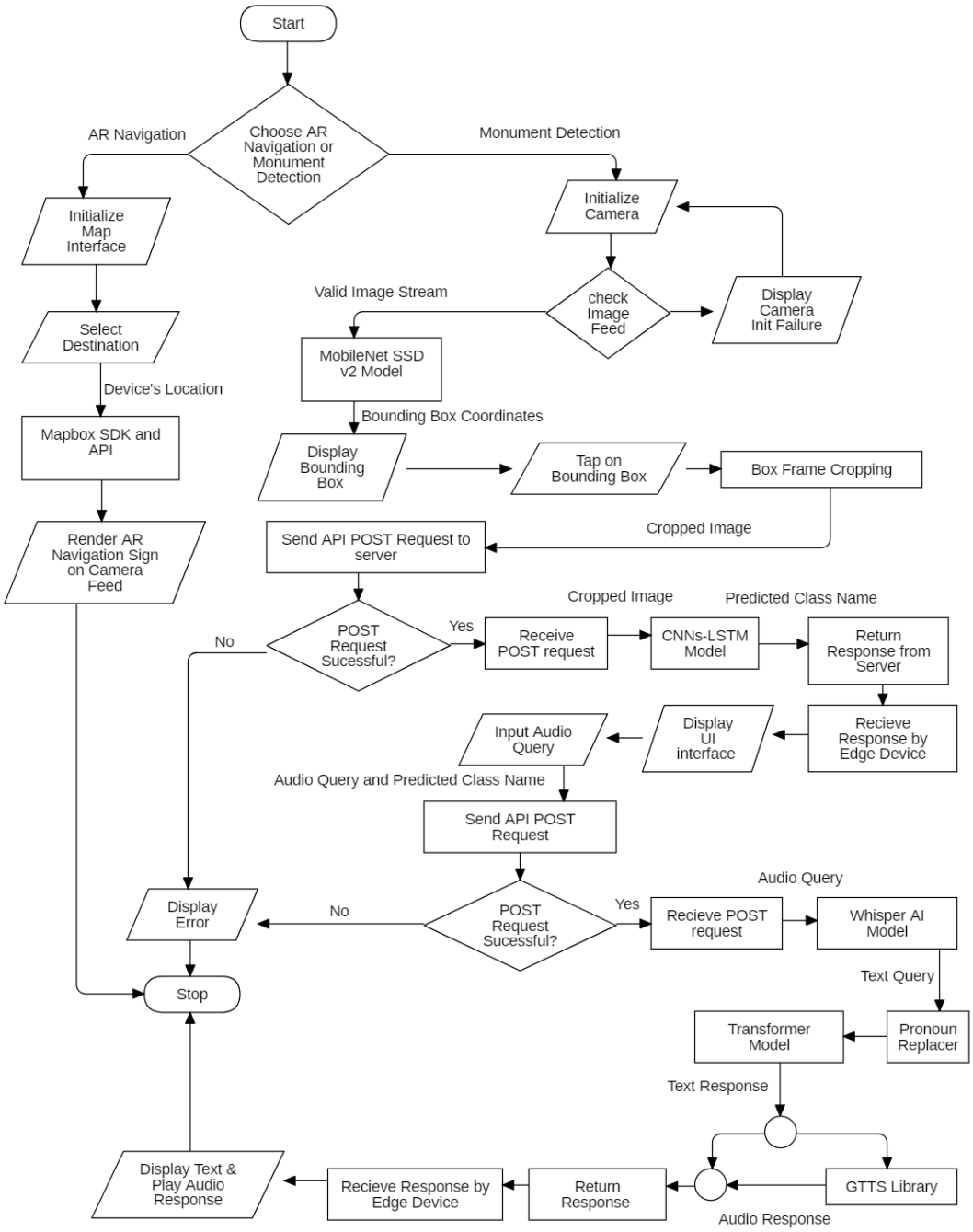


Figure 6.2: System Flowchart Diagram

6.3 Sequence Diagram

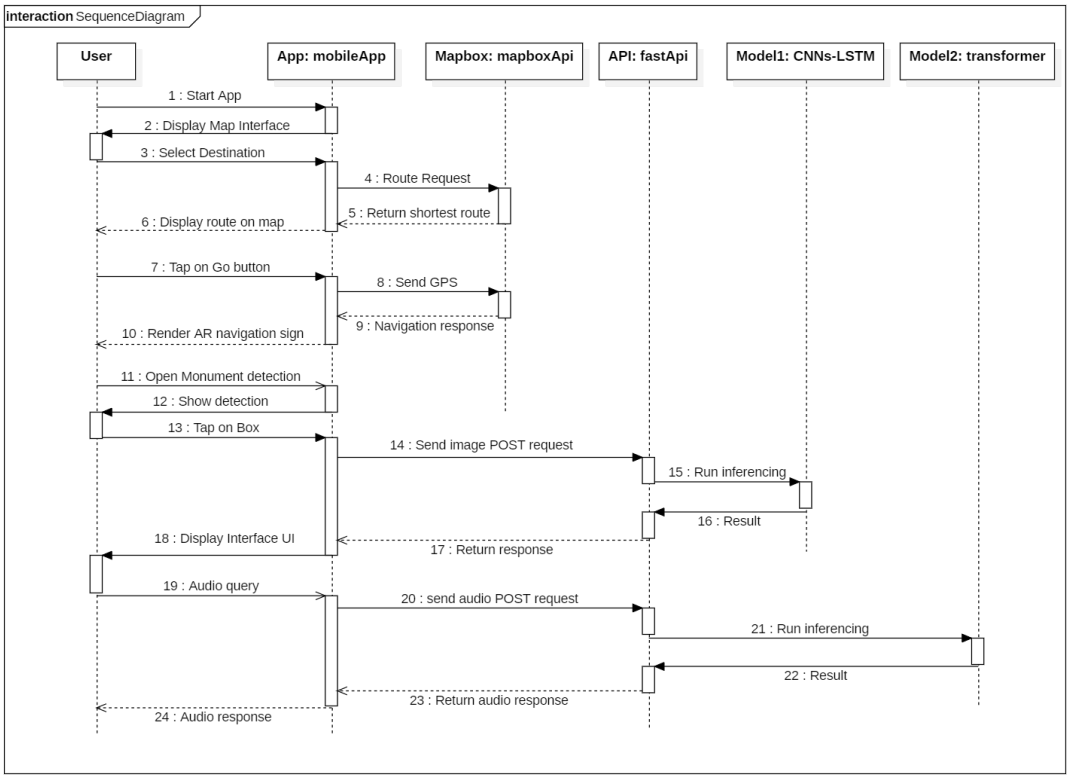


Figure 6.3: Sequence Diagram of the System

6.4 System Block Diagram

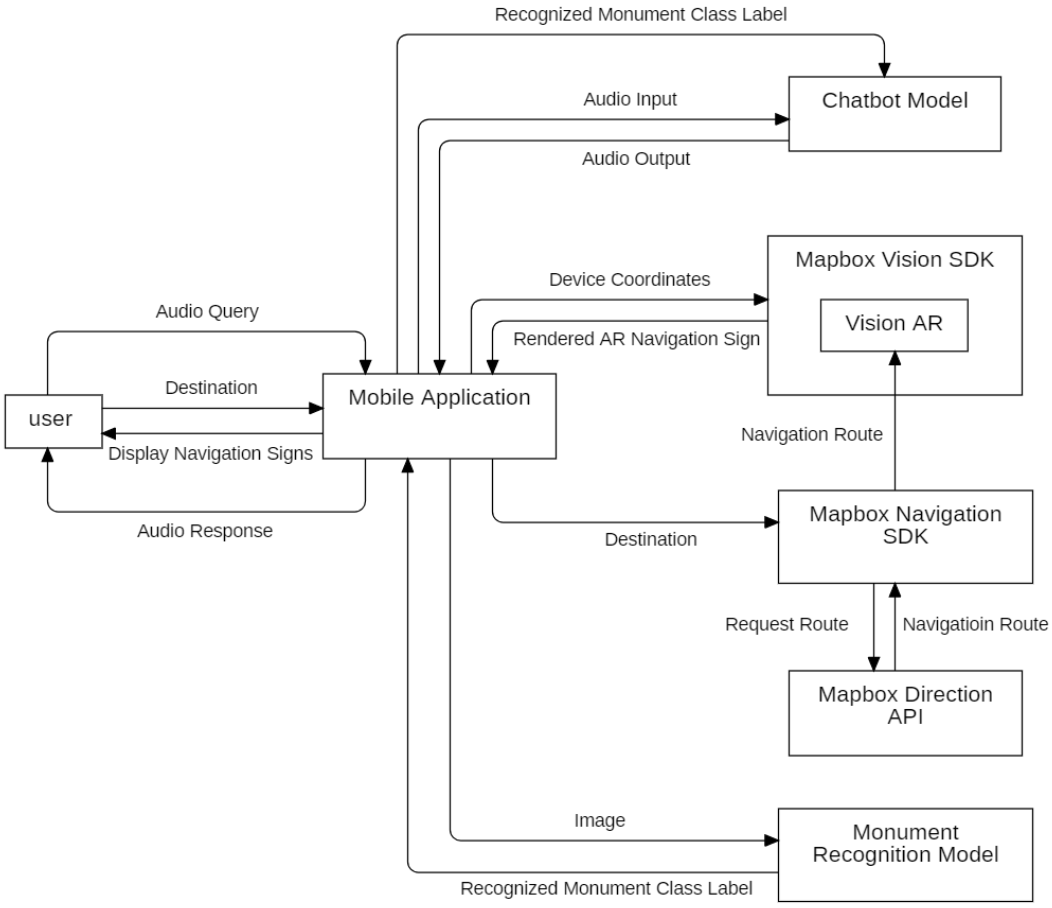


Figure 6.4: System Block Diagram

6.4.1 Description of System Block Diagram

The mobile application opens the map interface through Mapbox SDKs implemented in the Android application. After the user selects a destination on the map interface, the Mapbox Navigation SDK creates a new request to Mapbox Direction API whose response will be Navigation Route. Vision AR then manages the navigation route, translates it to the core library, and renders an AR navigation route on top of the live video stream from the device's built-in camera.

Once the user reaches the destination, the user can proceed with the Monument Recognition Model to recognize monuments in the image of the live video stream from the device's camera. After the monument is recognized UI interface of the Chatbot Model is enabled, the user can proceed with the audio query and get audio audio-based response from the Chatbot model using the device's microphone and speaker respectively.

6.5 Block Diagram for Monument Recognition Model

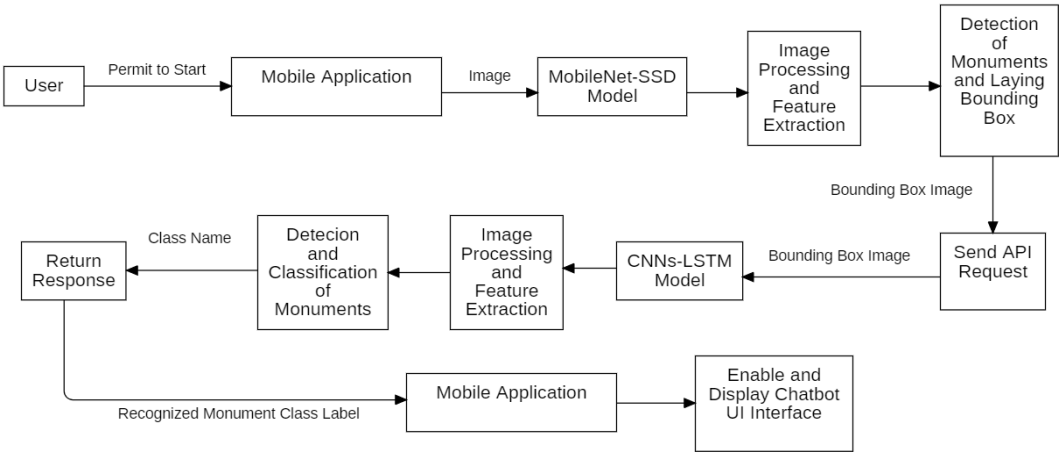


Figure 6.5: Block Diagram of Monument Recognition Model

6.5.1 Description

The camera is initialized through an Android application. MobileNet-SSD analyzes the image from the live camera feed for the presence of monuments and, if any monument is found, generates the bounding box coordinates. After the bounding box is formed, using those coordinates, the image within the box part is cropped on tapping. The cropped image is submitted to the server via an API POST request. The CNNs-LSTM model deployed on the server takes cropped images that have been received as input and returns the predicted label. The server responds with a class name, which the mobile application uses while sending API POST requests to the Chatbot Model.

6.6 Block Diagram for Chatbot Model

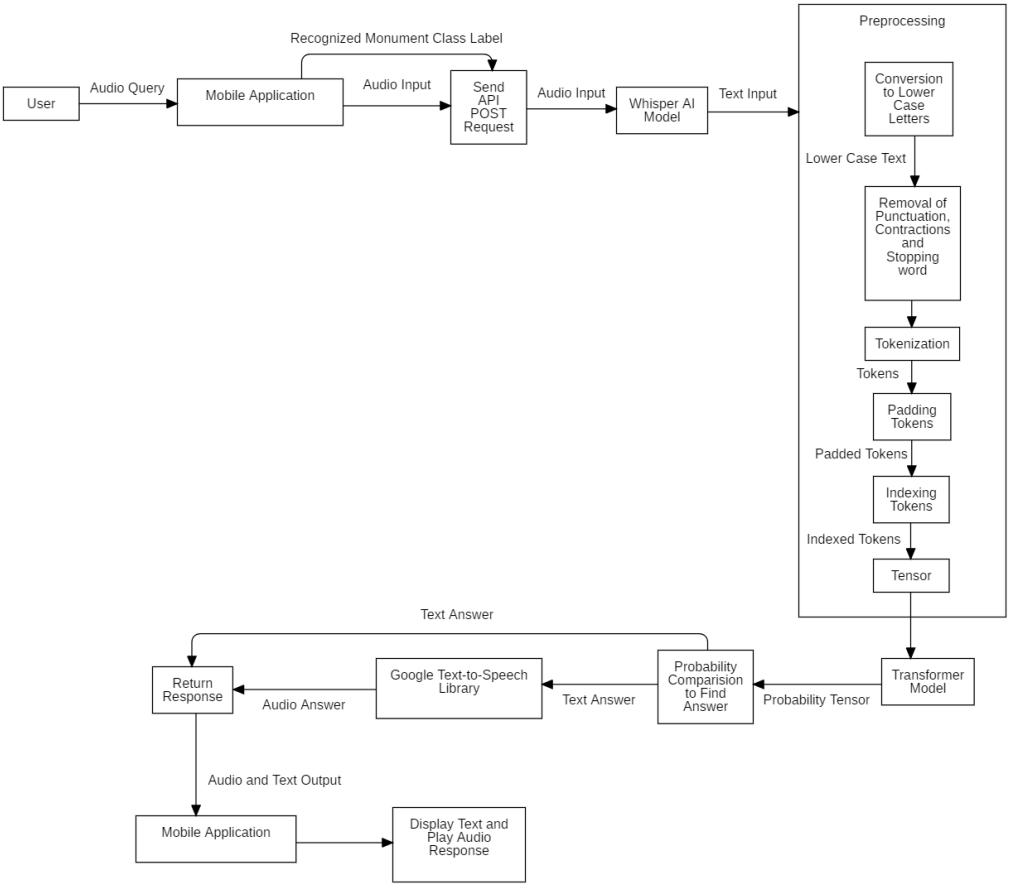


Figure 6.6: Block Diagram of Chatbot Model

6.6.1 Description

The mobile application sends the monument class name and audio query via API post request to the server. Since our Transformer model, stored on the server, requires text input, we used the Whisper AI model for speech-to-text conversion which is stored on the same server. Preprocessing is performed to convert the text input into a suitable tensor. As shown in the above figure, the input text is converted into lowercase then punctuation, stopping words, and contractions are removed from the text so that tokenization will be easier. Then the query is tokenized and converted into a suitable tensor. The Transformer model receives the tensor and processes it, returning the probability tensor that contains our answer. The probability tensor is compared with the tokens to find out the actual answer. GTTS library of Python is used for text-to-speech conversion. Finally, the server responds with audio and text output to the mobile application.

6.7 Models Description

6.7.1 Transformer Model

A transformer model is a neural network that learns context and thus meaning by tracking relationships in sequential data like the words in this sentence. Transformer, proposed in the paper [19], is a neural network architecture solely based on the self-attention mechanism and is very parallelizable. First, Transformers made it possible to process entire sequences in parallel, making it possible to scale the speed and capacity of sequential deep-learning models to unprecedented rates. Second, they introduced "attention mechanisms" that made it possible to track the relations between words across very long text sequences in both forward and reverse directions.

Transformers could be parallelized with great efficiency. Three key ideas summarize the innovation of Transformers:

1. Self-Attention - In order to construct a representation of a sequence, self-attention is an attention mechanism that links various points inside a single sequence. In simpler terms, self-attention helps us create similar connections within the same sentence. Self-attention helps neural networks disambiguate words, do part-of-speech tagging, entity resolution, learning semantic roles, and a lot more.

2. Positional Encodings - Positional encoding is added to Transformer since it lacks repetition and convolution, providing the model with additional context regarding the relative positions of the words in the sentence. The positional encoding vector is added to the embedding vector. Embeddings represent a token in a d-dimensional space where tokens with similar meanings will be closer to each other. However, the embeddings do not encode the relative position of words in a sentence. So after adding the positional encoding, words will be closer to each other based on the similarity of their meaning and their position in the sentence, in the d-dimensional space.

3. Attention - The ability to concentrate on one thing while disregarding other things that don't appear important at the moment is referred to as attention in general. In machine learning, this concept is applied by teaching the model to focus on certain parts of the input data and disregard others to better solve the task at hand. In tasks like machine translation, for example, the input data is a sequence of some text. When we humans read a piece of text, it seems natural to attend to some parts more than others. Usually, it's the who, when, and where part of a sentence that captures our attention.

   The modules that make up the encoder and decoder can be stacked on top of one another repeatedly. The inputs and outputs (target sentences) are first embedded into an n-dimensional space since we cannot use strings directly. It consists of two main components i.e.Encoder and Decoder.

Transformer Parameters	
Layer (type:depth-idx)	Param #
Encoder: 1-1	
Embedding: 2-1	1,789,952
L-Embedding: 2-2	25,600
Modulelist: 2-3	
TransformerBlock: 3-1	789,760
TransformerBlock: 3-2	789,760
TransformerBlock: 3-3	789,760
TransformerBlock: 3-4	789,760
Decoder: 12	
Embedding: 2-5	1,789,952
Embedding: 2.6	25,600
ModuleList: 2-7	
DecoderBlock: 3-5	1,053,448
DecoderBlock: 3-6	1,053,440
DecoderBlock: 3-7	1,053,440
-Linear: 2-8	
DecoderBlock: 3-8	1,053,440
	1,796,944
Dropout: 2-9	
Total params:	12,800,848
Trainable params:	12,800,848
Non-trainable params:	0

Table 6.1: Summary of Transformer model built on PyTorch.

1. Encoder –
   (a) Receives input sequences (questions) and processes them.
   (b) Utilizes self-attention mechanisms and feed-forward neural networks across multiple layers.
   (c) Captures and encodes the meaning and context of input sequences into a fixed-length vector representation.
   (d) Creates a rich contextual understanding of the input, allowing the model to comprehend the nuances and relationships within the input data.

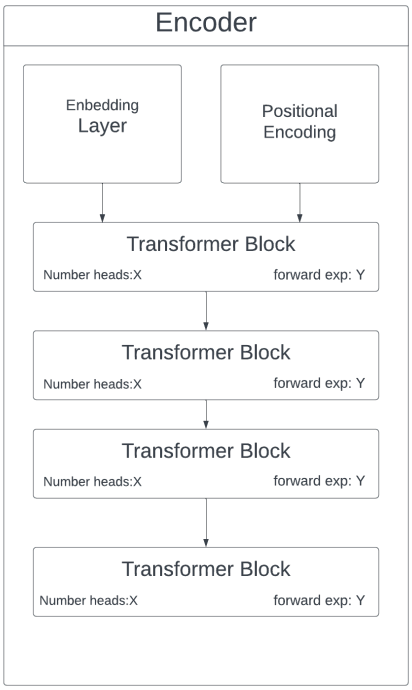


Figure 6.7: Encoder block diagram used in our Project.

2. Decoder -

   (a) Takes the encoded representation generated by the encoder.

   (b) Also consists of multiple layers with self-attention mechanisms and feed-forward neural networks.

   (c) During training, it predicts each token of the output sequence (answers) conditioned on the encoded input representation and the previously generated tokens.

   (d) Utilizes the contextual information from the encoder to generate accurate and contextually relevant output sequences.

   (e) Outputs a sequence token by token, attending to relevant parts of the input representation at each step, ensuring coherence and relevance in the generated output.

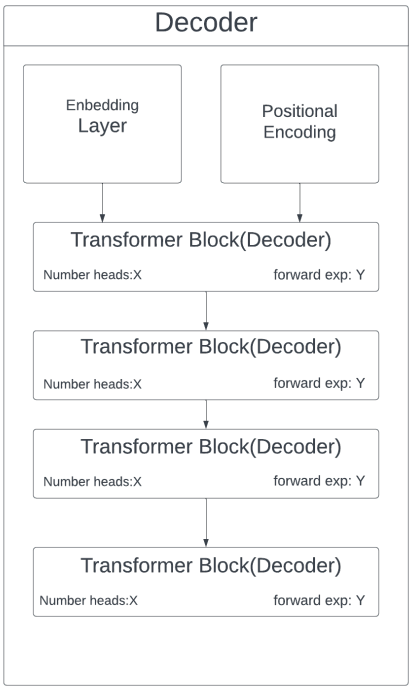


Figure 6.8: Decoder block diagram used in our Project.

6.7.2 Transformer Training

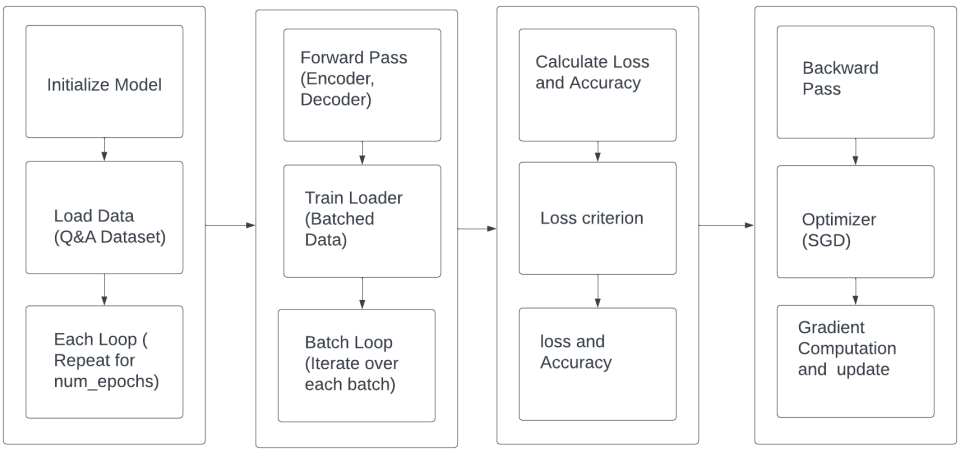


Figure 6.9: Block Diagram of Training.

6.7.2.1 Loading the dataset and preparing for training

1. **Splitting the Dataset** Divided the dataset into training and validation sets using the train_test_split function from the sklearn.model_selection module.

2. **Tokenization** Defined a tokenizer function using the get_tokenizer function from torchtext.data.utils . This tokenizer breaks down each sentence into a sequence of tokens (words).

3. **Building Vocabulary** Created a vocabulary from the training dataset using the build_vocab_from_iterator function from torchtext. vocab. This vocabulary maps each token to a unique index and also handles out-of-vocabulary (OOV) tokens.

4. **Defining Dataset Class** Defined a custom dataset class QADataset that inherits from torch.utils.data.Dataset. This class takes the DataFrame and tokenizer as inputs and tokenizes each question-answer pair in the dataset.

5. **Padding Sequences** Within the QADataset class, we padded or truncated the token sequences to a maximum sequence length using torch.nn.utils.rnn.pad_sequence. This ensures that all sequences have the same length for efficient batch processing.

6. **Creating DataLoader:** Created DataLoader objects for both the training and validation datasets using torch.utils.data.DataLoader. These DataLoader objects batch the data and provide iterators over the dataset during training.

6.7.2.2 Defining the Loss function

a) Loss Calculation

1. **Flattening the Tensors** The output tensor from the model and the target tensor are flattened to facilitate comparison. Flattening is done to convert the multi-dimensional tensors into 2D tensors.

2. **Ignoring Padding Tokens** Padding tokens in the target sequence are ignored during loss calculation. This is important because padding tokens do not contribute to the actual prediction task and including them in the loss computation may skew the results.

3. **Calculating CrossEntropyLoss** The CrossEntropyLoss function is applied to compare the output of the model with the target labels. This loss function computes the negative log-likelihood of the predicted probabilities compared to the actual target labels.

b) Accuracy Calculation

1. **Computing Predicted Labels** Predicted labels are obtained by selecting the token with the highest probability from the output tensor of the model. This is done using the argmax operation along the vocabulary dimension.

2. **Comparing Predicted and Target Labels** Predicted labels are compared with the actual target labels to determine if they match. This comparison is done element-wise.

3. **Calculating Accuracy** Accuracy is calculated as the ratio of correctly predicted tokens to the total number of non-padding tokens in the target sequence. It provides an indication of how well the model is performing in generating the correct outputs. Mathematically, accuracy A is defined as:

$$A = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Non-Padding Tokens}} \times 100\% \tag{6.1}$$

By following these steps, the loss and accuracy metrics are computed during the training process, providing valuable feedback on the model's performance and guiding its optimization.

6.7.2.3 Optimizer used

Optimizers play a crucial role in the training process of neural networks since they support in modifying the network's parameters to reduce the loss function and enhance performance. For the Transformer model, we used an SGD optimizer. SGD is a widely used optimization algorithm for training neural networks. It updates the model parameters in the direction opposite to the gradient of the loss function with respect to the parameters.
The basic update rule for SGD without momentum for a parameter θ at the time step t is:

$$\theta_{t+1} = \theta_t - \eta \nabla J(\theta_t)$$

SGD with momentum update rules:

$$v_{t+1} = \beta v_t + (1 - \beta) \nabla J(\theta_t)$$

$$\theta_{t+1} = \theta_t - \eta v_{t+1}$$

Where:

θ_t : value of the parameter at time step t
η : learning rate
$\nabla J(\theta_t)$: gradient of the loss function J with respect to θ_t
v_t : velocity at time step t
β : momentum coefficient, typically set between 0 and 1

Though we haven't performed randomized and grid searches, we manually tested for different cases with different values of momentum. Among them, momentum at 0.95 gave the best result with the following hyperparameters.

Parameter	Value
Embedding Size	256
Number of Layers	4
Forward Expansion	4
Number of Heads	4
Dropout Rate	0.05
Learning Rate	0.01
Number of Epochs	50

Table 6.2: Transformer Model Parameters

6.7.3 MobileNet-SSD v2

MobileNet-SSD v2 is a deep neural network for object detection that combines a MobileNet base architecture with a Single Shot MultiBox Detector (SSD) head. MobileNet-SSD v2 uses MobileNet base: a lightweight convolutional neural network based on depthwise separable convolutions, and SSD head: part of the network responsible for making object detection predictions.

6.7.3.1 Feature of MobileNet-SSD v2

Some of the features of MobileNet-SSD v2 are given below:

- It has a Depthwise separable convolutional layer which reduces the size and computational cost of the model.
- Loss function used is the combination of Localization loss and Confidence loss.
- It takes 320x320 size of the input image.
- Average pooling and Max pooling are used in MobileNet v2 and SSD architecture respectively.
- It can process video frames at an average of 19 frames per second.

6.7.3.2 MobileNet-SSD v2 Architecture

MobilneNet-SSD v2 shown in figure 6.10 consists of MobileNet v2 as base architecture and SSD head for detection. Mobilenet v2 consists of a standard inverse residual module, each inverse residual module contains a 1x1 convolutional layer, a 3x3 depth-wise separable convolutional layer, BN and Relu6. It provides a 38x38 output feature map which is fused by FPN to improve the performance of the detection network.

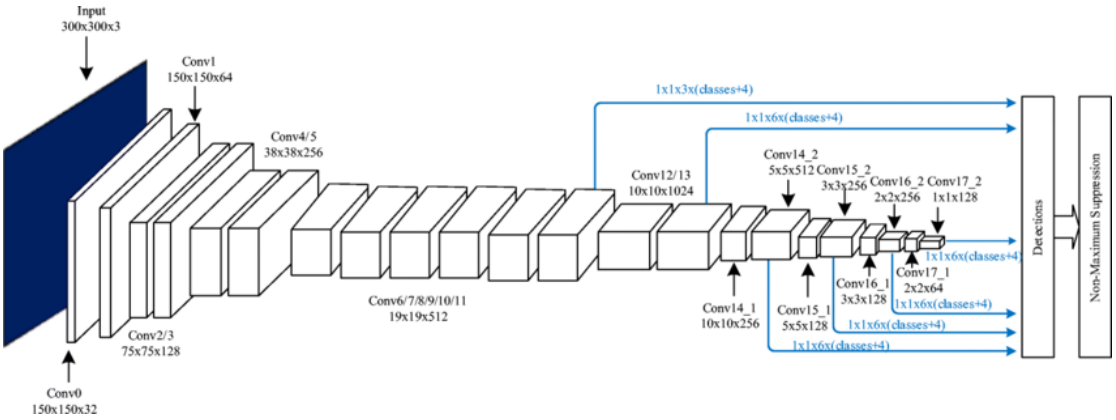


Figure 6.10: MobileNet-SSD v2 Architecture
(source: [21])

6.7.3.3 Activation Function

MobileNet v2 uses ReLU6 and SSD uses the ReLU activation function respectively. ReLU is a commonly used activation function in deep learning, defined as:

$$ReLU(x) = \begin{cases} x & \text{if} x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

ReLU6 is a variant of the standard ReLU function, which ensures that the output is always between 0 and 6, instead of being unbounded.

6.7.3.4 Loss Design

The MobileNet-SSD v2 object detection model uses a custom loss function that is designed to jointly optimize the localization and classification tasks in object detection. The mathematical expression [22] of the loss function used in MobileNet-SSD v2 can be written as follows:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{6.3}$$

where,

N represents the number of matched default boxes
x is the input image
c is the predicted class scores
l represents the predicted bounding box
g represents ground truth bounding box
α is a parameter that balances L_{conf} and L_{loc}
$L_{loc}(x, l, g)$ is the Localization loss defined as:

$$L_{loc}(x, l, g) = \sum_{i \in positive}^{N} \sum_{m \in \{cx,cy,w,h\}} {x_{ij}}^k smooth_{L1}(l_i^m - {(g^{'})_j}^m) \tag{6.4}$$

where,

Positive is the set of matched default boxes
m is the index of the bounding box coordinate
cx is the x-coordinate of the center of the bounding box
cy for the y-coordinate of the center of the bounding box
h for the height of the bounding box
w for the width of the bounding box
$smooth_{L1}(x)$ is the smooth L1 loss function, which is defined as:

$$smooth_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{6.5}$$

$${(g^{'})_j}^m = \begin{cases} ({g_j}^{cx} - {d_i}^{cx})/{d_i}^w, m = cx, \\ ({g_j}^{cx} - {d_i}^{cy})/{d_i}^h, m = cy, \\ \log({g_j}^w/{d_i}^w), m = w, \\ \log({g_j}^h/{d_i}^h), m = h. \end{cases} \tag{6.6}$$

$L_{conf}(x, c)$ is the Confidence loss defined as:

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} {x_{ij}}^{P} log(\hat{c_i}^{P}) - \sum_{i \in Neg} log(\hat{c_i}^{0}) \quad (6.7)$$

where,

c is the true class label for the object in the anchor box

$\hat{c_i}^{P}$ is the predicted objectness score for anchor box

$$\hat{c_i}^{P} = \frac{exp({c_i}^{P})}{\sum_{P} exp({c_i}^{P})} \quad (6.8)$$

Localization loss penalizes the difference between the predicted bounding box coordinates and the true bounding box coordinates for positive examples in training data. The confidence loss penalizes the difference between the predicted confidence score and the true confidence score for each positive and negative example in the training data.

6.7.4 CNNs-LSTM

6.7.4.1 VGG-16 Model

We used a pre-trained VGG-16 model from the Keras library. Fine-tuning was performed on the imported model by removing all dense layers then the convolutional layers were freezed except for the last convolutional layer, then two fully connected layers were added between with a batch normalization layer and a dropout layer before training the model. The architecture of the pre-trained VGG-16 and the summary of the constructed model are shown below.

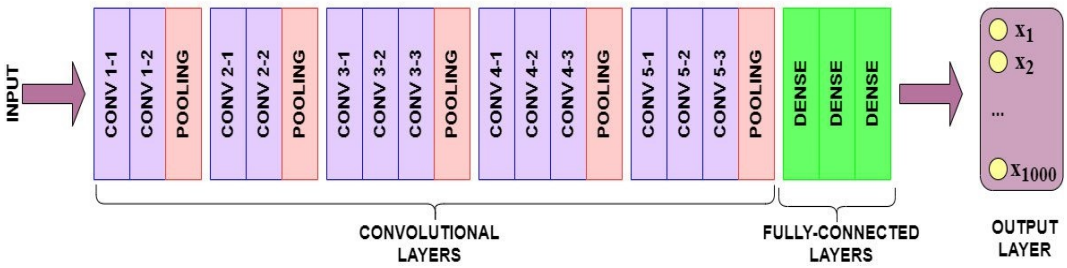


Figure 6.11: Architecture of VGG-16.
(source: [23])

Model: VGG-16 Model		
Layer	**Output Shape**	**Param #**
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 1024)	33555456
batch_normalization (BatchNormalization)	(None, 1024)	4096
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 4)	4100
Total params: 48,414,020		
Trainable params: 28,057,092		
Non-trainable params: 12,356,928		

Table 6.3: Fine Tuned VGG-16 Model Summary

6.7.4.2 VGG-19 Model

As with the VGG-16 model, we also used a pre-trained VGG-19 layer from the Keras library. All other actions were performed as in the VGG-16 model. The architecture of the pre-trained VGG-19 and the summary of the constructed model are shown below.

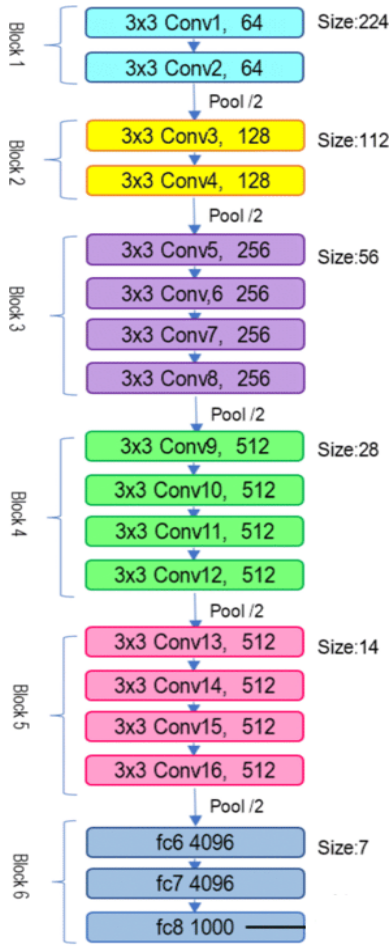


Figure 6.12: Architecture of VGG-19.
(source: [24])

Model: VGG-19 Model		
Layer	**Output Shape**	**Param #**
input_1 (InputLayer)	[(None, 224, 224, 3)]	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv4 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv4 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv4 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
dense (Dense)	(None, 1024)	33555456
batch_normalization (BatchNormalization)	(None, 1024)	4096
dropout (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 4)	4100
Total params: 45,723,716		
Trainable params: 28,057,092		
Non-trainable params: 17,666,624		

Table 6.4: Fine Tuned VGG-19 Model Summary

6.7.4.3 LSTM Model

Long short-term memory (LSTM) is an artificial neural network used in the fields of artificial intelligence and deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. Such a recurrent neural network (RNN) can process not only single data points (such as images) but also entire sequences of data (such as speech or video).
A common LSTM unit is composed of a cell, an input gate, an output gate, and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing, and making predictions. An LSTM network is the same as a standard RNN, except those summation units in the hidden layer are replaced by memory blocks. The multiplicative gates allow LSTM memory cells to store and access information over long periods of time [25].

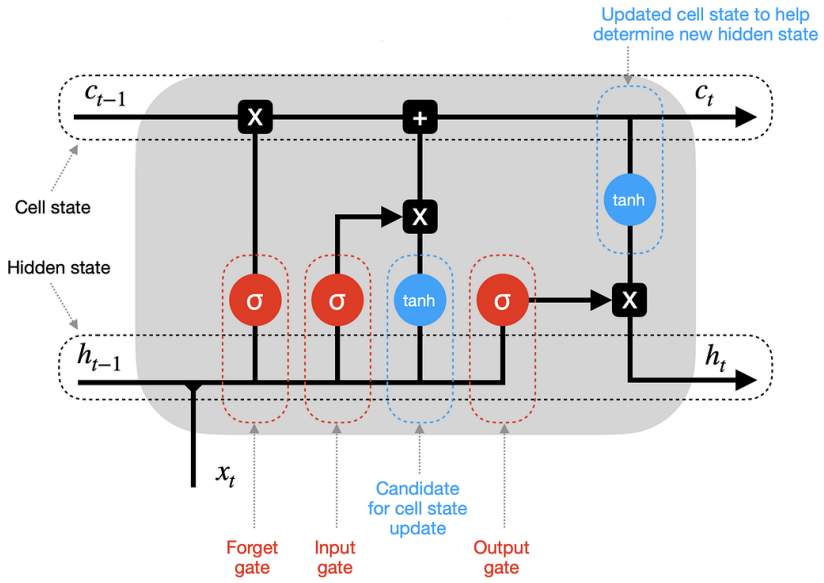


Figure 6.13: Architecture of LSTM.
(source: [26])

6.7.4.4 CNNs-LSTM Model

The proposed model is a combination of CNN (VGG model) and RNN (LSTM model). The model is composed of two CNN models, a single-layered LSTM model, and fully connected layers. The output of the CNN model is used to produce sequential data to help LSTM for making predictions so that it can easily extract highly accurate image features.

Model: VGG16-VGG19-LSTM Model			
Layer	**Output Shape**	**Param #**	**Connected to**
input_1 (InputLayer)	(None, 224, 224, 3)	0	
model (Functional)	(None, 7, 7, 512)	14,714,688	input_1[0][0]
model_1 (Functional)	(None, 7, 7, 512)	20,024,384	input_1[0][0]
flatten (Flatten)	(None, 25088)	0	model[0][0]
flatten_1 (Flatten)	(None, 25088)	0	model_1[0][0]
concatenate (Concatenate)	(None, 50176)	0	flatten[0][0], flatten_1[0][0]
lambda (Lambda)	(1, 224, 224)	0	concatenate[0][0]
lstm (LSTM)	(1, 224)	402,304	lambda[0][0]
dense (Dense)	(1, 4)	260	dropout[0][0]
Total params: 35,006,340			
Trainable params: 267,268			
Non-trainable params: 34,739,072			

Table 6.5: Integrated Model Summary

It consists of a single input layer taking RGB image of size 224x224 as input. Two models, model(VGG16) and model_1(VGG19) are connected to the same input layer. A flattened layer is added to both CNN models which are then concatenated together. The lambda layer is used to reshape the concatenated feature to the desired input shape for LSTM. Here the LSTM output(1, 224) represents the LSTM model that is trained a single image at a time and has 224 memory cells. Input to LSTM is provided in such a way that in each time step comparison of 224 features is performed and stored in a single memory cell. Finally, the LSTM layer is connected to a fully connected dense layer with four nodes for final prediction.

6.8 Dataset

6.8.1 Dataset Collection for Chatbot

In the Data collection stage, we collected information about various monuments around Bhaktapur Durbar Square. We scrolled through different websites for scrapping data as well as visited the monuments and collected data related to the monuments from local people. The following categories reflect the data that were gathered:

- Cultural aspects
- Historical events
- Religious aspects
- Architectural aspects
- Earthquake events
- Folklore stories

We collected information about monuments from different sources and compiled them. The sources from where data were collected are listed below:

- Books.
- Local Experts.
- Online Research (News Portals, Blogs, Articles, Travel Guides).
- Official Sources (Page of Bhaktapur Municipality).

We collected data in every way possible and tried our best to collect all the relevant information about the monument.

6.8.2 Dataset Collection for Monument Recognition Model

We needed to train two models using datasets. The two models include monument detection and monument recognition. We collected monument image dataset for model training using our mobile device's camera and collected around 2343 images of monuments which contained the images of 9 different monuments from Bhaktapur Durbar Square. We gathered images of monuments at different times of the day; morning, afternoon, evening, and night time to get different lighting conditions. We also gathered images of monuments from every angle in our reach. Finally, we also collected some negative images to improve the accuracy of our detection model.

6.9 Data Preprocessing

6.9.1 Data Preprocessing and Preparation for NLP MODEL

Once all the information obtained from different sources was combined, we were able to obtain plain text data regarding monuments. Question-answer formatted data was required to train the model. So, the following steps were followed for preprocessing data.

1. Anonymization.
2. Removing URLs.
3. Question-Answer Generation.
4. Data Structuring.
5. Data Splitting.
6. Text Cleaning.
7. Tokenization.
8. Building Vocabulary.

6.9.1.1 Anonymization

Anonymization refers to the process of removing or altering personal or sensitive information in a dataset or document to protect the privacy and confidentiality of individuals. Here's how anonymization was applied:

- Sensitive information such as personally identifiable information (PII) or sensitive data was identified and anonymized.
- Anonymization techniques like generalization and masking were used for anonymizing data.

6.9.1.2 Removing URLs

Removing URLs involves eliminating specific elements from text data. URLs are web addresses that direct users to specific web pages or resources. We removed the URL from the text.

6.9.1.3 Question-Answer Generation

After following the above steps we got a clean plain file that could be used to generate Question-Answer. Four team members divided the context to generate the questions and answers. We manually generated the questions and answers based on the context. We also used different online tools like GPT, Text summarization and question generation, and Question generation APIs like Quillionz API, etc for generating datasets. We generated around 5335 sets of unique question answers.

6.9.1.4 Data Structuring

After generation of data, it needs to be organized in a structured format. We used CSV and JSON file formats to organize the generated data.

6.9.1.5 Data Splitting

In order to prevent question-answer sets related to the same monuments from piling up, the preprocessed question-answer dataset was now shuffled. Then it was split into training and validation sets to evaluate the performance of the question-answering model. From the total dataset, 20% i.e. 3234 sets were used as Validation sets, and 80% i.e. 12939 sets were used as Training sets.

6.9.1.6 Text Cleaning

Removing any irrelevant or noisy information, such as special characters, or non-textual elements is termed Text cleaning. Sentence wise text cleaning was performed. All the sentences were converted to lowercase. The regular expression module was used to remove the noisy information. The following actions were performed using Regular Expressions:

- Special and Punctuation characters were separated from the word by whitespace.
- Special and Punctuation characters were replaced with whitespace.
- Contractions (like "I'm") were replaced by their full form (like "I am").
- NLTK library of Python was used for lemmatization which helped to reduce words to their base root form.
- NLTK library of Python was used for the removal of stop words like ("and", "is").

6.9.1.7 Tokenization

Tokenization is the process of breaking down a text document or a sequence of characters into smaller units. We implemented get_tokenizer from torchtext library to obtain basic_english tokenizer. Due to text cleaning, there were no special symbols to tokenize so words were tokenized only based on whitespaces.

6.9.1.8 Building Vocabulary

Compiling a list of every distinct term that appears in a set of text data is the process of building a vocabulary. The words are represented numerically by this list, which can then be used to feed a machine-learning model. For this model, we have set vocabulary size equal to the number of unique tokens generated from the dataset.

6.9.2 Data Preprocessing for Monument Recognition Model

From the total dataset, we manually selected around 200 clear images of 4 specific monuments such as Bhairavnath, Nyatapola, Vatsala Devi, and Gopinath Temple, and annotated those images for image segmentation and object detection using the Roboflow web app as shown in figure A.2.

Annotated Dataset		
Monuments	Training	Testing
Bhairavnath	179	43
Gopinath	168	44
Nyatapola	179	43
Vastsala Devi	179	43

Table 6.6: Annotated Data for Monument Detection and Recognition

The labeled dataset data was split into 80% train and 20% test data and was resized to 320x320 for training the MobileNet-SSD v2 model and 224x224 for training the CNNs-LSTM model using the features of the Roboflow web app.

6.10 Data Augmentation

6.10.1 Data Augmentation for NLP Model

Data Augmentation in NLP means generating new data from existing data in a way that preserves the original meaning while introducing variations. For Data Augmentation we performed the following things:

1. We replaced words in the text with their synonyms while maintaining semantic similarity.
2. We translated the text into Nepali and Hindi languages and then translated it back to the English language using Google Translate as shown in figure A.3.
3. We used different online paraphrasing tools to paraphrase data.
4. We performed random insertion and random deletion in some of the data preserving its actual meaning.

We were able to increase our dataset's 5335 unique dataset by 10838 after using all of these augmentation strategies, generating a total dataset of 16173 sets of data.

6.10.2 Data Augmentation for Monument Recognition Model

Data augmentation technique was applied to training data to expand our dataset, augmentation techniques used are:

1. Brightness
   The brightness of the annotated dataset was randomly altered between the range of 15%(.i.e brightness was either increased or decreased between 0 to 15% range) to generate 3 additional image data from an original image of the dataset as shown in figure A.4.1.
2. Rotation
   Similarly, the annotated dataset was randomly rotated between the range of 20° clockwise and 20° anti-clockwise to generate 3 additional image data from an original image of the dataset as shown in figure A.4.2.
3. 90° Rotate
   It includes 90° clockwise and anti-clockwise rotation on annotated training dataset as shown in figure A.4.3.

6.11 Mobile App Development

6.11.1 Frontend Development

The front end was designed using Android Studio and Koltin language. The app integrates Mapbox Vision, Navigation SDK, and MobileNet-SSD v2. Error handling is done. The front end contains basically three scenarios of the UX. Those three scenarios are given below.

1. AR Navigation
2. Monument Detection
3. Audio-based interaction UI for Chatbot Model

The Mapbox Vision SDK renders AR objects for navigation and MobieNet-SSD v2 detects monuments on the live feed of the device's camera. The detected monument image and speech query are sent to the inferencing server via POST Request using the Retrofit library.

6.11.2 Backend Development

In the backend, requests can be sent by end users to the server from the mobile app and then receive responses from the server using API which is built using FastAPI. Chatbot model (i.e. pre-trained Whisper AI, custom Transformer model, GTTS Library) and CNNs-LSTM model are launched on the cloud server. The conversion of the audio file to text and its preprocessing is performed in the backend Python script. The server takes audio files in the format MP3. The backend contains python scripts to run corresponding models deployed on the server, and they are:

1. Chatbot Script
2. CNNs-LSTM Script

These two scripts are run through the end user request from UI. The output is returned as a response in JSON format, which may be the generated text and audio or the predicted monument class name.

6.12 Model Evaluation

Evaluation for output of Detection is done using F1 Score analysis. It is determined from the confusion matrix as:

		Predicted	
		Negative	**Positive**
Actual	**Negative**	True Negative	False Positive
	Positive	False Negative	True Positive

Figure 6.14: Confusion Matrix

True Positive (TP): It refers to the number of predictions where the classifier correctly predicts the positive class as positive.

True Negative (TN): It refers to the number of predictions where the classifier correctly predicts the negative class as negative.

False Positive (FP): It refers to the number of predictions where the classifier incorrectly predicts the negative class as positive.

False Negative (FN): It refers to the number of predictions where the classifier incorrectly predicts the positive class as negative.

6.12.1 Precision:

Precision is a good measure to determine when the cost of a False Positive is high. Precision talks about how precise/accurate a model is out of those predicted positives, and how many of them are actually positive. The denominator is the Total Predicted Positive.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{Total\ Predicated\ Positive} \tag{6.9}$$

6.12.2 Recall:

Recall calculates how many of the Actual Positives our model capture by labeling it as Positive (True Positive).

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{Total\ Actual\ Positive} \tag{6.10}$$

6.12.3 Accuracy:

It measures the proportion of correctly classified samples among all the samples in the test set, meaning the fraction of the total samples that were correctly classified

by the classifier. The formula for Accuracy considers the sum of True Positive and True Negative elements at the numerator and the sum of all the entries of the confusion matrix at the denominator.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{Number\ of\ correctly\ classified\ samples}{Total\ number\ of\ samples} \tag{6.11}$$

6.12.4 F1 Score:

F1 is a function of Precision and Recall. F1 Score is needed when you want to seek a balance between Precision and Recall.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.12}$$

6.12.5 Mean Opinion Score

Mean opinion score (MOS) is a measure used in the domain of Quality of Experience, representing overall quality of a system. It is the arithmetic mean over all individual values on a predefined scale that a subject assigns to his/her opinion of the performance of a system quality. Such ratings are usually gathered in a subjective quality evaluation test. In this paper, we have used MOS as performance metric for the evaluation of AR Navigation in the targeted tourism site Bhaktapur Durbar Square.
The MOS is expressed as a single rational number, typically in the range 1–5, where 1 is lowest perceived quality, and 5 is the highest perceived quality. We have used five levels of Likert rating scale as shown below:

Label	Rating
Extremely satisfied	5
very satisfied	4
Somewhat satisfied	3
Not so satisfied	2
Not satisfied at all	1

Table 6.7: Likert rating scale

The MOS is calculated as the arithmetic mean over single ratings performed by human subjects for a given system in a subjective quality evaluation test.

$$\text{MOS} = \frac{\sum_{n=1}^{N} R_n}{NXR_{max}} \tag{6.13}$$

Where, R represents the individual ratings for a given system given by N subjects and R_{max} is the maximum rating in the likert rating scale

Chapter 7

Research and Experiment

This section includes the research and experiment phase of our project and the reason why these explored alternatives were not feasible for implementation in our application use case.

7.1 NLP

7.1.1 BERT Model

BERT is based on Transformer architecture. It is designed to pre-train deep bidirectional representations from an unlabeled text by joint conditioning on both the left and right contexts. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer for a wide range of NLP tasks [27]. For the Question Answering System, BERT takes two parameters, the input question, and passage as a single packed sequence. In our experimentation phase, we tried to fine-tune the pre-trained BERT model on our custom dataset.


```
[
    {
        "context": "Bhairavnath temple is a three-storey rectangular temple dedicated to the god Bhairav (Nasa Dyo), an incarnation of Shiva and the god of
terror and infinite space. The shreds of evidence say that the temple was first inaugurated by King Ananda Deva in the 13th century. But it was not triple-
roofed, then. During the reign of King Vishwa Malla, the temple was gracefully restored in a one-storeyed temple. After, King Jagatjyoti Malla decorated it
with golden roofs. Later on, King Bhupatindra Malla modified it into a triple roofed temple at around  1718 AD and added the seven golden pinnacles in it.
The devasting earthquake of 1934AD collapsed it and optimistically reassembled in the same structure. Talking about myth of this temple According to
ancestors the lord bhairava of Varanasi once visited Bhaktapur to see Bisket jatra. During the festival, the lord Bhairava somehow was recognized by a
priest and he just cast a decoy spell on him.  As a quick reaction, he started to sink into the terra firma to go back to Varanasi. But the priest didn
\u2019t let him go. Terra firma means supernatural attempt to leave the current location. As if, he grabs the head of Bhairava in his hand and decollates
him with his sword. That\u2019s why people say that the head doesn\u2019t match the body of Kasi Vishwanath. It is also believed that the head is still
treasured in a box of this temple. There is no entrance on the temple's lengthy west facade that faces the square. Instead, priests access the interior from
the south. ",
        "qas": [
            {
                "id": "1",
                "is_impossible": false,
                "question": "What is the Bhairavnath temple dedicated to, and who is Bhairav?",
                "answers": [
                    {
                        "text": "Bhairavnath temple is a three-storey rectangular temple dedicated to the god Bhairav (Nasa Dyo), an incarnation of Shiva
                                and the god of terror and infinite space. ",
                        "answer_start": 0
                    }
                ]
            },
            {
                "id": "2",
                "is_impossible": false,
                "question": " Who initially inaugurated the Bhairavnath temple, and when was it first inaugurated?",
                "answers": [
                    {
                        "text": " The shreds of evidence say that the temple was first inaugurated by King Ananda Deva in the 13th century",
                        "answer_start": 162
```


Figure 7.1: custom dataset format for fine-tuning BERT Model

BERT Model has a limit of 512 tokens in each context of the training dataset, so multiple contexts are required to include data about the same monument. Since the model requires context and question as input parameters, the problem occurred in identifying the context paragraph that contains the answer to the user's query input before even running the model which made this approach infeasible as a solution.

7.1.2 Word2vec Model

Word2vec is a technique in NLP for obtaining vector representations of words. These vectors capture information about the meaning of the word and their usage in context. The word2vec algorithm estimates these representations by modeling text in a large corpus. It represents each distinct word with a particular list of numbers called a vector. simple mathematical function cosine similarity can indicate the level of semantic similarity between the words represented by those vectors.

In our experimentation phase, we separated the sample dataset into chunks of text, each chunk consisting of a sentence. We further removed stopwords to clean the texts and initially assigned unique random numbers to each word storing them in a Python dictionary. Finally, BoW (Bag-of-Words) representation as vector embeddings was obtained from each chunk of data using the genism python library as shown in figure 7.2.

```
bhairavanath temple is one of the temples that lies at the central of bhaktapur taumadhi square and bhairavanath temple is built in the pagoda style and
it is threestorey
[(0, 2), (1, 1), (2, 2), (3, 1), (4, 1), (5, 1), (6, 1), (7, 3), (8, 1), (9, 1), (10, 2), (11, 1), (12, 1), (13, 1), (14, 1), (15, 1), (16, 2), (17, 1),
 (18, 1), (19, 3), (20, 1)]

it is one of the temple devoted to lord bhairava is one of the rectangular based temples of bhaktapur
[(3, 1), (7, 2), (8, 1), (10, 3), (11, 2), (16, 1), (17, 1), (19, 2), (21, 1), (22, 1), (23, 1), (24, 1), (25, 1), (26, 1)]

curiously there is no entrance on the temples lengthy west facade that faces the square
[(7, 1), (13, 1), (17, 1), (18, 1), (19, 2), (27, 1), (28, 1), (29, 1), (30, 1), (31, 1), (32, 1), (33, 1), (34, 1), (35, 1)]

instead priests access the interior from the south
[(19, 2), (36, 1), (37, 1), (38, 1), (39, 1), (40, 1), (41, 1)]

the west facade is largely symmetrical and features a groundlevel shrine with bhairav depicted in miniature on a small metal throne
[(0, 1), (6, 1), (7, 1), (19, 1), (29, 1), (33, 1), (35, 1), (42, 2), (43, 1), (44, 1), (45, 1), (46, 1), (47, 1), (48, 1), (49, 1), (50, 1), (51, 1),
(52, 1), (53, 1), (54, 1)]

michael hutt notes that this small statue is frequently stolen and replaced
[(0, 1), (7, 1), (18, 1), (51, 1), (55, 1), (56, 1), (57, 1), (58, 1), (59, 1), (60, 1), (61, 1), (62, 1)]
```

Figure 7.2: chunk of text data with corresponding BoW vector embeddings

The vector embedding of the user's input query is generated (figure 7.3) and cosine similarity between vector embedding of question and data is calculated (figure 7.4) to return the most relevant content.

```
question = "how many storey is there in this temple?"
```

```
question = clean_sentence(question, stopwords=False)
question_embedding = dictionary.doc2bow(question.split())
print(question, "\n", question_embedding)
```

```
how many storey is there in this temple
 [(6, 1), (7, 1), (16, 1), (34, 1), (62, 1), (272, 1), (273, 1)]
```

Figure 7.3: Vector embedding of sample question

```
0.8186062258494451  bhairavanath temple is one of the temples that lies at the central of bhaktapur taumadhi square and
                    bhairavanath temple is built in the pagoda style and it is threestorey

0.7693010112265417  it is one of the temple devoted to lord bhairava is one of the rectangular based temples of bhaktapur

0.7451237151570229  curiously there is no entrance on the temples lengthy west facade that faces the square

0.5504000566063658  instead priests access the interior from the south

0.6159001459956239  the west facade is largely symmetrical and features a groundlevel shrine with bhairav depicted in
                    miniature on a small metal throne

0.5256848392314127  michael hutt notes that this small statue is frequently stolen and replaced

0.48218834397757687 to either side of the deity are hammered copper plates depicting bhairavs eyes two pairs of hands
                    and two dogs and his faithful companions

0.5801391764122583  at present perhaps to discourage theft the groundlevel shrine is blocked from casual access by a
                    makeshift wooden barricade

0.6739058113413318  the temple is dedicated to lord bhairav which is supposed to be the most ferocious form of lord shiva

0.6137632183759392  the bhairavanath established in this temple is a formidable incarnation of lord shiva who also prefers
                    to name kasi vishwanath and aakash bhairava
```

Figure 7.4: Cosine similarity between question embedding and chunk data embedding with corresponding chunk data

However, Word2vec only returns the token from the dataset that has the highest cosine similarity with question embedding as shown in figure 7.5. Also since the answers were tokenized on sentence level, which limited the model to answer specific sentences every time for a single question plus no answer longer than a sentence. Also due to the close similarity among multiple sentences of different monuments performance on multiple monument datasets was poor.

```
print("Question: ", question)
print("Answer: ", cleaned_sentences_with_stopwords[index])

Question:  how many storey is there in this temple
Answer:  bhairavanath temple is one of the temples that lies at the central of bhaktapur taumadhi square and
bhairavanath temple is built in the pagoda style and it is threestorey
```

Figure 7.5: output of Word2vec Model

7.1.3 Pre-trained Custom Transformer Model

The pre-trained custom Transformer Model is a question-answering model based on Transformer architecture as shown in table 7.1. We studied the model and prepared a custom dataset for monuments as shown in figure 7.6 and trained the model. We also tried to fine-tune the model for our specific use case. However, the performance of the model was poor during model testing. The model was not complex enough for our use case which resulted in low performance on validation. The model was used for the classification of query rather than token prediction which limited its usage for the large number of questions per monument and addition of multiple monuments.

Model:Pre-trained Transformer Model		
Layer	**Output Shape**	**Param #**
embedding_2 (Embedding)	(None, 20, 16)	16000
transformer_block_2 (TransformerBlock)	(None, 20, 16)	1137
global_average_pooling1d_2 (GlobalAveragePooling1D)	(None, 16)	0
dense_12 (Dense)	(None, 16)	272
dense_13 (Dense)	(None, 16)	272
dense_14 (Dense)	(None, 28)	476
Total params: 18157		
Trainable params: 18157		
Non-trainable params: 0		

Table 7.1: Fine Tuned Question Answer based Transformer Model


```
{
    "intents": [
      {
        "tag": "query1",
        "patterns": [
          "What is the location of Bhairavanath Temple within Bhaktapur?",
          "Where is the Bhairavanath temple located?",
          "Could you describe where the Bhairavanath Temple is in Bhaktapur?",
          "I'm trying to find out the specific location of the Bhairavanath Temple in Bhaktapur.",
          "Can you pinpoint where exactly the Bhairavanath Temple is situated within Bhaktapur?",
          "Hey, do you happen to know the exact spot of the Bhairavanath Temple in Bhaktapur?"
        ],
        "responses": [
          "Bhairavanath temple is one of the temples that lies at the central of Bhaktapur, Taumadhi Square."
        ]
      },
      {
        "tag": "query2",
        "patterns": [
          "What architectural style characterizes the construction of Bhairavanath Temple?",
          "How would you define the architectural style seen in the construction of Bhairavanath Temple?",
          "What specific architectural features define the construction style of Bhairavanath Temple?",
          "Can you describe the architectural design that's typical of Bhairavanath Temple's construction?",
          "In what architectural manner was the Bhairavanath Temple built?",
          "what is unique thing about the Bhairavanath temple?",
          "What is the architecture of the Bhairavanath temple?"
        ],
        "responses": [
          "Bhairavanath Temple is built in the pagoda style and It is three-storey."
        ]
      },
      {
        "tag": "query3",
        "patterns": [
```


Figure 7.6: Custom Dataset format for pre-trained custom Transformer Model

7.2 AR Navigation

7.2.1 Geospatial Creator

Geospatial Creator, powered by ARCore and Google Maps Platform, empowers to visualize, build, and launch robust and engaging 3D digital content in real-world locations through Photorealistic 3D Tiles (map tiles that contain Google's 3D geodata). Geospatial Creator with Unity allows selecting a location, and getting the 3D geometry in an area which can be utilized to develop an augmented reality experience. The ARCore API and Map Tiles API provided by Google Cloud Platform (GCP) is compatible with Unity for placing a 3D object or asset in a desired geographical location using Geospatial Creator Anchor. The GCP also provides $300 USD credit to newly created account for period of 90 days.

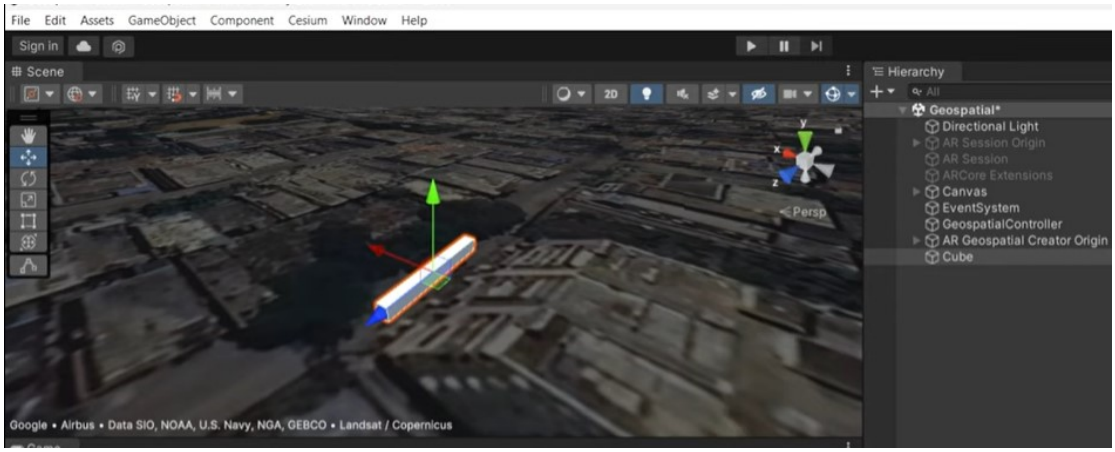


Figure 7.7: Placing 3D asset on map using Unity

In our research phase, we also explored a free 3D asset of the arrow on Sketchfab to use as a Navigation sign. However, during implementation, we faced a problem creating a Google Cloud Platform account while linking a credit card to a Google account.

7.2.2 Vuforia Engine

Vuforia Engine's Area Target is an environment-tracking feature that enables you to track and augment areas and spaces. By using a 3D scan as an accurate model of the space to create an Area Target Device Database augmentations to stationary objects in the scanned environment can be delivered. This enables the creation of navigation applications, and spatial instructions that are all using the surroundings as interactive elements. In the paper [28], Vuforia Engine's Area Target is used to create a practical application of Indoor AR navigation. However, during our research, we learned that the Basic Free plan of Vuforia only provides 20 Area Targets for prototype use which is not enough to cover our targeted tourism site Bhaktapur Durbar Square. We learned about the Matterport mobile application for creating 3D scans which provides one active space with 200 scan points for subscription-free accounts that can cover about 930 meters square. However, Vuforia Area Targets only supports scans specifically made from:

- ARKit enabled selected ios devices with inbuilt LiDAR sensors
- Matterport, in particular with the Pro2 3D and the Pro3 3D camera
- NavVis M6 and VLX scanners
- Leica BLK360 and RTC360 scanners

The requirement of a specific device for 3D scans and a limited number of Area targets made this approach infeasible for our use case.

Chapter 8

Result and Discussion

8.1 System Testing

System testing is a type of software testing that evaluates the entire system as a whole, rather than individual components or modules. The purpose of system testing is to ensure that the system meets its requirements and is functioning as intended before it is released to users or customers. System testing is an important part of the software development lifecycle, as it helps to ensure that the system is ready for release and that it meets the needs of users and stakeholders We did the system testing in two types, as described below.

8.1.1 Unit Testing

Unit testing is a type of software testing that focuses on verifying the functionality of individual units or components of a software system. We developed separate units of our system and performed tests as these units were being developed. After inspecting the results of these tests, we made the necessary modifications and improvements in these units before integrating all the units. Test cases for our different units are shown below:

8.1.1.1 AR Navigation Component

We tested our AR Navigation system in different locations around the Bhaktapur Durbar Square area.

Test Scenario ID	AR Navigation Component
Test Case Description	Check Maps interface and direction guidance accuracy of Rendering of AR navigation signs.
Pre-conditions	Device Location on, Camera access and Internet access

Test execution steps:

1. Open mobile app
2. Click on 'Select A Destination' Button
3. Check scrollbar action in Bottom Sheet
4. Click on the image of a destination
5. Check the destination marked on the map
6. Alternatively tap on the map to select a destination

7. Click on the Go button
8. Check camera feed from a mobile device
9. Follow the rendered AR Navigation Sign and check its guidance accuracy.

S.N	**Action**	**Expected Output**	**Actual Output**	**Result**
1	Open the mobile app and click on Select A Destination button	Bottom Sheet pops on the bottom of the screen	Appendix B.1 figure 10.4	Pass
2	Scroll Bottom Sheet UI in the horizontal and vertical direction.	Smoothly scrolls through Destination image buttons.	Appendix B.1 figure 10.5 and 10.6	Pass
3	Click on an image of a Destination	Display the shortest route from the User location to the selected destination on the map	Appendix B.1 figure 10.7	Pass
4	Tap on the map to select a destination	Display shortest route from User location to destination tapped on map interface	Appendix B.1 figure 10.8	Pass
5	Click on Go Button	AR navigation sign render on input camera feed in the desired direction	Appendix B.1 figure 10.9 and figure 10.10	Pass
6	Check AR Navigation sign around the corner for accuracy	Render two AR navigation signs accurately along the road path	Appendix B.1 figure 10.11 and figure 10.12	Pass

Table 8.1: Alpha Testing of AR Navigation Component

8.1.1.2 MobileNet-SSD v2 Model

Test Scenario ID	MobileNet-SSD v2 Module
Test Case Description	Check detection module with image as input
Pre-conditions	Valid Image with only one monument

Test execution steps:

1. Open module
2. Insert Image name in which objects are to be detected
3. Run the module

S.N	**Input**	**Expected Output**	**Actual Output**	**Result**
1	Image of Nyatapola	Bounding box around monument in image	Appendix B.2 figure 10.13	Pass
2	Image of Vastala Devi Temple	Bounding box around monument in image	Appendix B.2 figure 10.14	Pass
3	Image of Bhairavnath	Bounding box around monument in image	Appendix B.2 figure 10.16	Pass
4	Image of Gopinath temple	Bounding box around monument in image	Appendix B.2 figure 10.16	Pass

Table 8.2: Unit test case of Detection Module

8.1.1.3 CNNs-LSTM Model

Test Scenario ID	CNN-LSTM Module
Test Case Description	Check classification Module with image as input
Pre-conditions	Valid Image with only one monument

Test execution steps:

1. Open module
2. Insert Image name in which image is to be classified
3. Run the module

S.N	**Input**	**Expected Output**	**Actual Output**	**Result**
1	Image of Nyatapola	'nyatapola' class with highest probablity	Appendix B.3 figure 10.17	Pass
2	Image of Siddhilaxmi Temple	'siddhilaxmi' class with highest probablity	Appendix B.3 figure 10.18	Pass
3	Image of Bhairavnath	'bhairav' class with highest probablity	Appendix B.3 figure 10.19	Pass
4	Image of Gopinath temple	'gopinath' class with highest probability	Appendix B.3 figure 10.20	Pass

Table 8.3: Unit test case of Detection Module

8.1.2 Integration Testing

Integration testing is a type of software testing that focuses on verifying the interactions between different modules or components of a software system. The purpose of integration testing is to ensure that the system is functioning correctly as a whole, rather than just individual components in isolation. The integration testing under different functionalities and requirements is given in the table below. The current system can handle only one request at a time.

Test Scenario ID	AR Tour Companion with Voice Assistance
Test Case Description	Check the mobile app frontend integration with the backend and test the camera-feed input
Pre-conditions	Internet access, Device's Location on, Camera access, Server in running state

The integrated component testing requires the following execution steps:

1. Open mobile app
2. Click on the Monument Detection button
3. Obtain video feed of monument from device's built-in camera
4. Tap on the screen after the predicted bounding box appears
5. Check enabling of UI interface for Chatbot Model
6. Press and hold the microphone button to record speech and release the button to send a POST request
7. Check displayed text and played audio response of Chatbot
8. Click on replay and pause buttons to replay audio and pause audio response respectively

S.N	Action	Expected Output	Actual Output	Result
1	Open the mobile app and click on the Monument Detection button	Bounding box around the monument in the live stream of the device's camera	Appendix figure 10.21 and figure 10.22a	Pass
2	Tap on the bounding box on the screen	Inferencing server returns class label of identified monument & UI interface for Chatbot Model enabled	Appendix figure 10.22b and figure 10.22c	Pass
3	Long press mic button to record speech & release to send POST request	Display text response & play audio response of chatbot Model	Appendix figure 10.23a and figure 10.23b	Pass
4	Click on replay button	plays audio response again	Replays audio	Pass
5	Click on pause button	stop playing audio response if playing	Pause playing audio	Pass

Table 8.4: Integration test of Monument Recognition & Chatbot Models

8.2 Model Evaluation Result

8.2.1 Evaluation of Transformer Model

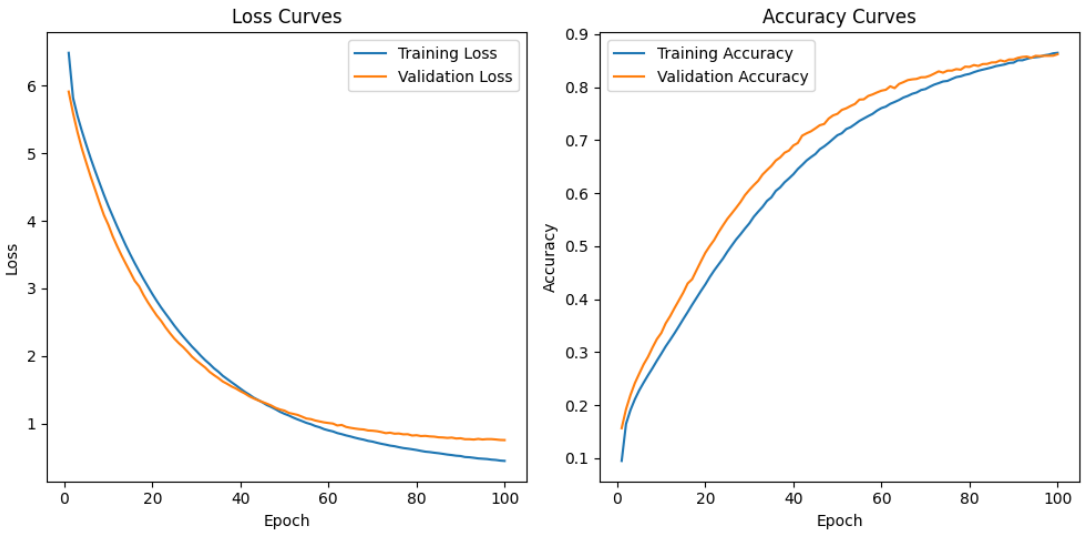


Figure 8.1: Training Loss Graph and Training Accuracy Graph of Transformer model

Example: Consider a scenario with the following output and target sequences
Output: [0.9, 0.1, 0.1, 0.2, 0.7, 0.1, 0.3, 0.4, 0.3]
Target: [1, 0, 0, 0, 1, 0, 0, 0, 1]
Here,
Number of Correct Predictions= 7
Total Number of Non-Padding Tokens= 9
Calculating Accuracy(A) as:

$$\text{A} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Non-Padding Tokens}} \times 100\% = \frac{4}{9} \times 100\% \approx 77.77\%$$

From the validation dataset evaluation, the accuracy of the custom Transformer Model is calculated as 88.48%

8.2.2 Evaluation of MobileNet-SSD V2 Model

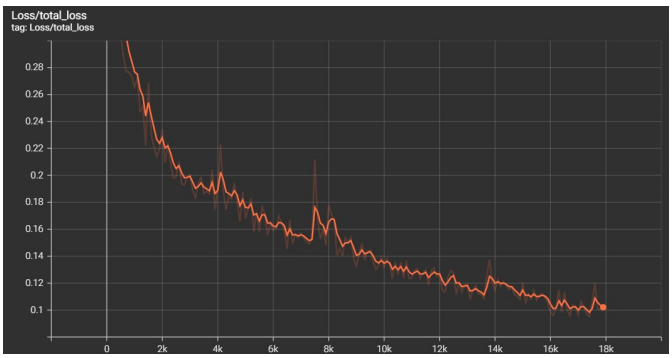


Figure 8.2: Training Loss Graph of MobileNet-SSD v2 model

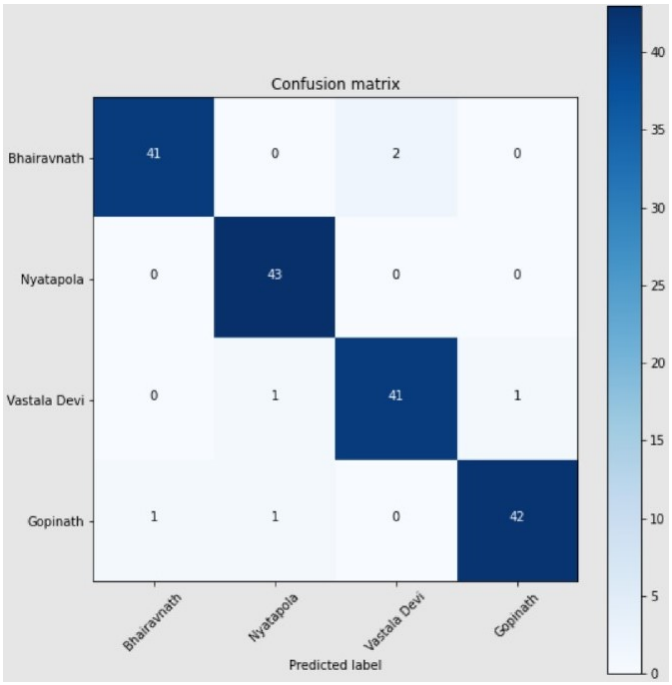


Figure 8.3: Confusion Matrix of MobileNet-SSD v2 Model

SN	Monuments	Accuracy	Precision	Recall	F1 Score
1	Bhairavnath	98.27%	0.98	0.95	0.96
2	Nyatapola	98.84%	0.96	1.0	0.98
3	Vatsala Devi	97.69%	0.95	0.95	0.95
4	Gopinath	98.27%	0.98	0.95	0.97

Table 8.5: Evaluation result of MobileNet-SSD v2 module

Example of calculation of evaluation metrics from the confusion matrix for Vatsala Devi.

True Positive(TP)= 41
True Negative(TN)= (41+43+42+1+1) =128
False Positive(FP)= 2
False Negative(FN)= (1+1)=2

$$1.\text{Precision} = \frac{TP}{TP + FP} = \frac{41}{41 + 2} = 0.95$$

$$2.\text{Recall} = \frac{TP}{TP + FN} = \frac{41}{41 + 2} = 0.95$$

$$3.\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} = \frac{41 + 128}{41 + 128 + 2 + 2} = 0.9769 = 97.69\%$$

$$4.\text{F1 Score} = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * 0.95 * 0.95}{0.95 + 0.95} = 0.95$$

From the test dataset evaluation, the accuracy of the MobileNet-SSD V2 Model is calculated as 96.53%.

8.2.3 Evaluation of CNNs-LSTM Model

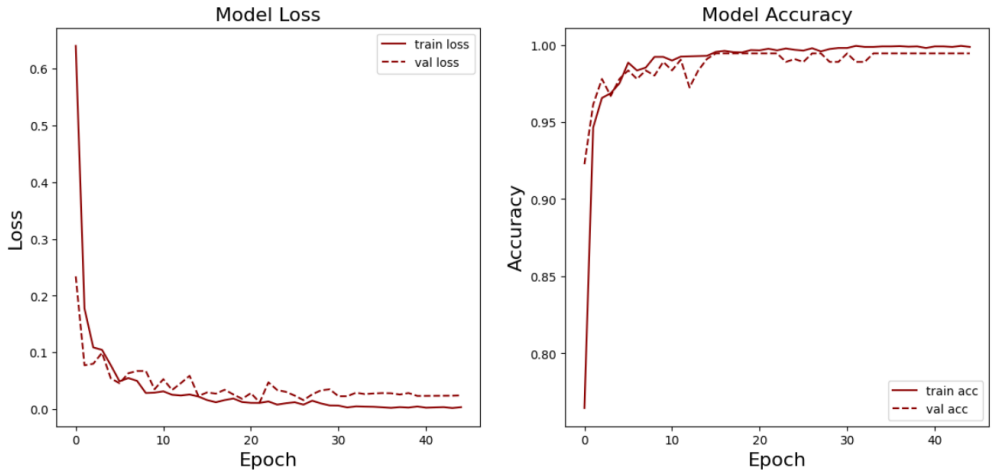


Figure 8.4: Training Loss Graph of CNNs-LSTM Model

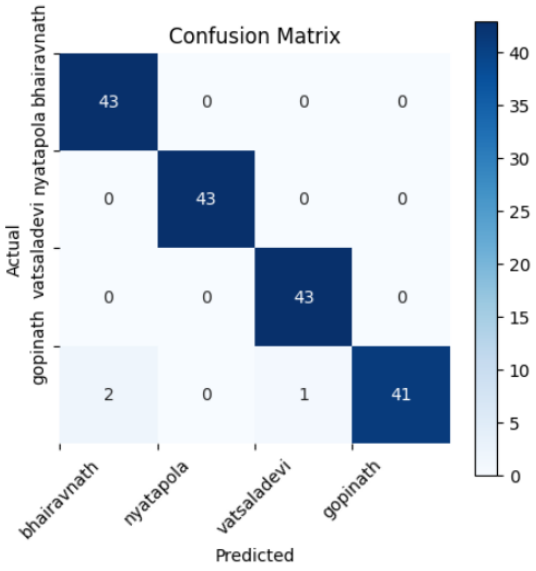


Figure 8.5: Confusion Matrix of CNNs-LSTM Model

SN	Monuments	Accuracy	Precision	Recall	F1 Score
1	Bhairavnath	98.84%	0.96	1.00	0.98
2	Nyatapola	100.00%	1.00	1.00	1.00
3	Vatsala Devi	99.41%	0.97	1.00	0.98
4	Gopinath	98.26%	1.00	0.93	0.96

Table 8.6: Evaluation result of CNNs-LSTM module

Example of calculation of evaluation metrics from the confusion matrix for Vatsala Devi.

True Positive(TP)= 43
True Negative(TN)= (41+43+43+2) =129
False Positive(FP)= 1
False Negative(FN)= 0

$$1.\text{Precision} = \frac{TP}{TP+FP} = \frac{43}{43+1} = 0.97$$
$$2.\text{Recall} = \frac{TP}{TP+FN} = \frac{43}{43+0} = 1.00$$
$$3.\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} = \frac{42+129}{43+129+1+0} = 0.9941 = 99.41\%$$
$$4.\text{F1 Score} = \frac{2*Precision*Recall}{Precision+Recall} = \frac{2*0.97*1.00}{0.97+1.00} = 0.98$$

From the test dataset evaluation, the accuracy of the CNNs-LSTM Model is calculated as 98.26%.

8.2.4 Model Comparision for Image Classification

Various model-tuning actions were performed to increase the accuracy of the model and to increase the stability of the model. Initially, VGG16 and VGG19 were trained. The models were unstable. Combining the VGG16 feature extraction layer and LSTM layer, similarly with VGG19 and LSTM. The models were comparatively stable. Combining the feature extraction layer of VGG16 and VGG19 with LSTM increased the accuracy of the model. Considering Stability and accuracy, the VGG16-VGG19-LSTM model was chosen for the deployment.

SN	Model	Train Accuracy	Test Accuracy	Epoch
1	VGG16-VGG19-LSTM	98.86%	98.44%	45
2	VGG16-LSTM	91.75%	90.34%	25
3	VGG19-LSTM	89.58%	89.45%	21
4	VGG16	86.15%	80.01%	15
5	VGG19	84.70%	78.83%	13

Table 8.7: Comparision of Classification Models

8.2.5 Evaluation of AR Navigation Component

In order to measure the effectiveness of this navigation system, a small sample of 20 students from KhCE was chosen for the system evaluation. The purpose of this test is to determine whether this system could facilitate the effectiveness and usability of navigation on Bhaktapur Durbar Square. Data about user experience was collected from Google Form applying five levels of the Likert rating scale to each assessment question.

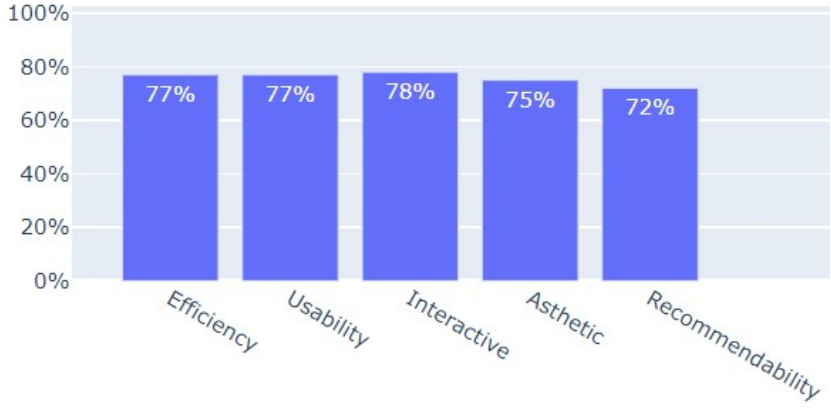


Figure 8.6: MOS evaluation of User Experience on AR Navigation

Example of MOS calculation from the Likert rating scale data for Efficiency:

Number of Users (N) = 20
Maximum rating score (R_{max}) = 5

one person chose not so satisfied (2 score), five people chose somewhat satisfied (3 score), ten people chose very satisfied (4 score) and four people chose extremely satisfied (5 score).

$$\text{MOS} = \frac{\sum_{n=1}^{N} R_n}{N \times R_{max}} = \frac{2 + 5 \times 3 + 10 \times 4 + 4 \times 5}{20 \times 5} = 77\%$$

Chapter 9
Conclusion

When exploring new destinations, people seek to explore a range of aspects such as culture, history, religion, and Architecture to enrich their travel experience. By exploring all these aspects it enhances their traveling experience. Our app facilitates this journey by offering a voice assistant that provides comprehensive insights into the destination's cultural, historical, religious, and architectural significance, and records of earthquake events. By seamlessly integrating audio guidance, users can immerse themselves in the essence of the place they're visiting, enhancing their overall travel experience. Object detection module i.e.MobileNet-SSD v2, the monument recognition module i.e.CNNs-LSTM, NLP model i.e.Transformer, Mapbox Vision SDK, Navigation SDK, and Direction API for AR navigation are successfully implemented in this project.

In a nutshell, the project AR Tour Companion: Enhancing Travel Experiences with Voice Assistance is completed and successfully deployed as an Android application for enhancing tour experiences.

Chapter 10

Limitations and Future Enhancement

10.1 Limitaions

There are number of limitations in our system despite our best attempts to construct a comprehensive and accurate model. These are a few of the limitations:

1. It needs internet access to operate.
2. It can only assist to travel in Bhaktapur Durbar Square.
3. Mapbox sdk only operates in specific android devices.

10.2 Future Enhancement

Even though we have worked very hard to create a solid and useful model, there is always space for development and additions in the future. Some potential areas for future improvement and development include:

1. To expand the usefulness of our system, we can incorporate more datasets.
2. Even though there are two different models for text and images, the same model can be used.
3. To make the model better at understanding local terms, we can train it to translate text to audio and audio to text.
4. We can improve the navigation ability in streets.
5. We can deploy 2d or 3d bot which will be available in screen and interact with the user.

Bibliography

[1] J. Adam, "What is agile software development?" https://kruschecompany.com/agile-software-development/, Nov. 2022.

[2] F. Lu, H. Zhou, L. Guo, J. Chen, and L. Pei, "An arcore-based augmented reality campus navigation system," *Applied Sciences*, vol. 11, no. 16, p. 7515, 2021.

[3] S. Koo, J. Kim, C. Kim, J. Kim, and H. S. Cha, "Development of an augmented reality tour guide for a cultural heritage site," *Journal on Computing and Cultural Heritage (JOCCH)*, vol. 12, no. 4, pp. 1–24, 2019.

[4] S. M. H. Asraf, A. F. M. Hashim, and S. Z. S. Idrus, "Mobile application outdoor navigation using location-based augmented reality (ar)," in *Journal of Physics: Conference Series*, vol. 1529, no. 2. IOP Publishing, 2020, p. 022098.

[5] R. Hashimoto and M. Cohen, "Outdoor navigation system by ar," in *SHS Web of Conferences*, vol. 102. EDP Sciences, 2021, p. 04002.

[6] F. Clarizia, F. Colace, M. Lombardi, and D. Santaniello, "A chatbot for supporting users in cultural heritage contexts," 2020.

[7] B. R. Ranoliya, N. Raghuwanshi, and S. Singh, "Chatbot for university related faqs," in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2017, pp. 1525–1530.

[8] G. Sperlí, "A cultural heritage framework using a deep learning based chatbot for supporting tourist journey," *Expert Systems with Applications*, vol. 183, p. 115277, 2021.

[9] M. Lombardi, F. Pascale, and D. Santaniello, "An application for cultural heritage using a chatbot," in *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, 2019, pp. 1–5.

[10] Y. W. Chandra and S. Suyanto, "Indonesian chatbot of university admission using a question answering system based on sequence-to-sequence model," *Procedia Computer Science*, vol. 157, pp. 367–374, 2019.

[11] A. Crudge, W. Thomas, and K. Zhu, "Landmark recognition using machine learning," *CS229, Project*, 2014.

[12] V. Palma, "Towards deep learning for architecture: A monument recognition mobile app." *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2019.

[13] S. Gada, V. Mehta, K. Kanchan, C. Jain, and P. Raut, "Monument recognition using deep neural networks," in *2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*. IEEE, 2017, pp. 1–6.

[14] M. Etaati, B. Majidi, and M. T. Manzuri, "Cross platform web-based smart tourism using deep monument mining," in *2019 4th International conference on pattern recognition and image analysis (IPRIA)*. IEEE, 2019, pp. 190–194.

[15] R. Fatima, I. Zarrin, M. A. Qadeer, and M. S. Umar, "Mobile travel guide using image recognition and gps/geo tagging: A smart way to travel," in *2016 Thirteenth International Conference on Wireless and Optical Communications Networks (WOCN)*. IEEE, 2016, pp. 1–5.

[16] M. Mishra, "Convolutional neural networks explained," https://shorturl.at/cnB18, Aug. 2020.

[17] O. Sheremet, "Intersection over union (iou) calculation for evaluating an image segmentation model," https://towardsdatascience.com/intersection-over-union-iou-calculation-for-evaluating-an-image-segmentation, Jul. 2020.

[18] S. K, "Non-maximum suppression (nms)," https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c, Oct. 2019.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[20] A. Zugaldia, "Announcing the mapbox navigation sdk v1.0 for android," rb.gy/wmqgz, Jun. 2020.

[21] Z. Wang, J. Feng, and Y. Zhang, "Pedestrian detection in infrared image based on depth transfer learning," *Multimedia Tools and Applications*, vol. 81, no. 27, pp. 39 655–39 674, 2022.

[22] J. Meng, P. Jiang, J. Wang, and K. Wang, "A mobilenet-ssd model with fpn for waste detection," *Journal of Electrical Engineering & Technology*, vol. 17, no. 2, pp. 1425–1431, 2022.

[23] J. McDermott, "Hands-on transfer learning with keras and the vgg16 model," https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/.

[24] A. Khattar and S. Quadri, "Generalization of convolutional network to domain adaptation network for classification of disaster images on twitter," *Multimedia Tools and Applications*, vol. 81, no. 21, pp. 30 437–30 464, 2022.

[25] A. Graves, "Long short-term memory," *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

[26] S. Dobilas, "Lstm recurrent neural networks — how to teach a network to remember the past," https://shorturl.at/bPY02, Feb. 2022.

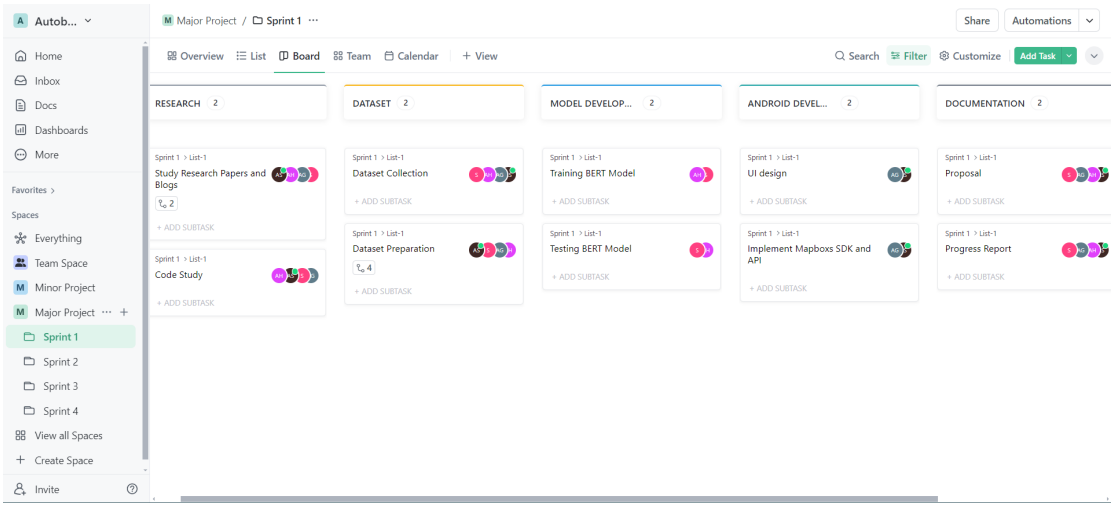
[27] N. Nishanth, "Question answering system with bert," https://rb.gy/1eq6uc, Jul. 2020.

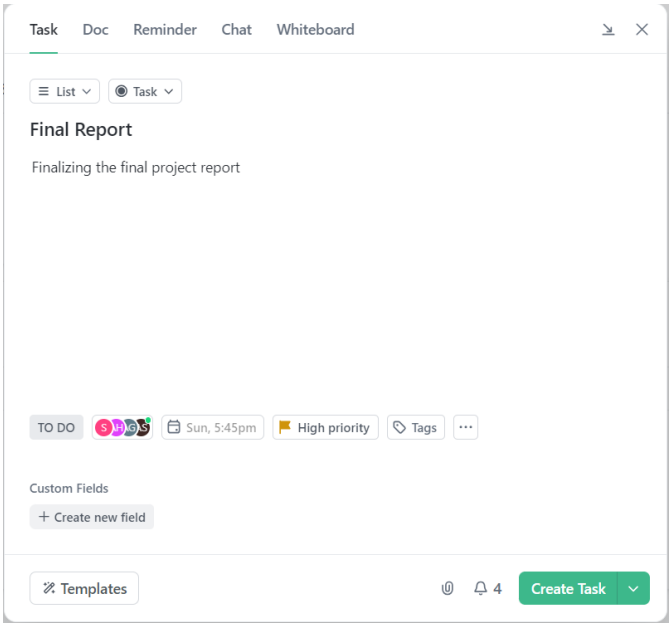
[28] J. Drewlow, M. Däppen, and M. Lehmann, "Navigation with augmented reality in a hospital," *Studies in Health Technology and Informatics*, vol. 292, pp. 111–114, 2022.

Appendix

A Snapshot

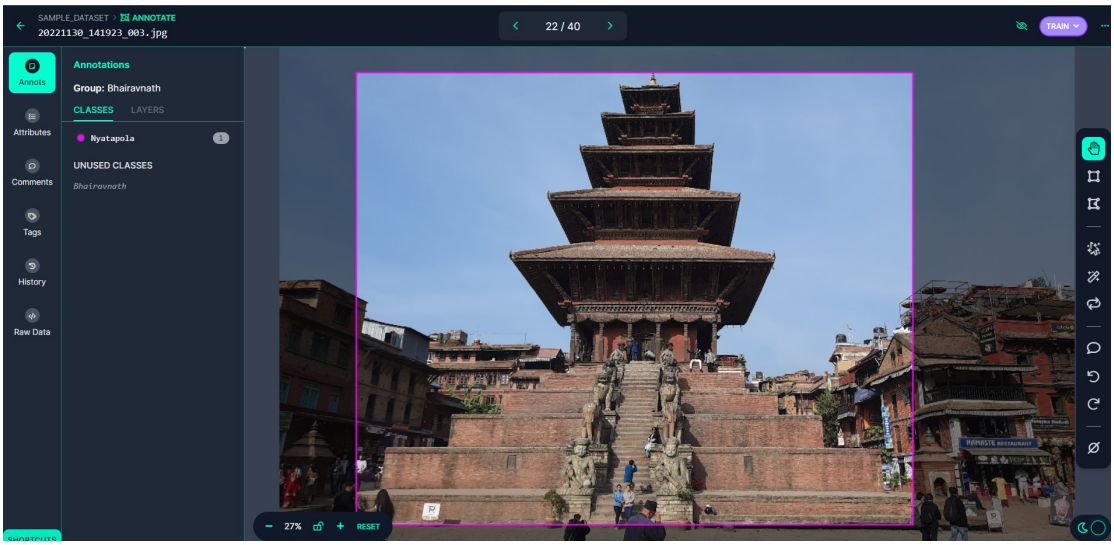
A.1 ClickUp





A.2 Data Annotation

A.2.1 Annotating data for object detection using Roboflow web app



A.3 Data Augmentation for custom Transformer Model

A.3.1 Original Dataset

	A	B
1	**Question**	**Answer**
2	What distinguishes the statue of King Bhupatindra Malla in Bhaktapur?	One notable aspect of the statue is its exceptional metallic craftsmanship, marking it as one of Bhaktapur's fines
3	Where can tourists closely inspect King Bhupatindra Malla's statue?	Tourists can meticulously examine the statue's intricacies at the Taleju Bell , Tagou Ghan.
4	In which direction does King Bhupatindra Malla's statue face at Bhaktapur Durbar Square?	Positioned opposite the Golden Gate, the statue faces northward within Bhaktapur Durbar Square.
5	Describe the posture of King Bhupatindra Malla in his statue.	King Bhupatindra Malla is depicted with crossed legs, joined hands, and a bowed head, a rare portrayal.

A.3.2 Translated to Nepali Language

	A	B
1	**प्रश्न**	**जवाफ**
2	भक्तपुरमा रहेको राजा भूपतिन्द्र मल्लको शालिकले के फरक पार्छ ?	मूर्तिको एक उल्लेखनीय पक्ष यसको असाधारण धातु शिल्पकला हो, यसलाई भक्तपुरको उत्कृष्ट उदाहरणको रूपमा चिन्ह लगाइएको छ।
3	राजा भूपतिन्द्र मल्लको शालिकलाई पर्यटकले कहाँ नजिकबाट हेर्न सक्छन् ?	पर्यटकहरूले तलेजु बेल, त्यागो घाँसमा मूर्तिको जटिलताहरू सावधानीपूर्वक जाँच गर्न सक्छन्।
4	भक्तपुर दरबार स्कायरमा रहेको राजा भूपतिन्द्र मल्लको शालिक कुन दिशामा रहेको छ ?	भक्तपुर दरबार स्कायर भित्र गोल्डेन गेटको बिरुद्धमा रहेको यो मूर्ति उत्तरतर्फ फर्किएको छ।
5	राजा भूपतिन्द्र मल्लको मुर्तिमा रहेको आसनको वर्णन गर्नुहोस्।	राजा भूपतिन्द्र मल्लालाई खुट्टा जोडिएको, हात जोडिएको र झुकेको टाउको, दुर्लभ चित्रण गरिएको छ।

A.3.3 Back Translation from Nepali Language

	A	B
1	**Question**	**Answer**
2	What is the difference between the sculpture of King Bhupatindra Malla in Bhaktapur?	notable aspect of the statue is its extraordinary metalwork, marking it as the best example of Bhaktapur.
3	Where can tourists see the statue of King Bhupatindra Malla up close?	Tourists can carefully examine the intricacies of the sculpture on Taleju Bell, Tagou grass.
4	In which direction is the statue of Raja Bhupatindra Mall in Bhaktapur Durbar Square?	Facing the Golden Gate inside the Bhaktapur Durbar Square, the statue faces north.
5	Describe the seat in the statue of King Bhupatindra Malla.	King Bhupatindra Malla is depicted with folded legs, folded arms and bowed head, a rare depiction.

A.4 Data Augmentation for Monument Recognition Model

A.4.1 Random alteration of brightness between the range of ± 15%

(a) brightness1 (b) brightness2 (c) brightness3

Figure 10.1: Brightness Alteration of Nyatapola Image Data

A.4.2 Random rotation between the range of ± 15%

(a) rotation1 (b) rotation2 (c) rotation3

Figure 10.2: Rotation of Nyatapola Image Data

A.4.3 90° rotation clockwise and anti-clockwise

(a) anti-clockwise rotation (b) clockwise rotation

Figure 10.3: Anti-clockwise and Clockwise Rotation of Nyatapola Image Data

B Unit Test

B.1 Unit Testing AR Navigation Component

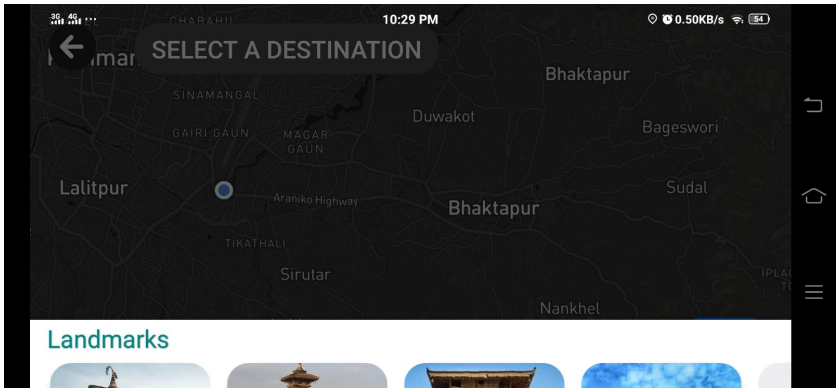


Figure 10.4: Bottom Sheet UI interface

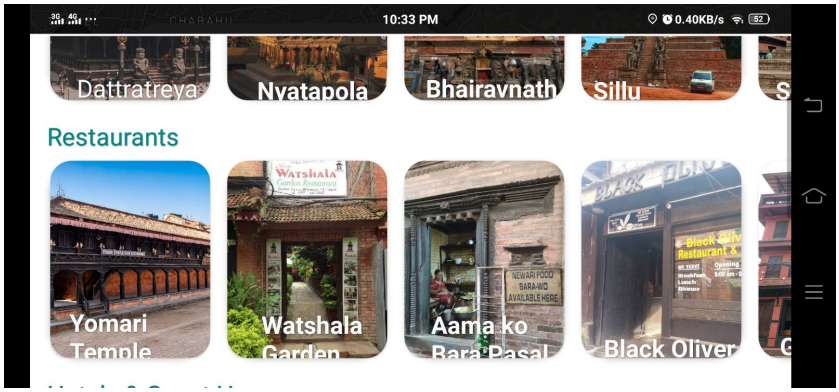


Figure 10.5: Scrollbar action in vertical direction

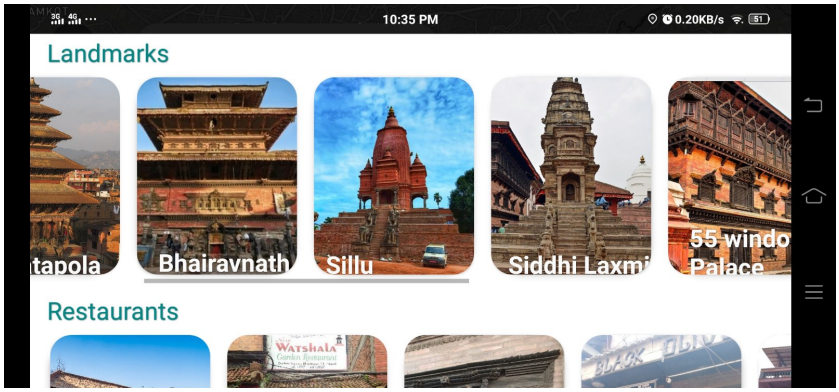


Figure 10.6: Scrollbar action in horizontal direction

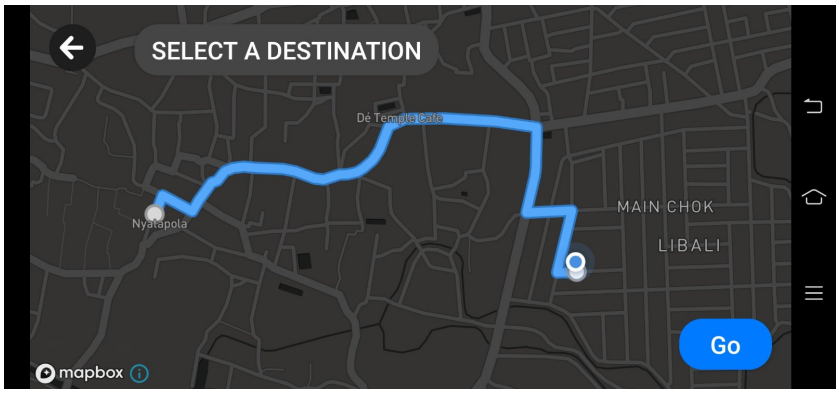


Figure 10.7: Tapping on image button to select destination

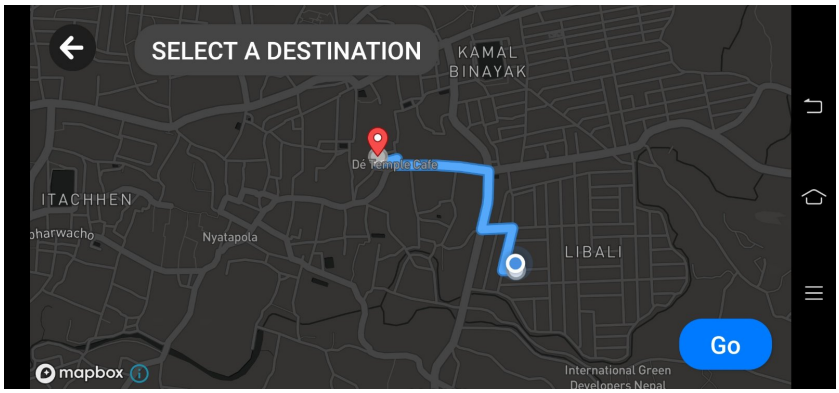


Figure 10.8: Tapping on Map Interface to select destination

Figure 10.9: Rendering of AR Lane on straight path 1

Figure 10.10: Rendering of AR Lane on straight path 2

Figure 10.11: Rendering of AR Lane around corner

Figure 10.12: Rendering of AR Lane and AR Fence around corner

B.2 Unit Testing MobileNet-SSD v2

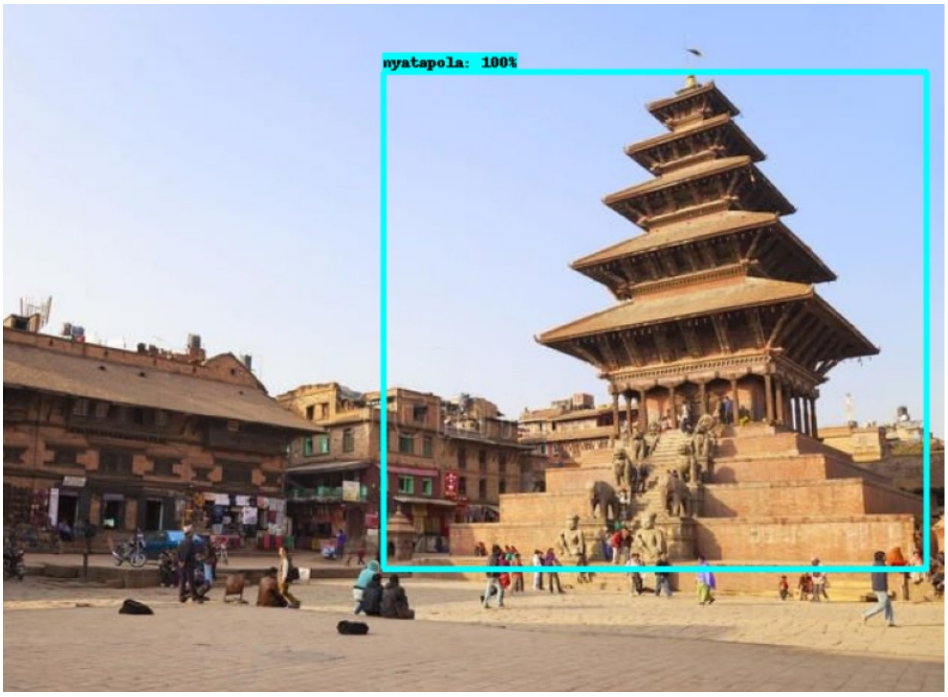


Figure 10.13: Unit test 1 for detection module

Figure 10.14: Unit test 2 for detection module

Figure 10.15: Unit test 3 for detection module

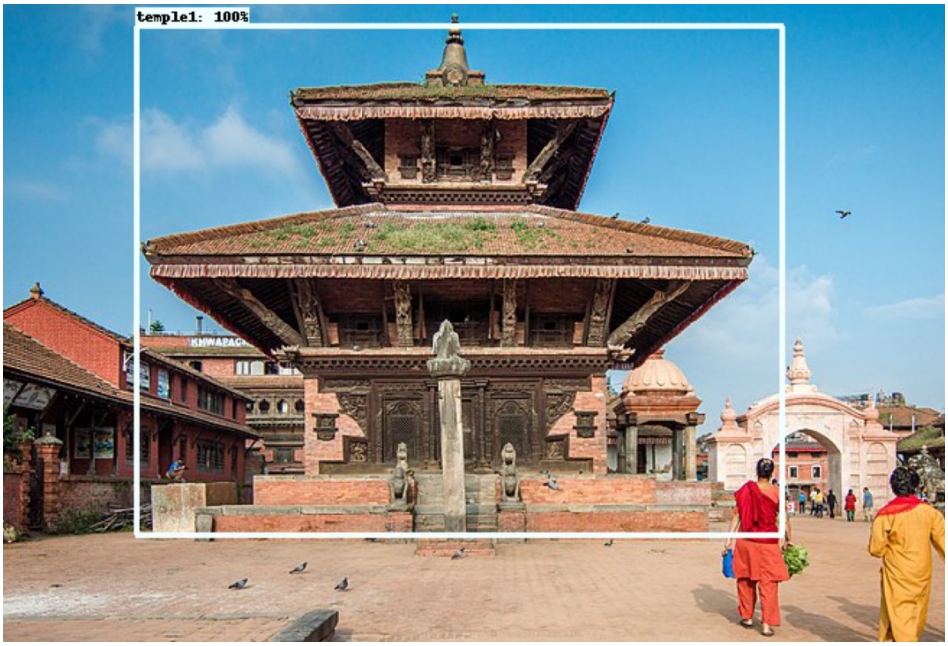


Figure 10.16: Unit test 4 for detection module

B.3 Unit Testing CNNs-LSTM

```
classify(img)
```

```
1/1 [==============================] - 0s 55ms/step
'nyatapola'
```

Figure 10.17: Unit test 1 for Classification module

```
classify(img)
```

```
1/1 [==============================] - 0s 39ms/step
'vatsaladevi'
```

Figure 10.18: Unit test 2 for Classification module

```
classify(img)
```

```
1/1 [==============================] - 0s 39ms/step
'Bhairav'
```

Figure 10.19: Unit test 3 for Classification module

```
classify(img)
```

```
1/1 [==============================] - 0s 31ms/step
'Gopinath'
```

Figure 10.20: Unit test 4 for Classification module

C Integration Test

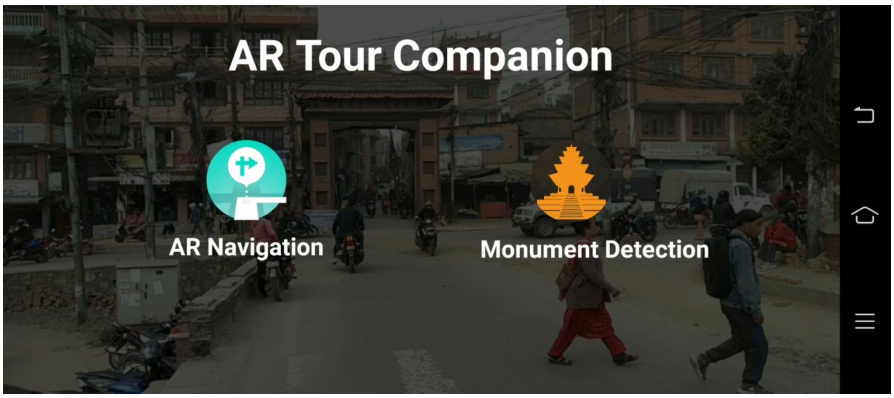
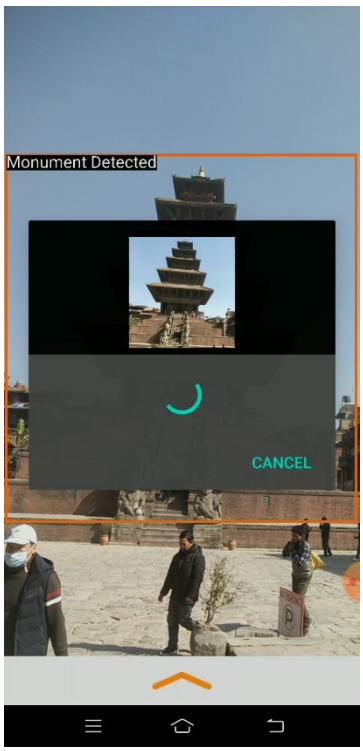
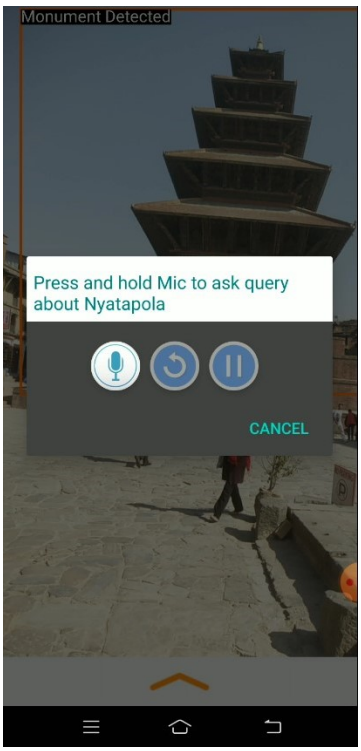


Figure 10.21: open mobile app

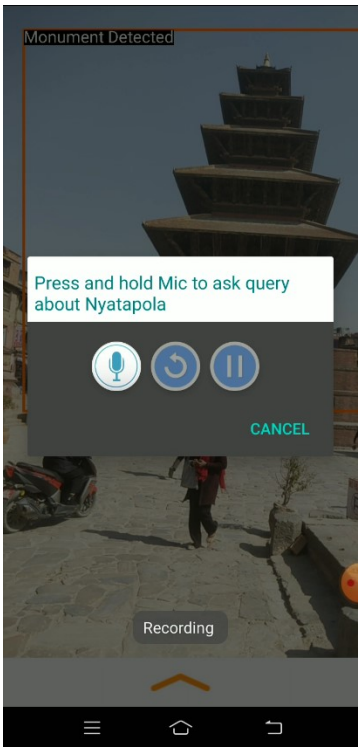
(a) Bounding Box



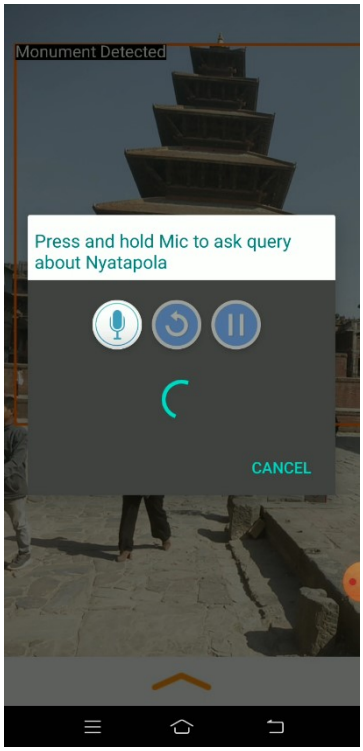
(b) Sending POST Request



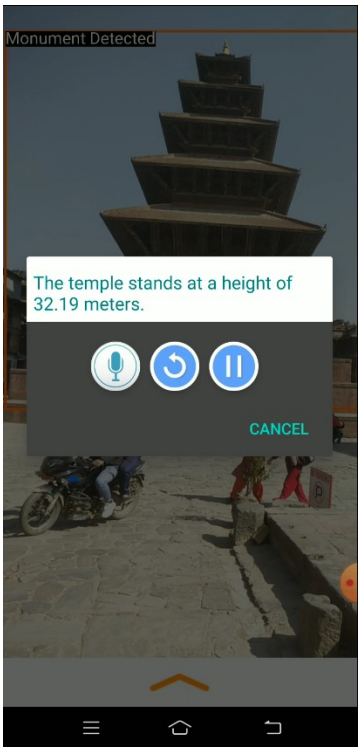
(c) Chatbot UI Interface



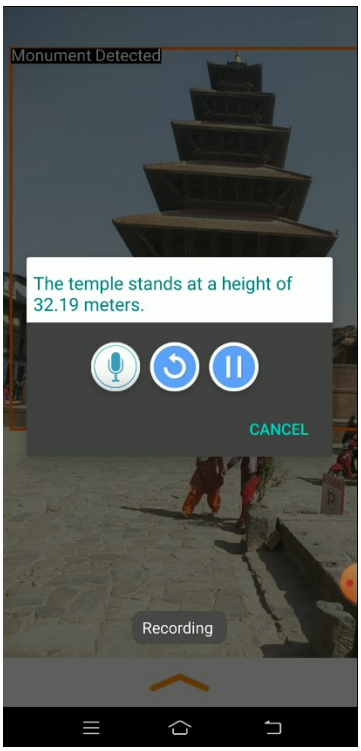
(a) Input Audio Query



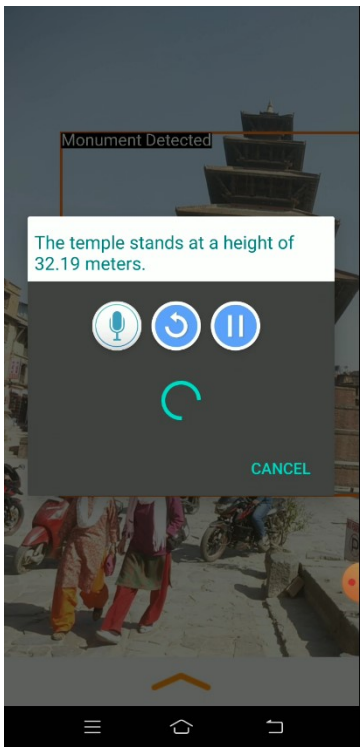
(b) Sending POST Request



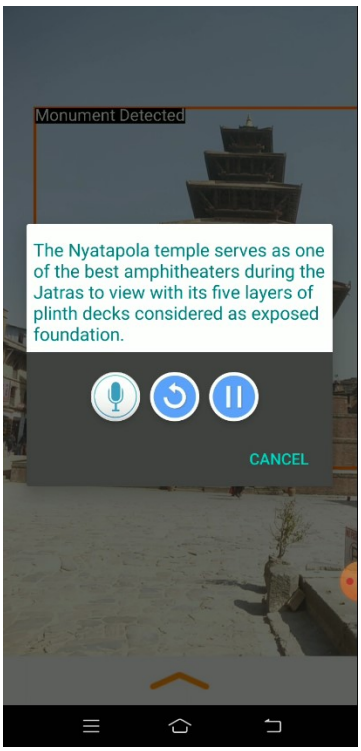
(c) Server Response



(a) Input 2nd Query



(b) Sending POST Request



(c) Server Response