TRIBHUVAN UNIVERSITY
INSTITUTE OF ENGINEERING

Khwopa College Of Engineering
Libali, Bhaktapur
Department of Computer Engineering

A FINAL REPORT ON

MONUMENT RECOGNITION SYSTEM USING CNN

Submitted in partial fulfillment of the requirements for the degree

BACHELOR OF COMPUTER ENGINEERING

Submitted by

Abhishek Hyangol	KCE076BCT003
Amit Shrestha	KCE076BCT004
Jenish Twayana	KCE076BCT019
Samshrita Ghimire	KCE076BCT039

Under the Supervision of
Er.Mukesh Kumar Pokharel
Department Of Computer Engineering

Khwopa College Of Engineering
Libali, Bhaktapur
2022-23

Certificate of Approval

This is to certify that this minor project work entitled "Monument Recognition System Using CNN" submitted by Abhishek Hyangol (KCE076BCT003), Amit Shrestha (KCE076BCT004), Jenish Twayana (KCE076BCT019) and Samshrita Ghimire (KCE076BCT039) has been examined and accepted as the partial fulfillment of the requirements for the degree of Bachelor in Computer Engineering.

.....
Er. Santosh Giri
External Examiner
Assistant Professor,
Department of Electronic and Computer Engineering,
IOE, Pulchowk Campus

.....
Er.Mukesh Kumar Pokharel
Project Supervisor
Assistant Lecturer,
Department of Computer Engineering,
Khwopa College of Engineering

.....
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 computer department, Er.Dinesh Man Gothe sir for his advice, encouragement and suggestions for this project.We are also deeply indepted to Er.Niranjan Bekoju for his valuable suggestions and online training program. Also we would like to thanks our co-ordinator Er.Bindu Bhandari for her support. We would like to express our gratitude to our supervisor Er.Mukesh Kumar Pokharel for his supervision, encouragement and advices during the course of this project.

We gratefully acknowledge help of Aaju Prajapati, Anish Shrestha, Ankit Kayastha, Anuj Gaida, Rahul Khatri and Safal Raj Manandhar during dataset collection. We would also like to thank all our classmates who helped us during the course of the project. Finally we would like to acknowledge everyone who played a role directly or indirectly for completing the project.

Abhishek Hyangol	KCE076BCT003
Amit Shrestha	KCE076BCT004
Jenish Twayana	KCE076BCT019
Samshrita Ghimire	KCE076BCT039

Abstract

Monument Recognition is a practical application of deep learning and computer vision, and can be used to detect and recognize the monuments of Bhaktapur Durbar Square using static image datasets. In this paper, we have developed an android application which can detect monument in the image using MobileNet-SSD v2 and recognize the detected monument using YOLOv7 CNN algorithm in real time. The working mechanism of Monument Recognition System involves four main stages, they are Monument detection, segmentation, Monument Recognition and displaying description of recognized monument from database. For better accuracy of recognition system we have implemented Semantic Segmentation using YOLOv7 model. The dataset collected were annotated using roboflow web app for semantic segmentation. Besides that, the technique of comparing mobile device's live location with the monuments geographical coordinates is also implemented for enhancing accuracy.

Keywords: *CNN, YOLO, MobileNet-SSD v2, Transfer Learning, Monument Recognition, Mobile App*

Contents

List of Tables

List of Figures

List of Symbols and Abbreviation

BN	Batch Normalization
CAT	Concatenation Attention
CBS	Cascade, Branch, Merge
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Computer Vision
DCNN	Deep Convolutional Neural Network
E-ELAN	Extended Efficient Layer Aggregation Network
ELAN	Efficient Layer Aggregation Network
ELAN-W	Efficient Lightweight Anchor-Free Network with Weighted Feature Fusion
GPU	Graphical Processing Unit
HOG	Histogram of Oriented Gradients
MP-1	Multipath-1
NiN	Network in Network
ORB	Oriented FAST and Rotated BRIEF
RAM	Random Access Memory
ReLU	Rectified Linear Unit
REP	Reparameterized Convolutional Block
ROM	Read Only Memory
SPPCSPC	Spatial Pyramid Pooling and Cascaded Spatial Pyramid Convolution
SSD	Single Shot MultiBox Detector
SVM	Support Vector Machine
VGG	Visual Geometry Group
YOLO	You Only Look Once

Chapter 1

Introduction

1.1 Introduction

Monument Recognition System is designed to detect monument from image and recognize the monument as well as provide information regarding historical, architectural, religious and cultural aspects of the recognized monument. In recent years, there has been various studies on monument and landmark recognition across the globe but none in the context of Nepal. This system is deployed as an android app with the motive to promote tourism and attract attention to preserve cultural heritage of Nepal. In this system we have included few monuments of Bhaktapur Durbar Square for training our object detection and recognition models. Furthermore, to enhance the prediction accuracy of the system technique like comparing the geographical coordinates of a monument with the current location of user is implemented. Since, Monument Recognition system is deployed as mobile application anyone with smart phone and internet connection has privilege to use the system. Therefore, Monument Recognition system can be useful as a virtual tour guide to domestic as well as foreign tourists. Additional feature of guiding users to nearby local services like hotels,ATMs,etc makes the system a fully functional guide.

1.2 Motivation

Nepal is a country with flourishing tourism industry.But most of the country people occasionally visiting the cultural heritage sites do not hire tourist guide. In context of Nepal, system related monument recognition hasn't been developed. This was one of the motivating factor 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 hardcopy format they get lost at some point. Similarly, general public seems to have poor knowledge and declined interest on 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 propose a system that detects the monument and shows its historical information. 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 that is easily available to everyone.

1.4 Objective

1.4.1 General Objective

- To build a mobile application that can identify and provide description of the recognized monument.

1.4.2 Specific Objective

- To click photos of monuments.
- To augment and preprocess it for training machine learning model.
- To implement the convolutional neural network and train it with the collected training dataset.
- To evaluate the performance measurement of implemented CNN.
- To integrate all the components of system and perform integration testing.

1.5 Scope and Applications

The major scope of this project is to provide information of detected monument in an image or video frame and the system can have following application :

- Virtual Tour guide
- Digital booklet

1.6 Limitations

Due to lack of resources like GPU, dataset and time our system has following limitations:

- As we have used only four monuments(Nyatapola, Bhairavnath, Gopinath and Vatsala Devi temple) of Bhaktapur Durbar Square in dataset our system can only detect and recognize these monuments.
- Currently our detection model can only detect one monument in given input image.
- Needs internet connection to run.

Chapter 2

Literature Review

Monument Recognition system has been in state of research and development from past seven years but in the context of Nepal due to lack of proper dataset no related research has been made. In this feild, the system has significantly improved over years with evolution of image recognition using different machine learning and deep learning techniques like SVM,CNN,DCNN,etc.

2.1 Landmark Recognition Using Machine Learning

In this paper [2], 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%

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

This project [3] 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 Monument Recognition using Deep Neural Networks

In this paper [4], 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% .

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

The proposed platform [5] 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.

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

This mobile app [6] 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.

2.6 Image based Indian Monument Recognition using Convoluted Neural Networks

This paper [7] experiments on different method of extraction of features using hand crafted features like HOG, LBP and GIST and then CNN. The manually acquired dataset comprises of 100 different monuments with 50 images per monument. HOG, LBP and GIST features extracted on training dataset classified by classification algorithms like SVM obtained very low accuracy. Finally, DCNN model obtained accuracy of 92.7% using fc6 layer.

S.N	Title	Summary
1	Landmark Recognition Using Machine Learning [2]	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 [3]	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 [4]	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 [5]	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 [6]	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.
6	Image based Indian Monument Recognition using Convoluted Neural Networks [7]	This paper experiments on different method of extraction of features using hand crafted features and CNN. The models trained on manually obtained dataset of 50 images per monument accuracy of 92.7% on DCNN using fc6 layer.

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

Chapter 3

Requirement Analysis

3.1 SOFTWARE REQUIREMENT

Software requirement for our prepared system includes:

a. Android Studio

b. FastAPI

c. Google Collaboratory

d. Google Drive

e. IntelliJ IDEA

f. Microsoft Azure

g. Microsoft Team

h. OverLeaf

i. PostgreSQL

j. Postman

k. Python

l. PyTorch

m. Roboflow

n. Tensorflow

o. Tensorflow Lite

p. Visual Studio Code

q. Clickup

3.2 Hardware Requirement

Our prepared system of Monument Recognition requires following hardware requirements:

a. An Android OS based mobile phone for deployment with the:
   - Octa core CPU or above
   - RAM: 2GB or above
   - ROM: 16GB or above
   - Camera: 8 Megapixels or above
   - Android Version 7.0 or above

b. A computer system with the:
   - Intel Xeon Gold Processor 12Cores/24VCores
   - RAM: 32 GB DDR3
   - Storage Device: 512 GB NVMe SSD
   - GPU: CUDA enabled GPU (Graphics card by NVIDIA)

c. Google Colabs GPU '12 GB NVIDIA Tesla K80 GPU'.

d. 12GB Nvidia RTX 3080Ti

3.3 Functional Requirement

These requirements describe the main functionality of the system and services provided to the user. Functional requiremnet of prepared system are:

a. The system must be able to capture image/video in real time.

b. It should detect monument in the video feed in real time.

c. It should recognize the detected monument.

d. It should display information related to recognized monument from the systems database.

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 metrices are conducted to test the accuracy of the system.

3.4.2 Maintainability

The System is breakdown into multiple sub module so that we could know where the problem is and maintain the sub module.

3.4.3 Portability

Since the system in implemented as a mobile application in android smart phone. It is portable.

3.4.4 Performance

The object detection require only single process. So, it will be able to provide better performance in real time.

3.4.5 Usability

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

3.4.6 Availability

User can detect and recognize monuments using on device detection model and sending request to inferencing server hosted at anytime. so, it is available at anytime.

Chapter 4

FEASIBILITY STUDY

4.1 Economic Feasibility

The total expenditure of the project is computational power. The dataset and computational power required for the project are easily available. Dataset is available on different web sites as well as can be collected manually by taking images from smart phone. The processing power of this system is mainly for creating and training deep learning models which requires good GPU and memory. This was done using own laptop along-side Google Collaboratory cloud computing service so, the project is economically feasible.

4.2 Technical Feasibility

We have included YOLO v7 and Mobilenet-SSD v2 weights file to train our custom dataset for semantic segmentation followed by recognition and object detection respectively. We have no problem with models. In terms of availability, the dataset is availabe.Training huge dataset takes a lot of computation power which is available from Google colab. Training the project is also feasible and is better with huge data points and large processing power. So implementing this system seems technically feasible.

4.3 Operational Feasibility

The systems monument detection model is deployed on the android app with very simple UI and the recognition model is deployed on inferencing server of Microsoft Azure. User can run the application and detect monuments from the camera feed ,the object detection model runs constatnly to detect monument present in the video frame and creates bounding box. The user will then upload image frame within bounding box to inferencing server, recognition model runs in the backend and returns required description of the monument.

4.4 Time Feasibility

The project is time feasible as we have reduced the domain of our recognition system to few monuments, thus drastically reducing the amount of dataset required. Furthermore, using transfer learning yeilds better result in few dataset, making it possible to prepare dataset and train the recognition model in the given time frame.

Chapter 5

Methodology

5.1 SOFTWARE DEVELOPMENT APPROACH

The Agile methodology is a method of managing project by breaking it into several phases focusing on releasing products frequently and updating the project as per the requirement. 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 approach for evolution and adaptation welcoming changes at any time of the SDLC process. Scrum, Kanban, Extreme Programming (XP), and Adaptive Project Framework (APF) are some of the development frameworks under Agile project management methodology. For this project we are going to use Scrum framework as it encourages team collaboration and quality software development. Scrum projects are broken down into small and consistent timeboxed iterations of SDLC.

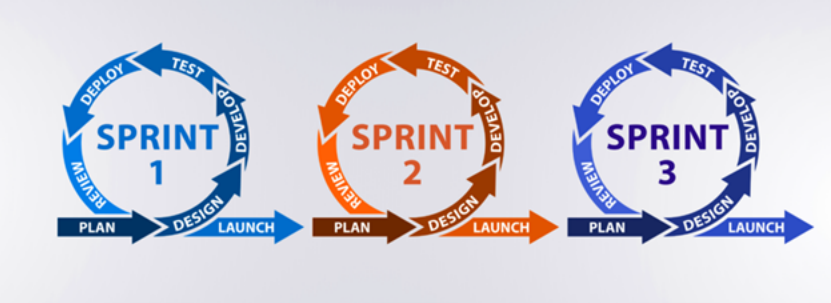


Figure 5.1: Agile Methodology, Scrum Model for Software Development [1]

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. We created tasks as shown in figure A.1.1. First we created a project plan then sprints and completed the tasks form project plan in these sprints.

5.2.1 Product Backlog

Producet Backlog consists of new features, bug fixes and activities of changes during the development phase of the project. The features of product backlog are combined together to create release, and those releases are further combined to form sprint backlog. Our product backlog includes following features:

1. Dataset Collection
2. Data Annotation for Monument Detection
3. Data Labeling for Semantic Segmentation
4. Training MobileNet-SSD v2 model
5. Testing MobileNet-SSD V2 model
6. Training YOLOv7 model
7. Testing YOLOv7 model
8. UI design
9. VM setup on Server
10. Deployment of YOLOV7 on Server
11. Frontend Development
12. Backend Development
13. Model Evaluation and Visualization
14. Deployment of MobileNet-SSD v2 in Mobile app

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

5.2.1.1 Sprint 1

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

1. Dataset Collection
2. Data Annotation for Monument Detection
3. Training MobileNet-SSD v2 model
4. Testing MobileNet-SSD V2 model

At the end of the first sprint we obtained a MobileNet-SSD v2 that can take an image as input and detect monument in that image. This sprint includes data collection and data annotating which had to be done manually.

5.2.2 Sprint 2

We started the second sprint by taking the dataset collected in first sprint for annotation. The sprint 2 consist of following tasks:

1. Data Labeling for Semantic Segmentation
2. Training YOLOv7 model
3. Testing YOLOv7 model

At the end of this second sprint we obtained a YOLOv7 model that can perform semantic segmentation on input image as well as classify the monument, which gave good accuracy.

5.2.3 Sprint 3

In sprint3 we deployed the trained models and developed working framework of the system. The sprint 3 consists of following tasks:

1. UI design
2. VM setup on Server
3. Deployment of YOLOV7 on Server
4. Frontend Development
5. Backend Development
6. Model Evaluation and Visualization
7. Deployment of MobileNet-SSD v2 in Mobile app

5.2.4 Sprint 4

In sprint4 we added labeled dataset to further improve models accuracy and in case of MobileNet-SSD v2 we also added false negative image in our dataset. Besides, training of model on increased dataset, frontend and UI design of app was improved. Further, database was updated with description of monuments and local services around the monuemnts.

5.3 Overall Phase Followed

The overall project has been completed in three main phase which are:

a. Planning Phase

b. Development Phase

c. Integration

5.3.1 Planning Phase

In planning phase, necessary works to be done and planning of overall project were discussed and the decision were documented for further procedures.
First the project was divided into four parts:

a. Monument Detection

b. Cropping Bounding Box Frame

c. Monument Segmentation

d. Monument Recognition

These parts were then assigned to each project members who then studied about the respective parts in detail.

5.3.2 Development Phase

In this phase, the divided parts were well studied and developed. Then each of those developed parts were tested separately.

5.3.3 Integration

In this phase, the separately developed parts were integrated to form a system and integration testing was done.

Chapter 6

System Design and Architecture

6.1 USE CASE DIAGRAM

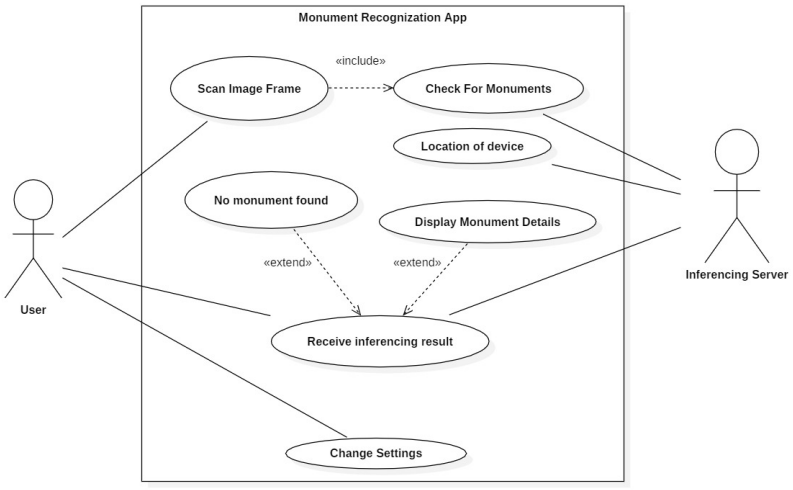


Figure 6.1: System Use Case Diagram

User acts as a primary actor and has ability to scan images, change application settings, and receive inferencing result.After scanning of image, application must check for presence of monuments.If the monument is not detected then it displays no monument found.If the monument is detected then proper classification is done and details of monument is visible to user.

6.2 SYSTEM FLOWCHART

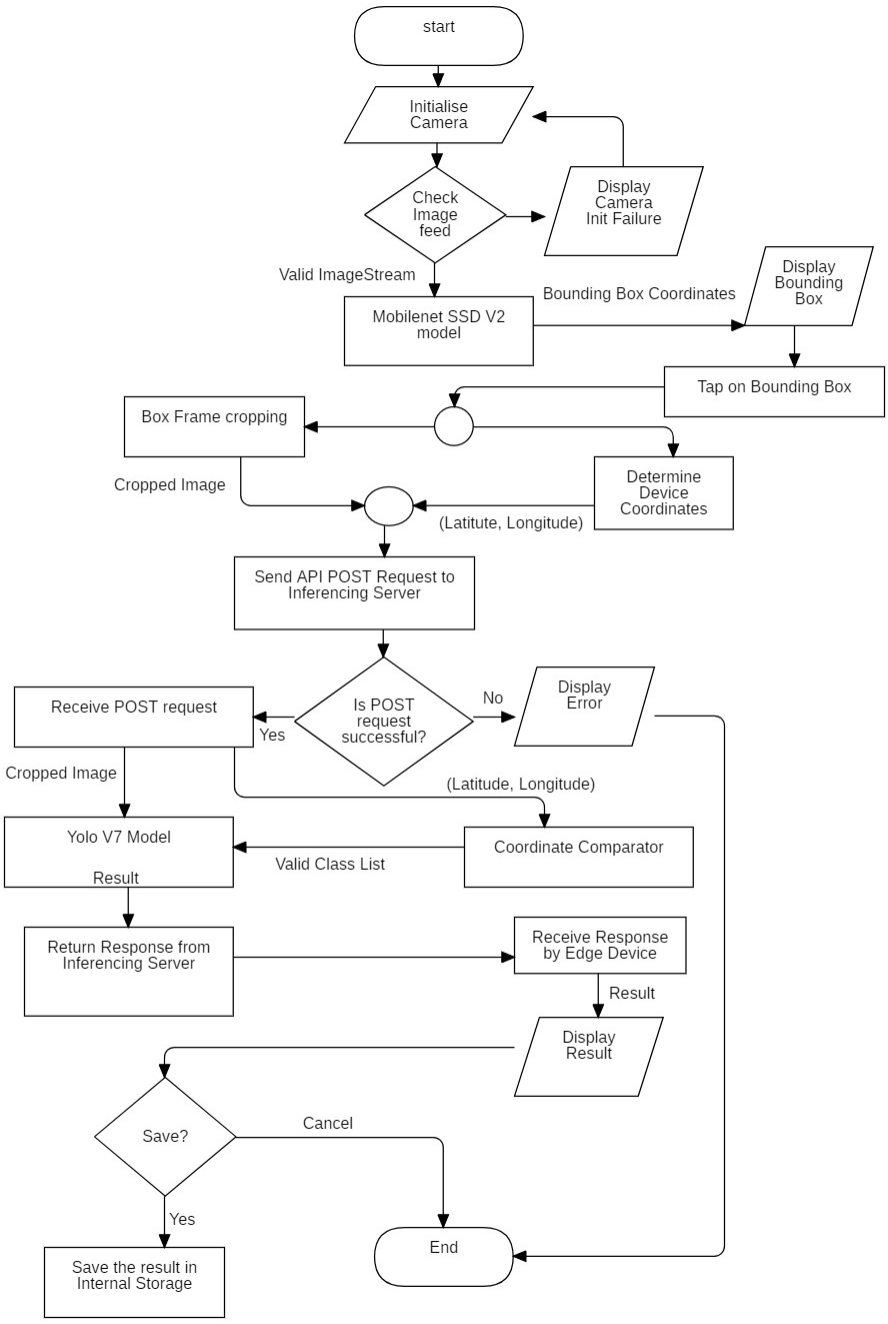


Figure 6.2: System Flowchart

6.3 SEQUENCE DIAGRAM

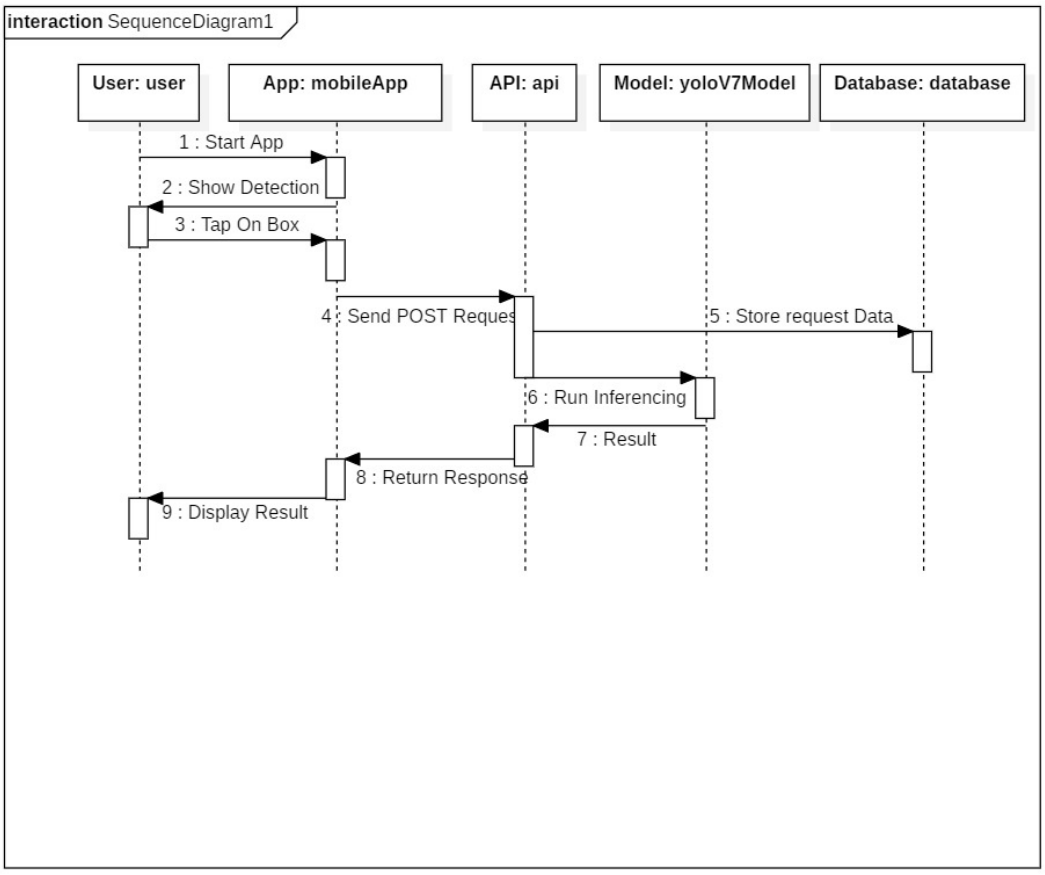


Figure 6.3: Sequence Diagram

On initialization of application camera is initiated.On captured image Mobilenet-ssdv2 checks the presence of monuments.If the monument is detected then it gives bounding box coordinates.Mobile application draws the bounding box, and user clicks on that box then cropped image and location of device is sent as API request to server.YOLO v7 predicts the image then response is sent back to the application.

6.4 Block Diagram and Description of Prepared System

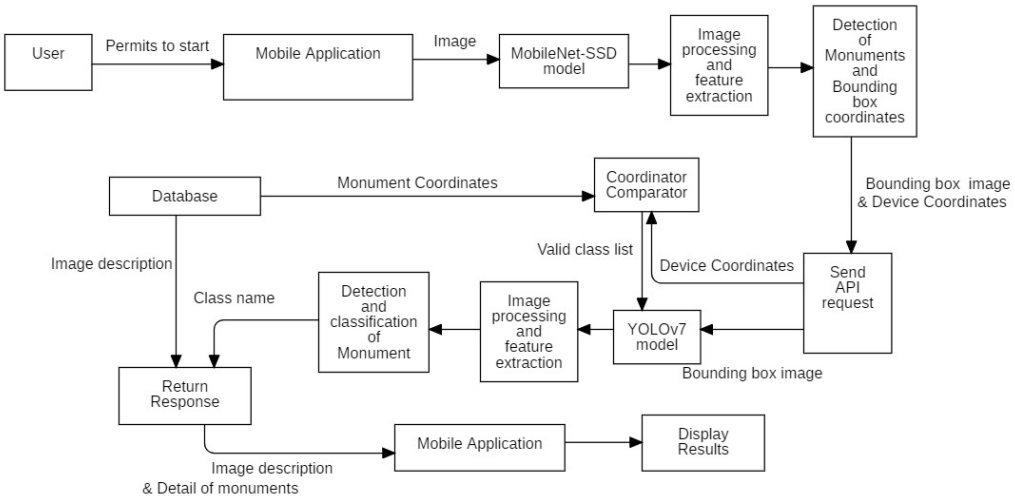


Figure 6.4: Block Diagram

6.4.1 Description of Prepared System

The camera is initialized through an Android application. MobileNet-SSD analyzes the camera-captured image for the presence of monuments and, if any are found, generates the bounding box coordinates.After the bounding box is formed using those coordinates, the image of the box part is cropped on tapping.The cropped image and devices geographical coordinates are submitted there via an API POST request.The YOLOv7 model and coordinator comparator are stored on the server.When using the Coordinator comparator, only classes that are present within a 200-meter radius of the user's location are displayed as valid class lists.Among legitimate class lists, YOLOv7 segments photos that have been cropped and predicts the class name.The server responds with a class name, which the application uses to provide a description of the monuments.

6.4.2 YOLO - You Only Look Once

6.4.2.1 What is YOLO?

YOLO stands for You Only Look Once. It's powerful, efficient and single pass object detection algorithm that uses features learned by a Deep Convolutional Network to detect an object. YOLO makes use of only Convolutional Layers. We implemented YOLO V7 in our project.

6.4.2.2 Feature of YOLO V7

Some of the features of YOLO V7 are given below:

- It has a higher resolution than the previous versions.
- It uses loss function called "Focal loss" which is helpful when there is imbalance class in datasets.
- It does not have any form of pooling.
- It is invariant to size of input image yet, we stick on constant input size.
- It can process images at a rate of 161 frames per second.
- The network downsample image by a factor called stride of network.

  for eg. If the stride of network is 32.

  input size = 640*640

  then, Output size = 20*20

  Output size is stride times smaller than input size.

6.4.2.3 How we implemented YOLO V7 ?

- Yolov7 model remained on the server.
- Input for YOLOv7 is image sent by MobileNet-SSD.
- Used for detection as well as classification.

6.4.2.4 YOLO-v7

The structure of YOLO-v7 is shown in fig 6.5. [8]It takes input image of 640x640 and then inputs it into the backbone network.There are several neural network architecture used in YOLO-v7 which makes it's speed faster along with increased accuracy. The popular neural network architectures used in YOLO-v7 are ELAN, CBS, MP-1, CAT, SPPCSPC [8]. All of them are capable of extracting features at different levels which increases the efficiency of model. SPPCSPC architecture is responsible for class prediction.It outputs three layers of feature maps of different sizes(i.e.80x80x255, 40x40x255,20x20x255) through the head network.

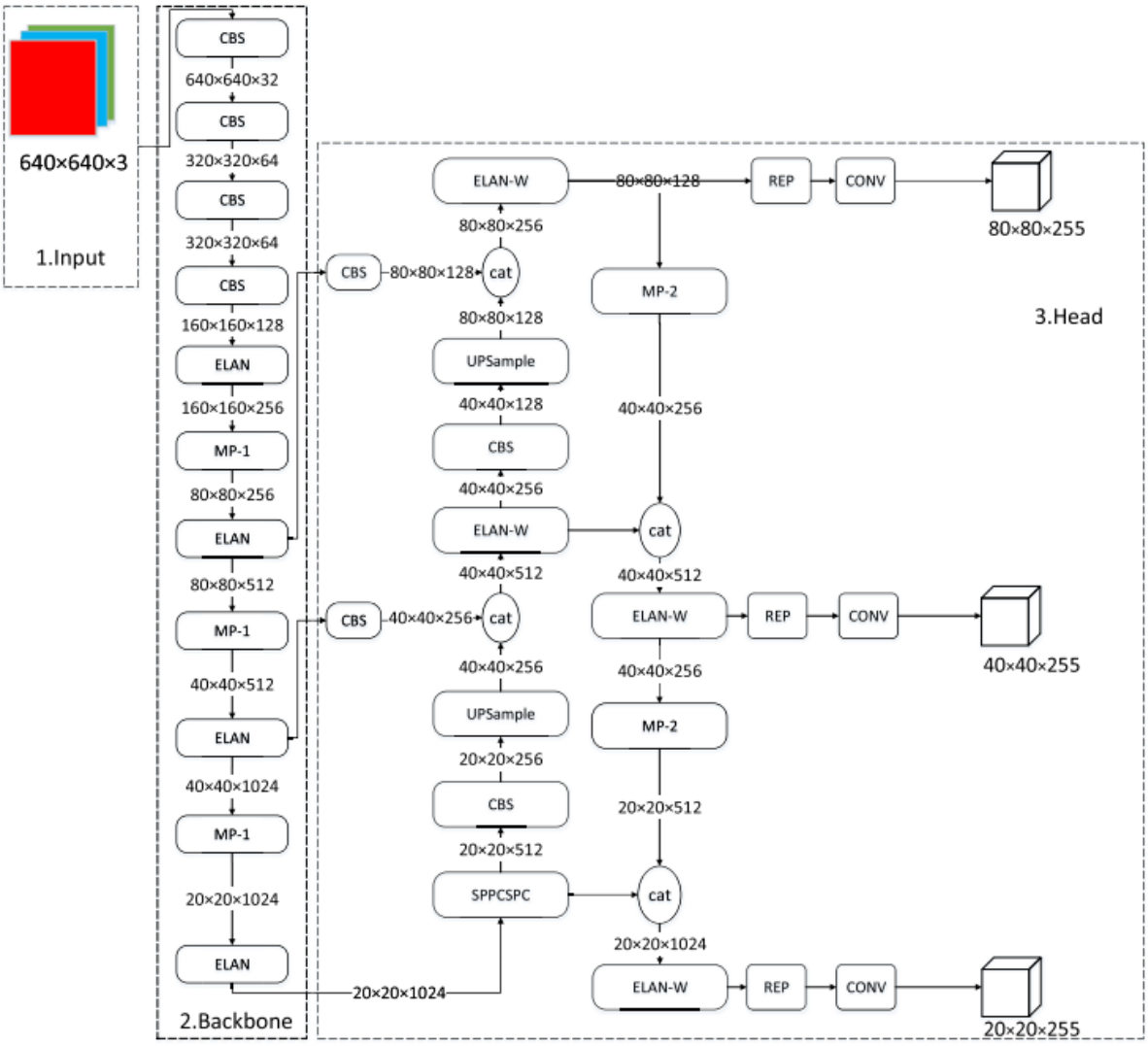


Figure 6.5: YOLO v7 Architecture
(source: [8])

6.4.2.5 Activation Function

One of the most widely used activation functions in deep learning, the SILU activation function, is utilized by YOLO-V7. The SILU activation function is defined as:

$$f(x) = \frac{x}{1 + e^{-x}} \tag{6.1}$$

where,

x = Input that can be scalar value or tensor of any shape.

It is also smooth and differentiable, which enables efficient backpropagation of gradients during training.The output of the function is also a tensor of the same shape as the input, where each element of the tensor is the result of applying the SILU activation function to the corresponding element of the input tensor.

6.4.2.6 Loss Design

YOLO-V7 model uses Focal Loss function as the backbone of loss.Focal loss function is used to penalize the model if there is imbalance classes which helps to minimize classification loss and also plays a vital role in Localization loss.The focal loss function introduces a modulating factor to the cross-entropy loss function, which reduces the weight assigned to well-classified examples and increases the weight assigned to misclassified examples. The Focal Loss is defined as:

$$FL(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{6.2}$$

where,

p_t is the predicted probability of class
$-\alpha_t$ is the class weighting factor
γ is the focusing parameter.
α_t can further be defined as:

$$\alpha_t = (1 - p_t)^{\beta} \tag{6.3}$$

where,

β is another hyperparameter that controls the weight assigned to the minority class.

The focusing parameter gamma is used to modulate the weight assigned to easy and hard examples. When gamma is set to 0, the focal loss function reduces to the standard cross-entropy loss function.

To use the focal loss for localization loss, the smooth L1 loss is multiplied by the focal loss weights for each positive example, as follows:

$$L_{loc}(t_i, v_i) = \sum_{i \in Pos} \sum_{j \in x,y,w,h} FL(p_i) smooth_{L1}(t_i^j - v_i^j) \tag{6.4}$$

where,

p_i is the predicted probability of object i.
L_{loc} is the localization loss function.

t_i is the ground-truth localization coordinates for the i^{th} anchor box.
v_i is the predicted localization coordinates for the i^{th} anchor box.
Pos is the set of anchor boxes for which the predicted class probabilities p_i are positive (i.e., for which the object is present).
$FL(p_i)$ is the focal loss function applied to the predicted class probability p_i.
$smooth_{L1}(t_i^j - v_i^j)$ is the smooth L1 loss function applied to the difference between the ground-truth and predicted localization coordinates for the j^{th} dimension of the bounding box.

By using the focal loss in combination with the smooth L1 loss, the localization loss can be more effective at training the object detection model to accurately detect and localize rare and difficult objects in the image.

6.4.3 MobileNet-SSD v2

6.4.3.1 What is 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.

6.4.3.2 Feature of MobileNet-SSD v2

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

- It has Depthwise Seperable 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 input image.
- Average pooling and Max pooling are used in MobileNet v2 and SSD architecture respectively.
- It can process video frames at average of 19 frames per second.

6.4.3.3 Benefits of MobileNet-SSD v2

- The model is light weight with high accuracy and fast inferencing making it suitable for real-time operation.
- It can be fine-tuned on custom dataset for specific application.

6.4.3.4 How we implemented Mobilenet-SSD V2 ?

- MobileNet-SSD model remained in Android application.
- Input is image captured by device camera.
- Detects presence of monuments and gives bounding box coordinates.

6.4.3.5 MobileNet-SSD v2 Design

MobilneNet-SSD v2 shown in figure 6.6 consists of MobileNet v2 as base architecture and SSD head for detection. Mobilenet v2 consists of a standard inverse residual modules, each inverse residual module contains a 1x1 convolutional layer, a 3x3 depth-wise separable convolutional layer, BN and Relu6. It provides 38x38 output feature map which is fused by FPN to improve performance of detection network.

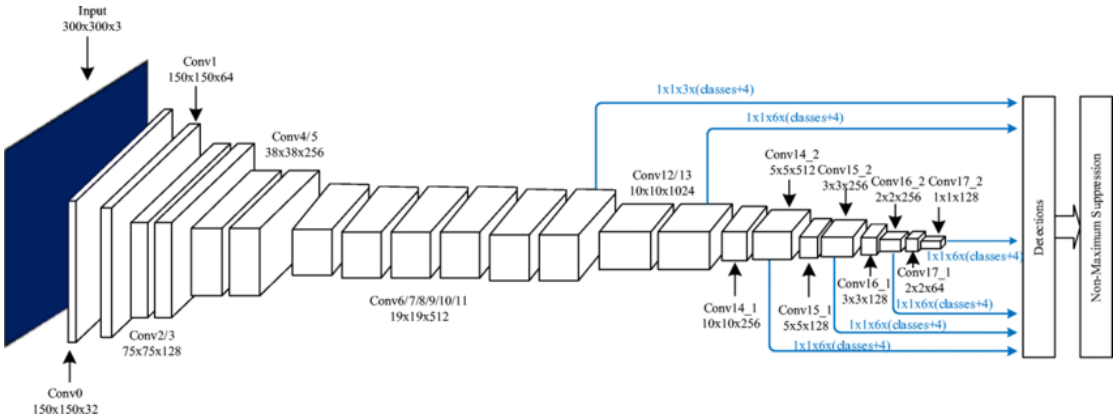


Figure 6.6: MobileNet-SSD v2 Architecture
(source: [9])

6.4.3.6 Activation Function

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

$$ReLU(x) = \begin{cases} x & \text{if} x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.5}$$

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

6.4.3.7 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 [10] of the loss function used in MobileNet-SSD v2 can be written as follows:

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g)) \tag{6.6}$$

where,

N represents the number of matched default boxes
x is the input image
c is the predicted class scores
l represents 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:

$$L_{loc}(x, l, g) = \sum_{i \in positive}^{N} \sum_{m \in \{cx,cy,w,h\}} {x_{ij}}^k smooth_{L1}(l_i^m - {(g^{'})_j}^m) \tag{6.7}$$

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:

$$smooth_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \tag{6.8}$$

$${(g^{'})_j}^m = \begin{cases} ({g_j}^{cx} - {d_i}^{cx})/{d_i}^w, m = cx, \\ ({g_j}^{cx} - {d_i}^{cy})/{d_i}^h, m = cy, \\ \log({g_j}^w/{d_i}^w), m = w, \\ \log({g_j}^h/{d_i}^h), m = h. \end{cases} \tag{6.9}$$

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

$$L_{conf}(x, c) = -\sum_{i \in Pos}^{N} {x_{ij}}^P log({c_i^{'}}^P) - \sum_{i \in Neg} log({c_i^{'}}^0) \tag{6.10}$$

where,

c is the true class label for the object in the anchor box
${c_i^{'}}^P$ is the predicted objectness score for anchor box

$${c_i^{'}}^P = \frac{exp({c_i}^P)}{\sum_P exp({c_i}^P)} \tag{6.11}$$

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.

6.4.4 Intersection Over Union

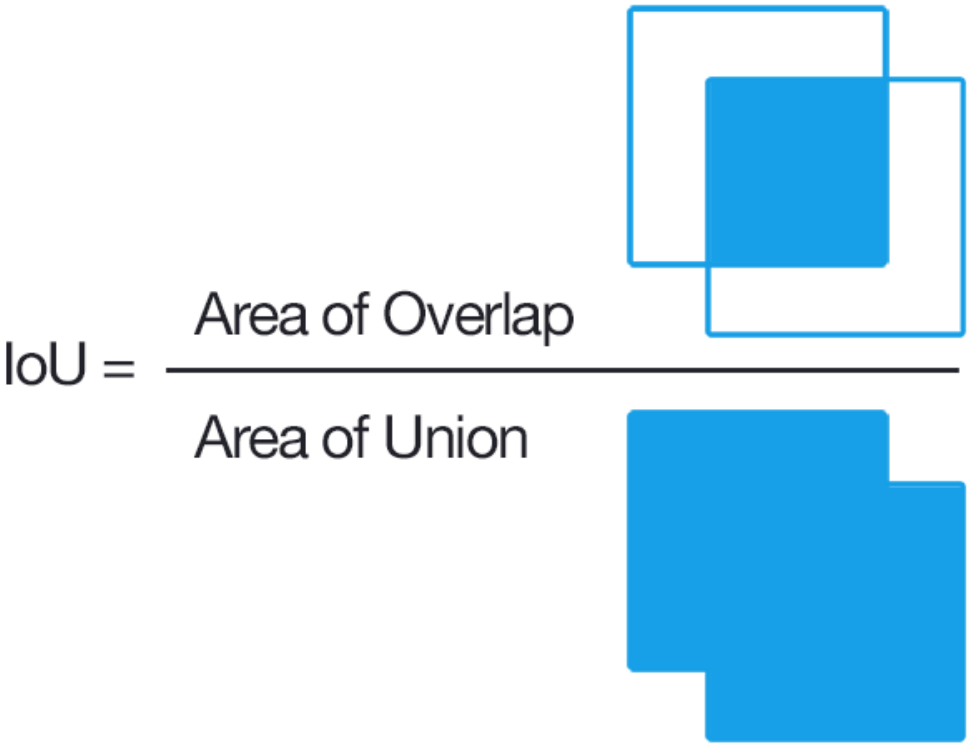


Figure 6.7: Intersection Over Union

Intersection over union(IOU) is a measure of overlap between two bounding box. 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 rectangular box. IOU is known to be a good metric for measuring overlap between two bounding boxes or masks. Following pictures shows what is goodness measure of IOU

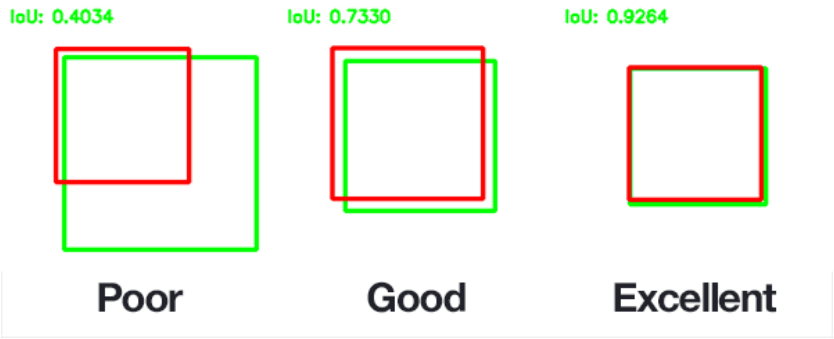


Figure 6.8: Goodness measure of IOU

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 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. [11]

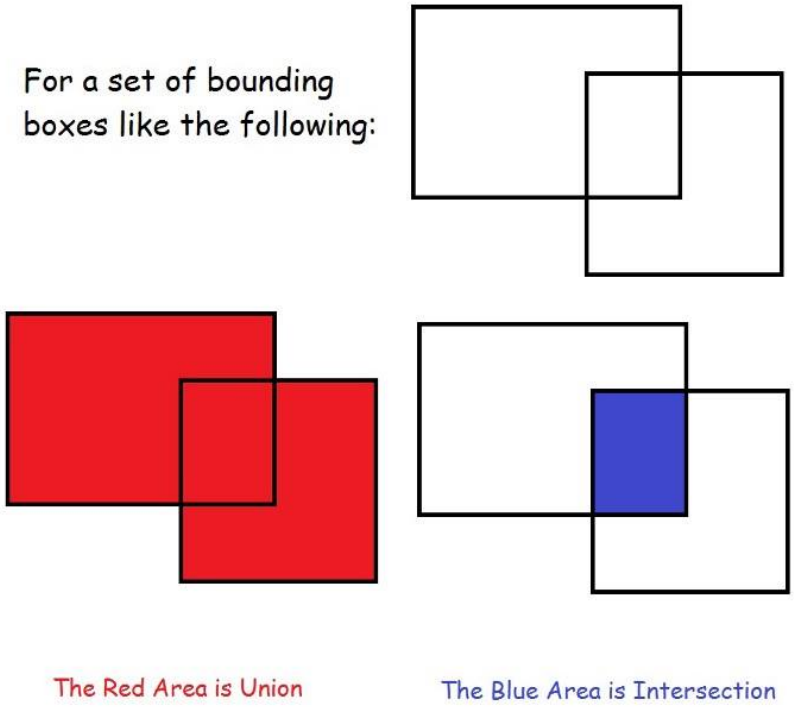


Figure 6.9: Bounding Box Union and Intersection

$$IOU(Box1, Box2) = \frac{Intersection_Size(Box1, Box2)}{Union_Size(Box1, Box2)} \tag{6.12}$$

6.4.5 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 is 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 supression.Non-max suppression cleans up the multiple detection and end with just one detection per object. For this it chooses the bounding box with highest probability and suppressed 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. [12]
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 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 which have high IOU than threshold.

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

Figure 6.10: Result before and after Non-max Suppression

6.4.6 Image Segmentation

Image segmentation is a computer vision technique that involves partitioning a digital image into multiple image segments, also known as image regions or image objects (sets of pixels).The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image. Each of the pixels in a region are similar with respect to some characteristic or computed property, such as color, intensity, or texture.

6.4.6.1 Semantic Segmentation

Semantic segmentation is one of the techniques to understand an image at pixel level. Specifically, it attempts to partition the image into semantically meaningful parts, and to classify each part into one of the pre-defined classes [13].

6.4.6.2 Instance Segmentation

Instance segmentation is a computer vision task that involves detecting objects in an image and precisely segmenting each object instance in the image with a unique mask.

The goal of instance segmentation is to provide a pixel-level understanding of the objects in an image. Instance segmentation algorithms typically use deep neural networks, which combine object detection and segmentation to predict bounding boxes and masks for each object instance in the image.

(a) Semantic Segmentation (b) Instance Segmentation

6.4.7 Dataset

6.4.7.1 Dataset Collection

We needed to train two models using datasets. The two models include the monument detection and monument recognition. We collected monument image dataset for model training. For monument detection using MobileNet-SSD V2 and monument recognition using YOLOv7, we collected image dataset using our mobile phones 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 a 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 accuracy of our detection mdoel.

6.4.7.2 Data Preprocessing

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 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 6.1: Data for Monument Detection and Recognition using MobileNet-SSD v2 and YOLOv7 respectively.

The labeled dataset data was split into 80% train and 20% test data and was resized to 320x320 for training MobileNet-SSD v2 model and 640x640 for training YOLOv7 model using the features of Roboflow web app.

6.4.7.3 Data Augmentation

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

1. Brightness
   The brightness of 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 a original image of dataset as shown in figure A.3.1.

2. Rotation
   Similary the annotated dataset was randomly rotated between the range of

20° clockwise and 20° anti-clockwise to generate 3 additional image data from a original image of dataset as shown in figure A.3.2.

3. 90° Rotate
   It includes 90° clockwise and anti-clockwise rotate on annotated training dataset as shown in figure A.3.3.

6.4.8 Model Evaluation

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

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

Figure 6.12: 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.

6.4.8.1 Precision:

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

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{Total\ Predicated\ Positive} \tag{6.13}$$

6.4.8.2 Recall:

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

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{Total\ Actual\ Positive} \tag{6.14}$$

6.4.8.3 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 of the 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.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{Number\ of\ correctly\ classified\ samples}{Total\ number\ of\ samples} \tag{6.15}$$

6.4.8.4 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.

$$F1Score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{6.16}$$

Chapter 7

Results and Discussion

7.1 Unit Tests

We developed separate units of our system and performed tests as these units were being developed. After inspecting results of these tests, we made necessary modifications and improvements in these units before integrating all the units. Different test cases for our different units are shown below:

7.1.1 MobileNet-SSD v2 Monument Detection Unit

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:

A. Open module

B. Insert Image name in which objects are to be detected

C. Run the module

S.N	Input	Expected Output	Actual Output	Result
1	Image of Nyatapola	Bounding box around monument in image	Appendix B.1 figure 4	Pass
2	Image of Vastala Devi Temple	Bounding box around monument in image	Appendix B.1 figure 5	Pass
3	Image of Bhairavnath	Bounding box around monument in image	Appendix B.1 figure 7	Pass
4	Image of Gopinath temple	Bounding box around monument in image	Appendix B.1 figure 7	Pass

Table 7.1: Unit test case of Detection Module

7.1.2 YOLOv7 Monument Recognition Unit

Test Scenario ID	Recognition Module
Test Case Description	Check recognition module with image as input
Pre-conditions	Valid Image

Test execution steps:

A. Open module

B. Insert Image name in which objects are to be detected

C. Run the module

1	Image of Nyatapola	segmentation with recognized monument name and confidence score.	Appendix B.2 figure 8a	Pass
2	Image of Vastala Devi Temple	segmentation with recognized monument name and confidence score. in image	Appendix B.2 figure 8b	Pass
3	Image of Bhairavnath	segmentation with recognized monument name and confidence score.	Appendix B.2 figure 8d	Pass
4	Image of Gopinath temple	segmentation with recognized monument name and confidence score.	Appendix B.2 figure 8d	Pass

Table 7.2: Unit test case of Recognition Module

7.2 Integration Testing

Test Scenario ID	Integrated Monument Recognition system
Test Case Description	Check the mobile app frontend integration with backend and test the camerafeed input
Pre-conditions	Function with all integrated models that takes camerfeed as input and returns monument class with description

Test execution steps:

A. Open mobile app

B. Obtain video feed of monument from mobile camera

C. Tap on screen after predicted bounding box appears

D. Click on more details button

E. check recognition output and description

F. Click on nearby services button check all services

S.N	**Action**	**Expected Output**	**Actual Output**	**Result**
1	open mobile app and input camera-feed of monument	Bounding box around monument	Appendix C.1 figure 10a	Pass
2	Tap on the screen	Inferencing server returns segmented image of identified monument	Appendix figure C.1 figure 10b	Pass
3	click on more detail buttons	Display description of recognized monuments	Appendix figure C.1 figure 10c	Pass
4	Check all the nearby services	Display list of geographically nearby services	Appendix figure C.2	Pass

Table 7.3: Integration test of Monument Recognition System

7.3 Model Evaluation Result

7.3.1 Evaluation of YOLOv7 Model

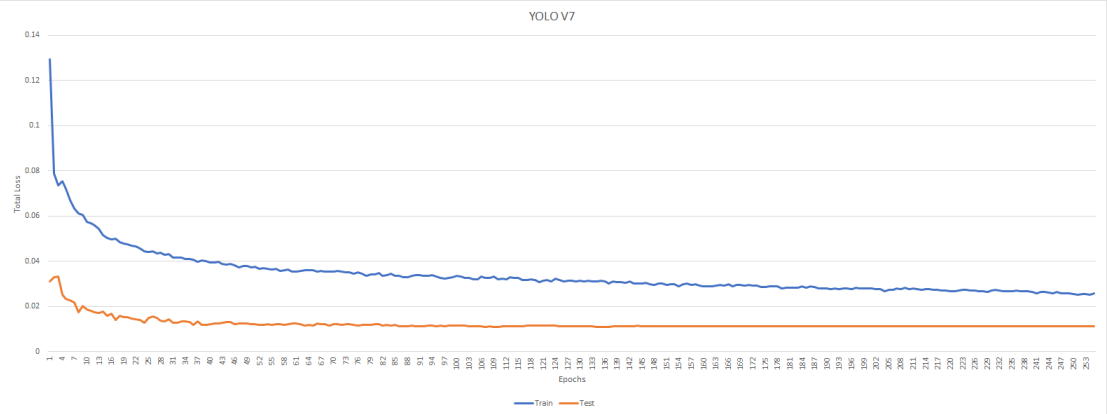


Figure 7.1: Train and Test Loss Graph of YOLOv7 model

In figure 7.1 we find the test dataset loss is much better than the training one, which reflects the test dataset was easier to predict than the training dataset.

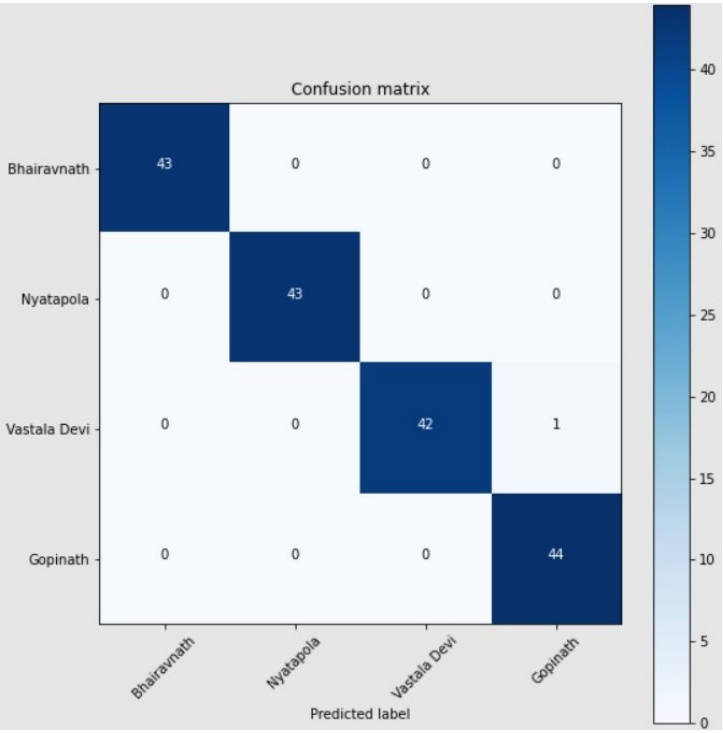


Figure 7.2: Confusion Matrix of YOLOV7 Model

SN	Monuments	Accuracy	Precision	Recall	F1 Score
1	Bhairavnath	100%	1.0	1.0	1.0
2	Nyatapola	100%	1.0	1.0	1.0
3	Vastala Devi	99.42%	1.0	0.98	0.99
4	Gopinath	99.42%	0.98	1.0	0.99

Table 7.4: Evaluation result of YOLOv7 module

Example of calculation of evaluation metrics from confusion matrix for Gopinath.

True Positive(TP)= 44
True Negative(TN)= (43+43+42) =128
False Positive(FP)= 1
False Negative(FN)= 0

$$1.Precision = \frac{TP}{TP + FP} = \frac{44}{44 + 1} = 0.98$$

$$2.Recall = \frac{TP}{TP + FN} = \frac{44}{44 + 0} = 1.0$$

$$3.Accuracy = \frac{TP + TN}{TP + TN + FP + FN} = \frac{44 + 128}{44 + 128 + 1 + 0} = 0.9942 = 99.42\%$$

$$4.F1Score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * 0.98 * 1.0}{0.98 + 1} = 0.99$$

From the test dataset evaluation, accuracy of YOLOv7 model is calculated as 99.42%.

7.3.2 Evaluation of MobileNet-SSD V2 Model

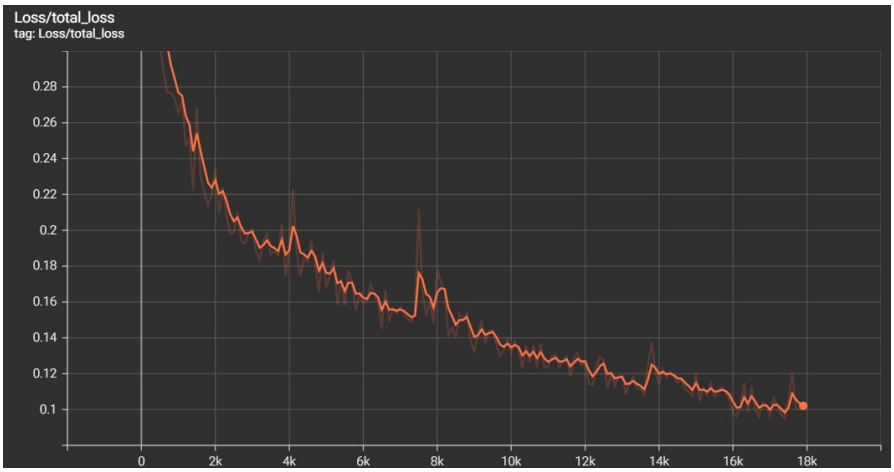


Figure 7.3: Training Loss Graph of MobileNet-SSD v2 model

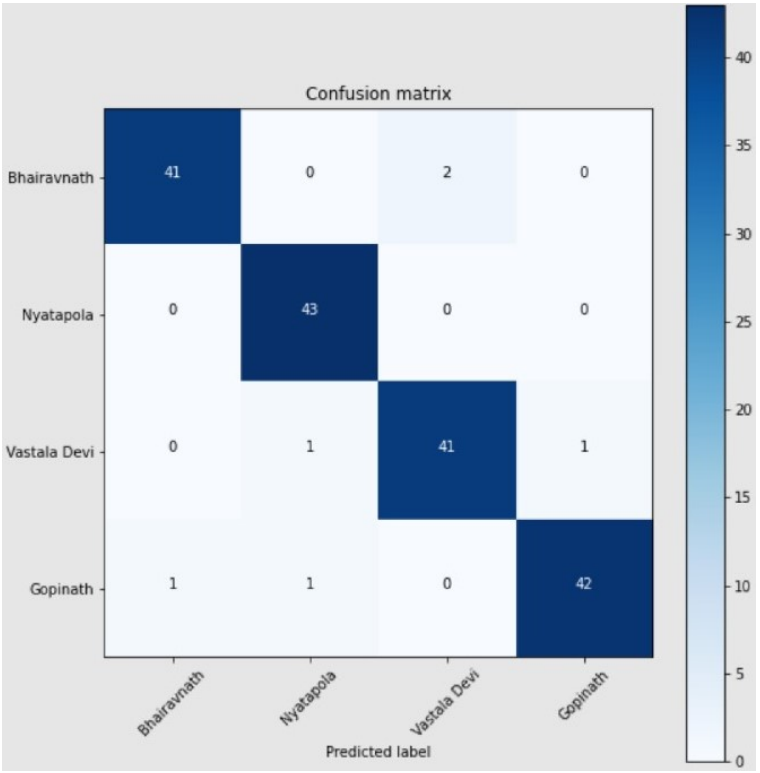


Figure 7.4: Confusion Matrix of MobileNet-SSD v2 Model

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 7.5: Evaluation result of MobileNet-SSD v2 module

Example of calculation of evaluation metrics from 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

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

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

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

$$4.F1Score = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * 0.95 * 0.95}{0.95 + 0.95} = 0.95$$

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

Chapter 8

Conclusion and Future Enhancements

8.1 Conclusion

To conclude, Monument Recognition System is completed successfully by making use of algorithms like YOLOv7 and MobileNet-SSD v2 and deployed as a mobile app which will help user to detect and recognize monuments and provide its description.

8.2 Future Enhancements

The Monument Recognition System currently works for recognizing limited number of monuments, mostly due to lack of time for preparing dataset. So, In near future this system can be enhanced and some future enhancements that we can implement in the project are given below:

- Training model on additional dataset of different monuments.
- Expanding training dataset on different lighting and weather conditions.
- Improving the detection and recognition accuracy.

Bibliography

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

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

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

[4] 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.

[5] 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.

[6] 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.

[7] A. Saini, T. Gupta, R. Kumar, A. K. Gupta, M. Panwar, and A. Mittal, "Image based indian monument recognition using convoluted neural networks," in *2017 International Conference on Big Data, IoT and Data Science (BID)*. IEEE, 2017, pp. 138–142.

[8] Y. Wang, H. Wang, and Z. Xin, "Efficient detection model of steel strip surface defects based on yolo-v7," *IEEE Access*, vol. 10, pp. 133 936–133 944, 2022.

[9] 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.

[10] 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.

[11] 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.

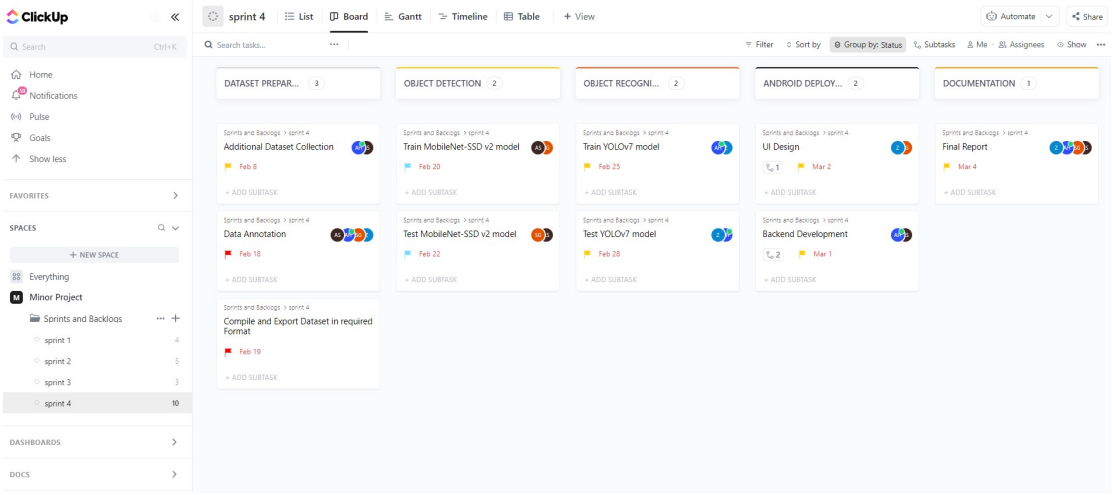
[12] S. K, "Non-maximum suppression (nms)," https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c, Oct. 2019.

[13] A. Arnab, S. Zheng, S. Jayasumana, B. Romera-Paredes, M. Larsson, A. Kirillov, B. Savchynskyy, C. Rother, F. Kahl, and P. H. Torr, "Conditional random fields meet deep neural networks for semantic segmentation: Combining probabilistic graphical models with deep learning for structured prediction," *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 37–52, 2018.

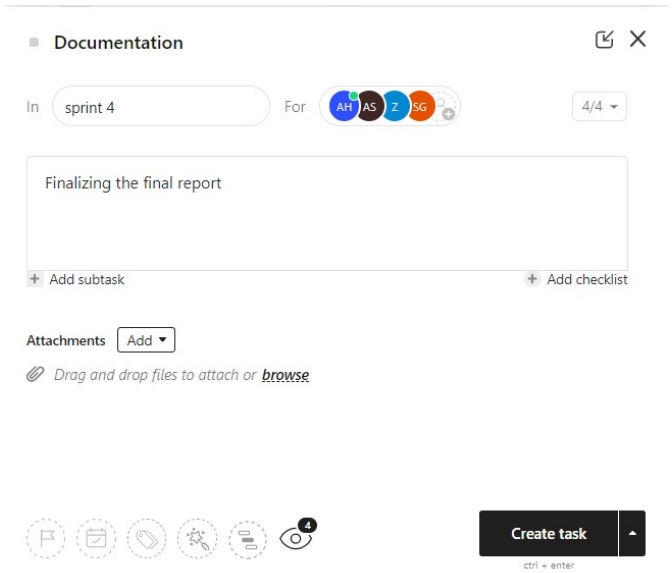
Appendix

A Snapshot

A.1 ClickUp

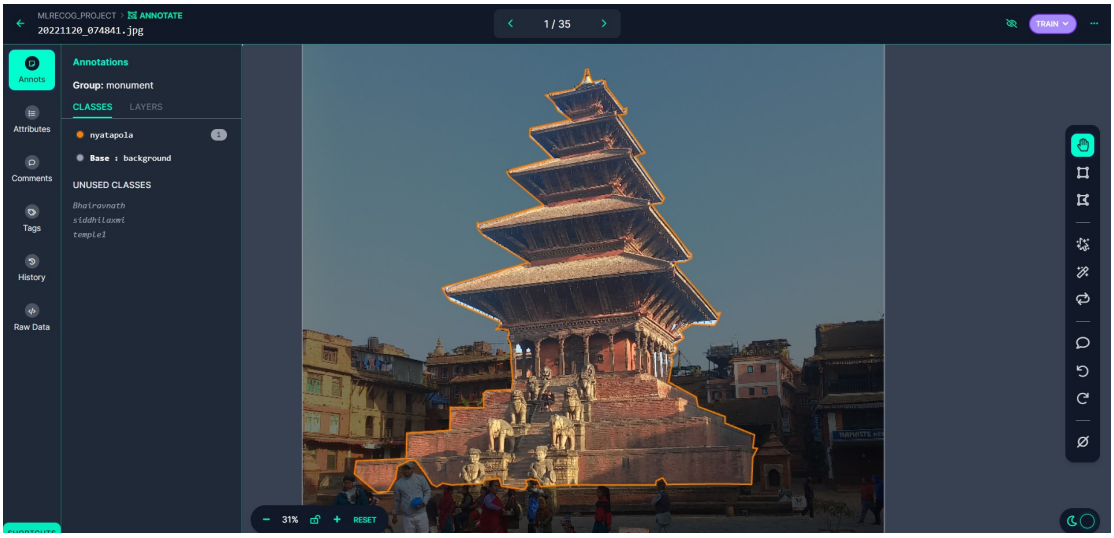


A.1.1 Create Task

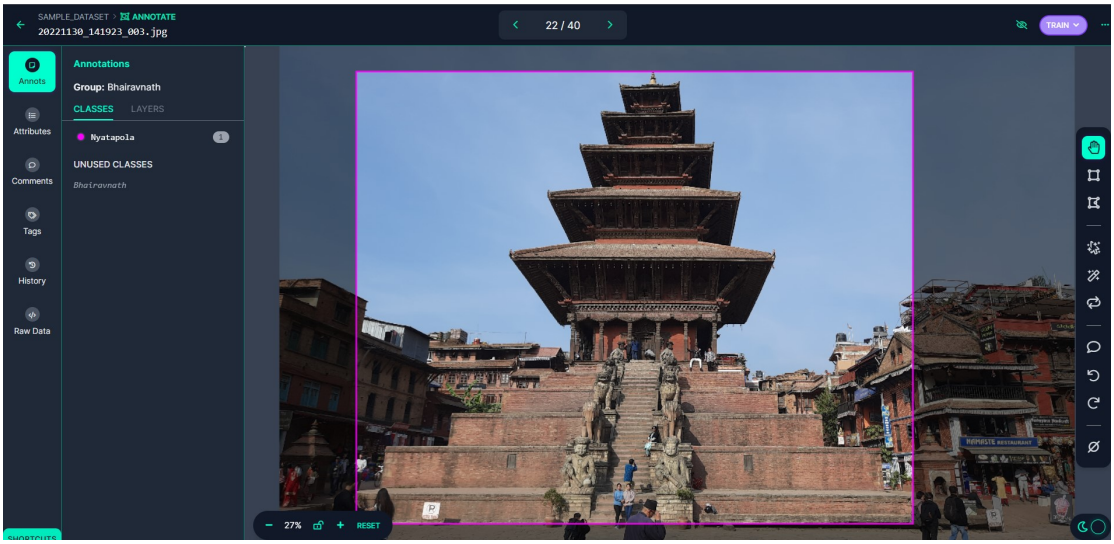


A.2 Data Annotation

A.2.1 Annotating data for Semantic Segmentation using Roboflow web app



A.2.2 Annotating data for object detection using Roboflow web app



A.3 Data Augmentation

A.3.1 Random alteration of brightness between the range of ± 15%

(a) brightness1

(b) brightness2

(c) brightness3

A.3.2 Random rotation between the range of ± 15%

(a) rotation1

(b) rotation2

(c) rotation3

A.3.3 90° rotation clockwise and anti-clockwise

(a) anti-clockwise rotation

(b) clockwise rotation

B Unit Test

B.1 Unit Test for Object Detection Module

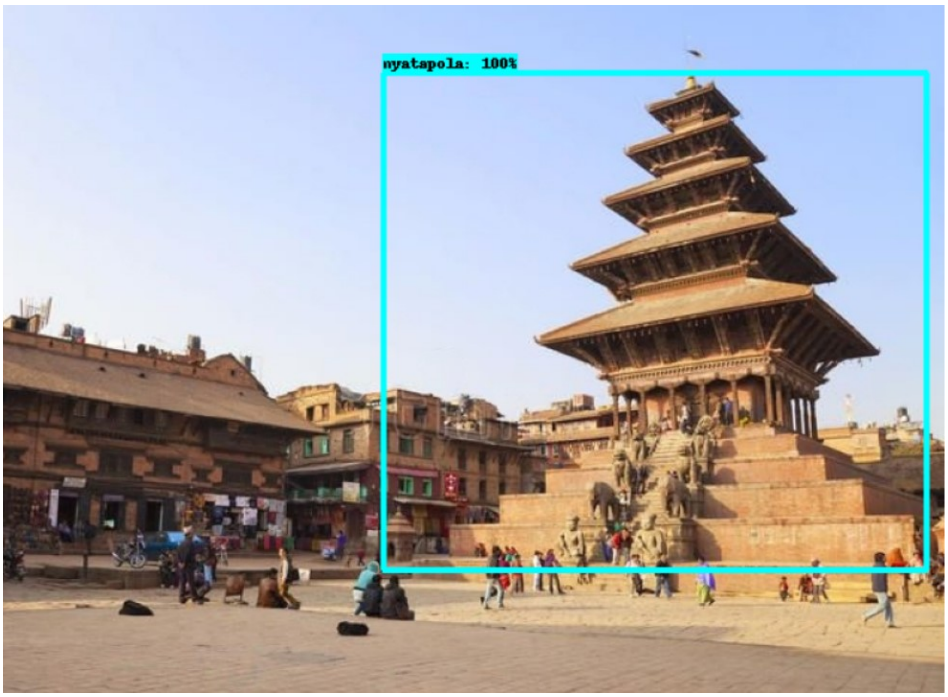


Figure 4: Unit test 1 for detection module

Figure 5: Unit test 2 for detection module

Figure 6: Unit test 3 for detection module

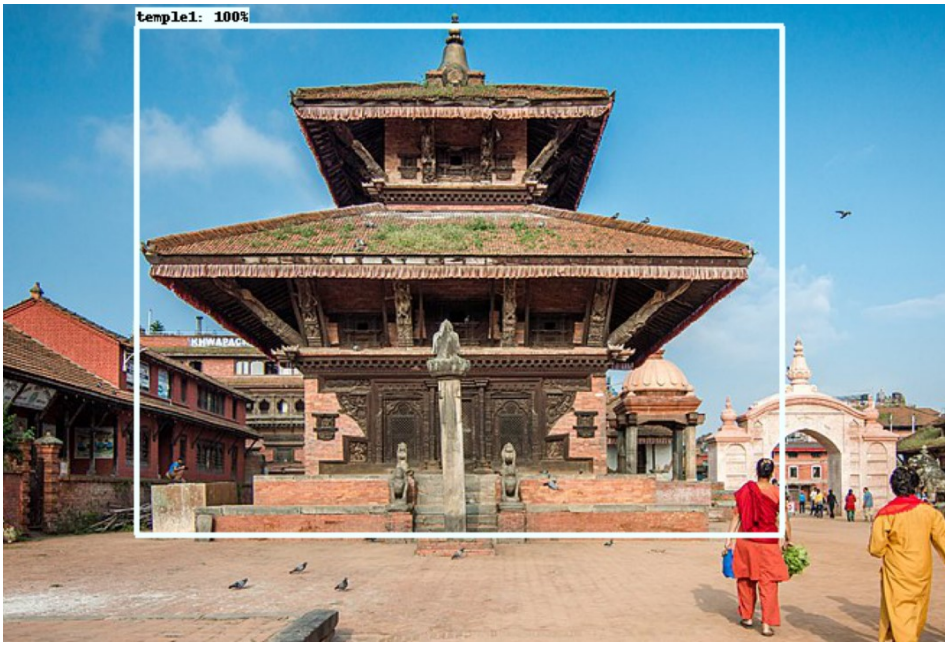
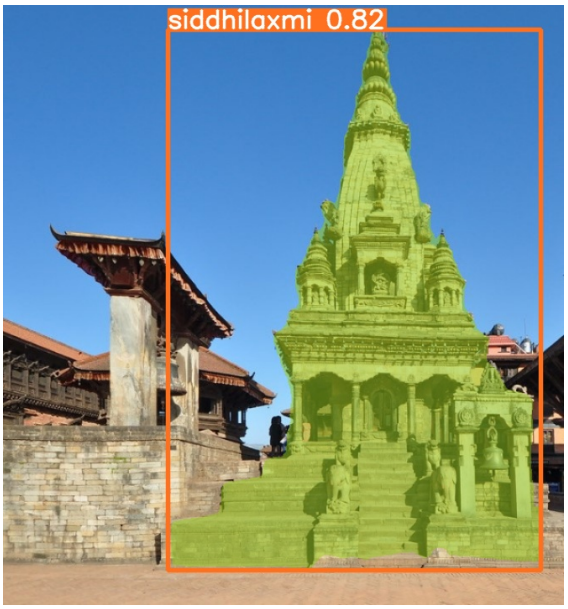


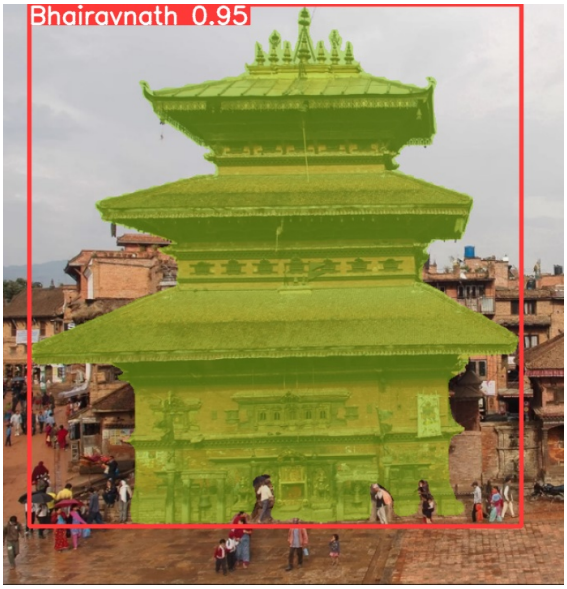
Figure 7: Unit test 4 for detection module

B.2 Unit Test for Monument Recognition Module

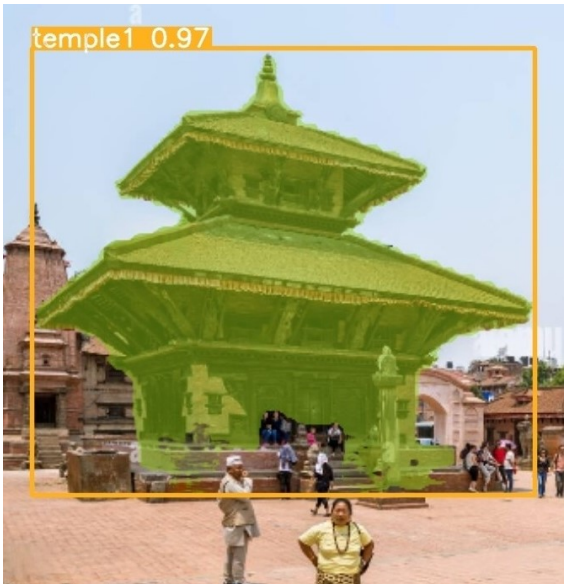
(a) Unit test 1 for recognition module



(b) Unit test 2 for recognition module



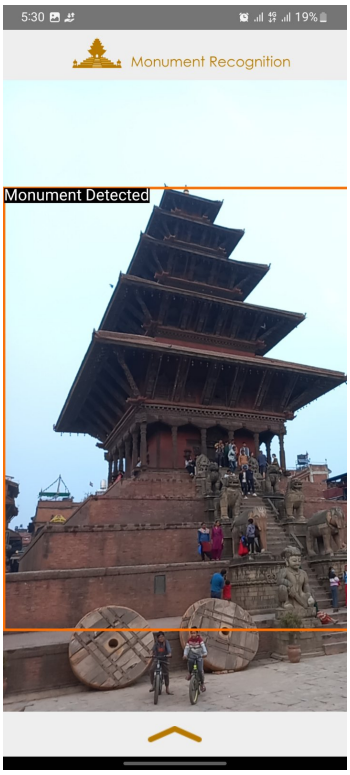
(c) Unit test 3 for recognition module



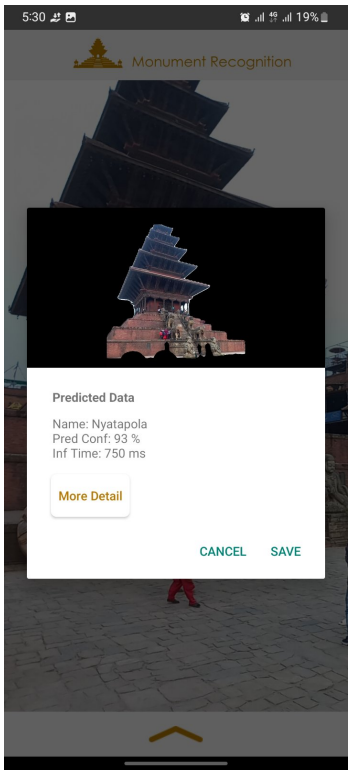
(d) Unit test 4 for recognition module

C Integration Test

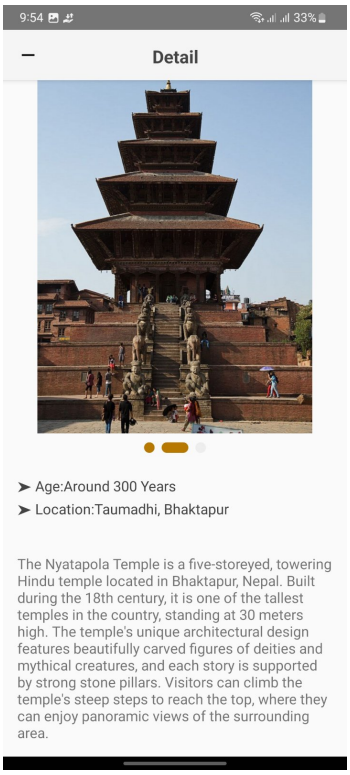
C.1 Monument Detection and Recognition Test Cases



(a) Monument Detection

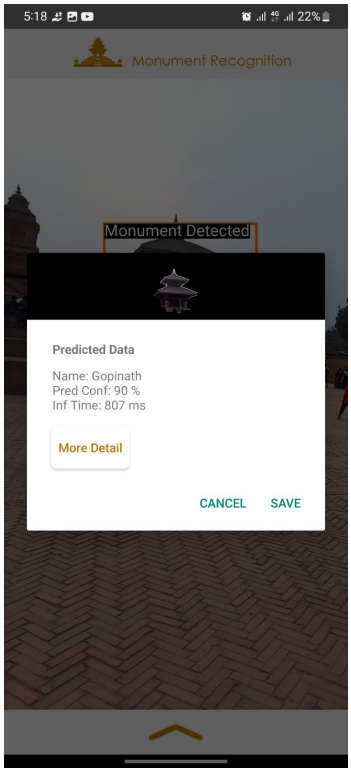


(b) Monument Recognition

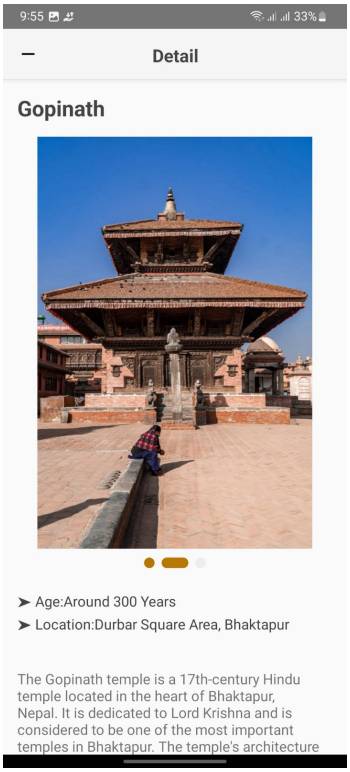


(c) Monument Description

(a) Monument Detection

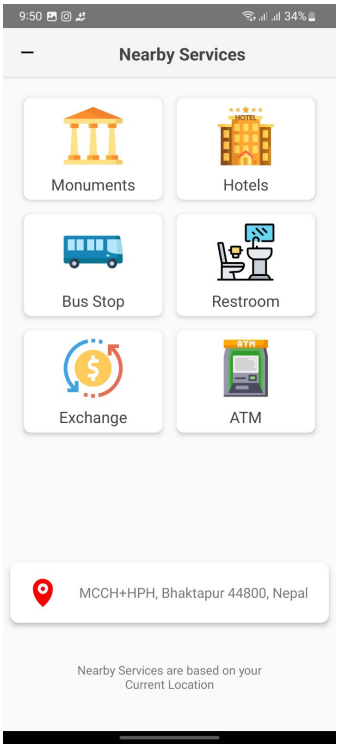


(b) Monument Recognition

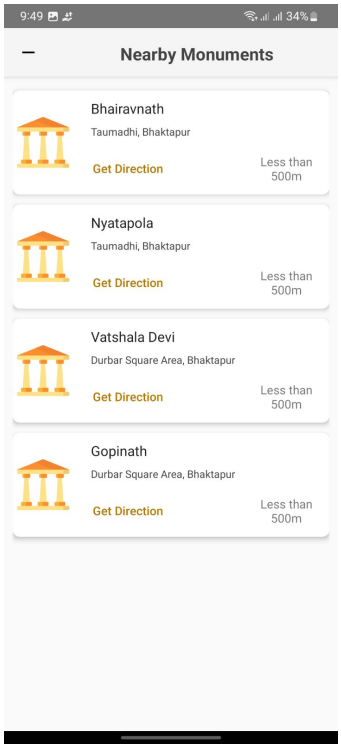


(c) Monument Description

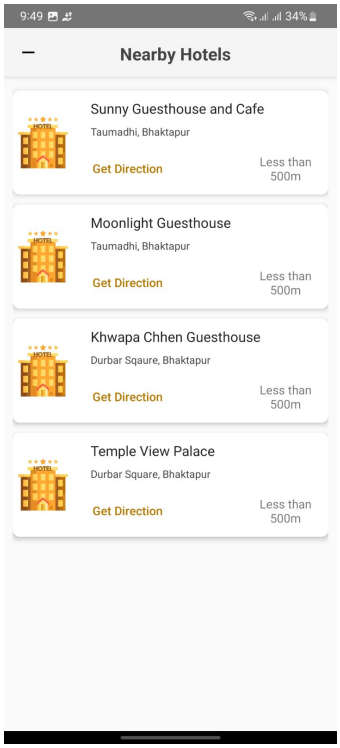
C.2 Nearby Services



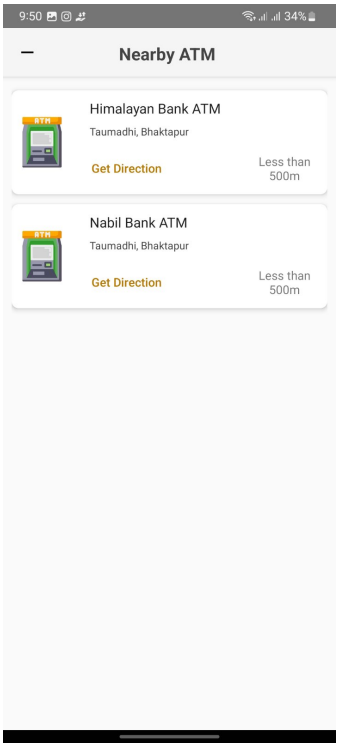
(a) Nearby service



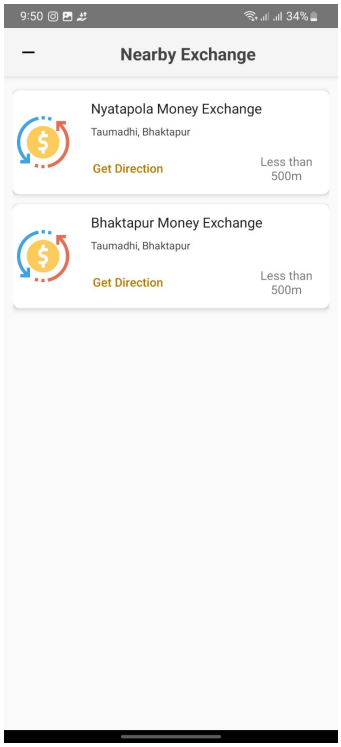
(b) Monuments



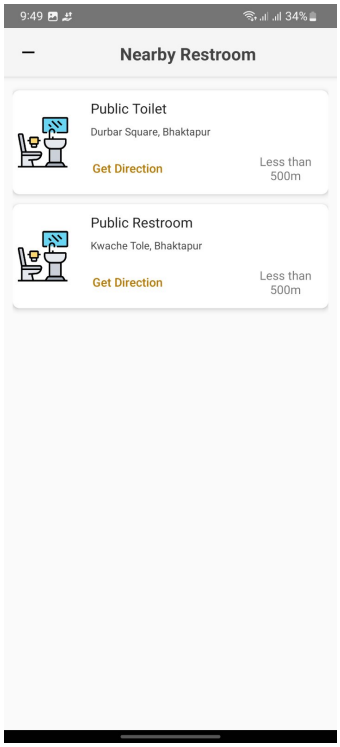
(c) Hotels



(d) ATM



(e) Exchange



(f) restroom