TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PASHCHIMANCHAL CAMPUS, POKHARA

Lamachaur, Pokhara-16

A final report on

"GANTAVYA: A LANDMARK RECOGNITION SYSTEM"

SUBMITTED BY:

Rupesh Ghimire	(PAS076BCT025)
Samir Gurung	(PAS076BCT032)
Sandesh G.C.	(PAS076BCT033)
Subek Sharma	(PAS076BCT043)

SUBMITTED TO:

Department of Electronics and Computer Engineering

March 2024

“GANTAVYA: A LANDMARK RECOGNITION SYSTEM”

Submitted by:

Rupesh Ghimire (PAS076BCT025)

Samir Gurung (PAS076BCT032)

Sandesh G.C. (PAS076BCT033)

Subek Sharma (PAS076BCT043)

Supervised by:

Urbara Bhandari

A MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE BACHELOR’S DEGREE IN COMPUTER ENGINEERING

Submitted to:

Department of Electronics and Computer Engineering

Pashchimanchal Campus

Lamachaur, Pokhara-16

March 2024

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PASHCHIMANCHAL CAMPUS

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING

The undersigned certify that they have read, and recommended to the Institute of Engineering for acceptance, a project entitled "GANTAVYA : A LANDMARK RECOGNITION SYSTEM" submitted by Rupesh Ghimire, Samir Gurung, Sandesh G.C., and Subek Sharma in partial fulfillment of the requirements for the Bachelor's degree in Computer Engineering.

Supervisor, Urbara Bhandari

Department of Electronics & Computer Engineering

External Examiner

Er. Khem Raj Koirala

Head of Department, Department of Electronics and Computer Engineering

DATE OF APPROVAL: March 2024

COPYRIGHT

ACKNOWLEDGEMENT

First, we are grateful to the Department of Electronics and Computer Engineering, Pashchimanchal Campus for providing us the opportunity to demonstrate our project work. We would also like to express gratitude to our project supervisor, Urbara Bhandari, for her continuous support during the development of this project. Also, we would like to mention the support from our parents who were always there for us, supporting us as we moved forward. Last but not the least, our friends, who provided us with encouragement and inspiration, are deeply appreciated.

Thank you for your support.

ABSTRACT

Landmarks are culturally, historically, and architecturally significant places that aid in navigation, foster community identity, and offer insight into a region's heritage and character. However, most people including tourists are unaware of the whereabouts of landmarks and their significance and a lot of time is wasted by people searching for details about the landmark. Landmark Recognition System aims to help people find their destination landmarks, related facts and information and guides them to their destination with its map functionality saving a lot of trouble. We opted to use object detection instead of classification, for which we studied various model architectures and chose to use YOLOv8, a state-of-the-art model at the time. To fine-tune the model, a custom dataset comprising landmarks images captured in person as well as scraped from the internet and labels for each image was created. The chosen model, YOLOv8 utilizes a hybrid backbone comprising CSPDarknet53 and PANet, employing cross-stage partial networks and path aggregation to efficiently detect objects with improved speed and accuracy. The backbone of this model acts as a feature extractor and sends the results of each kernel in the pyramid in the neck for further operations and finally, the head provides the final output of the model. A mobile application was developed by integrating the model with the Django backend that communicated to the React-Native frontend through RESTful API. The application takes an image as input from an authenticated user, predicts the landmark in the image and helps the user navigate to the place using maps. The results of the model developed, using AdamW optimizer and Cross Entropy Loss, when validated were found to be 97.7% box precision with 100% recall and the mAP scores were as 99.5% for mAP50 and 79.9% for mAP50-95 showing that the model developed is robust and accurate.

Keywords:YOLO, CSP, AdamW, Cross Entropy Loss,

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

AdamW	Adaptive Moment Estimation Weight Decay Fix
API	Application Programming Interface
GPU	Graphical Processing Unit
REST	Representational State Transfer
YOLO	You Only Look Once
SOTA	State Of The Art
CV	Computer Vision
COCO	Common Objects in Context
mAP	mean Average Precision
LAN	Local Area Network
CSP	Cross Stage Partial Connections
YAML	YAML Ain’t Markup Language
FNN	Feed-Forward Neural Network
SRC	Sparse Representational Classifier
RANSAC	RANdom SAmple Consensus
ORB	Oriented FAST and Rotated BRIEF
RCNN	Region Based Convolution Neural Network
ROI	Region Of Interest
ReLU	Rectified Linear Unit
CNN	Convolutional Neural Network
GLOPs	Giga Floating Point Operations per Second
DFL	Distribution Focal Loss

CHAPTER 1: INTRODUCTION

1.1 Background

Landmark Recognition is a quite challenging problem in the discipline of computer vision. There are various factors involved such as scale and resolution of an image, time of day, surroundings, foliage and so on. The landmark has to be correctly identified no matter the variables present in a photo. Computer Vision deals with such problems of visual identification of a subject.

A landmark is an object or feature of a landscape or town that is easily seen and recognized from a distance, especially one that enables someone to establish their location. Quite evident by its combination of words ‘land’ and ‘mark’, it is something that marks the location of a land. Landmarks have to be distinct and easily identifiable in certain spaces. For newcomers or tourists, exploring any new place can be a daunting task. With landmarks any location can be traversed easily and help with finding out the general vicinity they are in. With the help of computer vision assisted by machine learning it is possible to successfully identify the landmark given by a person.

A great amount of work has been done in the field of landmark recognition and classification. While popular landmarks which have acquired global popularity have been extensively researched upon and various models have been trained on them. But at a local level, landmarks which are not relatively popular have been left out. So, developing a robust, accurate and flexible Landmark Recognition Model where users can directly input photos of a random landmark can be considered as a challenging task due to the lack of proper training datasets on lesser-known landmarks.

Developing and training a Computer Vision model for landmark recognition involves several key steps. Firstly, sufficient data must be collected to train the model effectively. While many existing models scrape data from the internet, this approach may not suffice for lesser-known landmarks, as limited data may be available online. In such cases, custom datasets need to be prepared by shooting images of the landmarks on location and labelling them appropriately. This ensures that the model is trained on relevant and accurate data, enabling it to recognize the landmarks accurately.

In the context of developing a landmark recognition system for Pokhara, Nepal, the lack of proper datasets on local landmarks necessitates the creation of custom datasets. This involves capturing images of the landmarks in Pokhara and labelling them for training purposes. By creating custom datasets tailored to the specific landmarks of Pokhara, the model can be trained more effectively to recognize these landmarks with precision.

Once the landmark recognition model has been trained and deployed, it can be further enhanced by feeding it more datasets of other landmarks to expand its catalogue of recognizable landmarks. Additionally, the model can be developed into user-friendly apps that allow users to identify landmarks in real-time images. This application of Computer Vision technology not only enhances the user experience by providing instant access to information about landmarks but also contributes to the advancement of AI-driven solutions for real-world challenges.

1.2 Problem Statement

Tourists in various places visit numerous places daily and click several photos. During their visit, they might want to know about certain landmarks. Or they might encounter a certain landmark that they are curious about. For this, a landmark recognition system can be used. But problems arise in the various factors of the photo taken. No photo is a 100% match to another photo. They all have different resolutions, scales, illumination, angles and other various image descriptors of the same landmark. For the model to recognize such different image parameters is a major task. The photo taken by the user can be quite different to the dataset of photos the model has been trained on. However, the model should most definitely provide the correct landmark classification regardless of the quality of the photo.

1.3 Objectives

The main objective of our project is: To provide a user-friendly interface for users to upload photos and leverage the power of advanced models to identify landmarks and receive the details regarding them.

1.4 Applications

The major application areas of our project are

I. To enhance public accessibility and awareness of landmarks in Pokhara through the development of a user-friendly mobile application.

II. To empower travelers with accurate landmark identification for informed exploration of Nepal's

CHAPTER 2: LITERATURE REVIEW

2.1 Related Works

The area of landmark recognition is a field that hasn't been explored much. Some research studies in the field of landmark detection have been done using classifiers and feature-matching algorithms while very few have been further developed as a web-based application that uses static images to predict landmarks. Landmark recognition systems leverage computer vision, utilizing signs, perspectives, and patterns in images to identify landmarks. Applications span navigation, travel, augmented reality (AR), and information provision. Vision models utilizing CNN architectures like RANSAC [20], SPK with FNN[3], RCNN[16], and YOLO[5][18]. Popular tools and technologies in building these systems include Tensorflow [14], OpenCV [8], PyTorch [15], CNN[19] and Deep Residual Networks[17].

The use of a heavy model in edge devices like mobile phones poses the challenge of utilizing the computation power to perform well in real time. Keypoint detection, involving the identification and localization of interesting points in images, is essential. The studies done in [11] by S. Matsuzaki et al. on global localization in object-based map images and in [12] by S.N. Sinha on improved scene landmark detection for camera localization showed the use of extracting correspondences based on conceptual similarity using a Vision Language Model, leading to more accurate localization with fewer iterations compared to baseline methods.

Tao Chen et al. in [1] have presented a small-scale landmark recognition system using classification methods for mobile phones to aid in information retrieval. It synthesizes findings to provide insights into the landscape of mobile landmark recognition, highlighting prevalent methods and their efficacy in retrieving information based on captured landmarks.

The study was done by Yan-Tao Zheng et al. under Google Inc. in [2], leveraging over 20 million photos of over 5000 landmarks from 144 countries collected by mining millions of GPS-tagged photos and online tour guide web pages. The project employs a methodology that combines GPS-tagged photos and travel guide articles to compile a diverse dataset of landmarks, utilizing unsupervised learning techniques for visual model construction. The findings underscore the project's success in overcoming challenges related to data collection

and system efficiency, showcasing its potential for widespread applications in various domains.

Researchers propose a collaborative framework for landmark recognition in [3], integrating SPK-Bag of Words, FNN and SRC to enhance accuracy and speed. This involves extracting features with SPK-BoW, training FNN using extreme learning machines (ELM), and refining accuracy with SRC. Testing on NTU campus images yields improved recognition rates, with SRC achieving up to 81.43% accuracy, emphasizing experimentation with various extreme machine learning algorithms for system refinement.

In study [4] done by researchers at Stanford University employs GPS data and advanced image processing techniques, such as "Bag-of-Visual Words" and histogram matching, to accurately identify landmarks within the Stanford University area. Key algorithms like RANSAC and ORB are utilized for image comparison and keypoint detection, contributing to Monulens' accuracy and real-time processing capabilities. Through rigorous testing, Monulens demonstrates optimal performance when images originate from the same device, with efforts to optimize processing time yielding significant improvements, showcasing its potential for enhancing landmark recognition systems' efficiency and effectiveness in real-world applications.

In extending Suenderhauf et al.'s work in [7] by Pilailuck Panphattarasap and Andrew Calway, introducing descriptors constructed from landmark features that encode spatial distributions within views, improving visual place recognition. Matching these descriptors enforces consistency of landmark relative positions between views, yielding significant performance enhancements. Experimentation on 10 image-pair datasets, each containing 200 urban locations with varying viewing conditions, showed an average precision of approximately 70% (at 100% recall) which outperformed previous methods, with whole image CNN features achieving 58% and the method achieving 50% precision.

The paper introduces a novel approach to landmark recognition, utilizing deep metric learning and curriculum learning techniques to efficiently handle a large number of landmarks. By training a deep convolutional neural network (CNN) with curriculum learning and modified Center loss, the method achieves high recognition accuracy while minimizing false positives. The proposed inference algorithm efficiently identifies landmarks in user photos, leveraging embeddings and centroid distances. Deployed at scale

within the mail.ru cloud application, the method demonstrates comparable performance to state-of-the-art approaches, offering speed and scalability for real-world applications in landmark recognition.

[10] presents a novel approach to place recognition in dynamic environments, incorporating dynamic object detection to enhance place representations. Results demonstrate a significant improvement in recognition accuracy, with the proposed approach outperforming the traditional Bags of Binary Words (BoBW) algorithm. Specifically, when incorporating dynamic object information, the recognition accuracy improves by 43.12% on images with more than 10% coverage by dynamic objects, reaching improvements of 55.75% and 62.22% on images with more than 20% and 30% dynamic object coverage, respectively. Additionally, the proposed method generates smaller databases, reducing storage space by up to 23.9% compared to traditional approaches, while also decreasing the time required for matching places by several milliseconds. Overall, the study underscores the efficacy of incorporating dynamic object information for improved place recognition accuracy and efficiency in dynamic environments.

CHAPTER 3: METHODOLOGY

For the Landmark Recognition System, we leveraged the power of Machine Learning for the detection of landmarks from the user's input image. For that, we had to go through a series of steps to train our machine-learning model to predict landmarks accurately. We then deployed the model using Django Backend and React Native Frontend to build a mobile application for our users. The communication between the frontend and backend takes place through RESTful API and for accessing all the functionalities within the application, the user should be logged in. This series of workflow covering image collection and preparation, machine learning model preparation and programming for the mobile application can be better understood from the figure below:

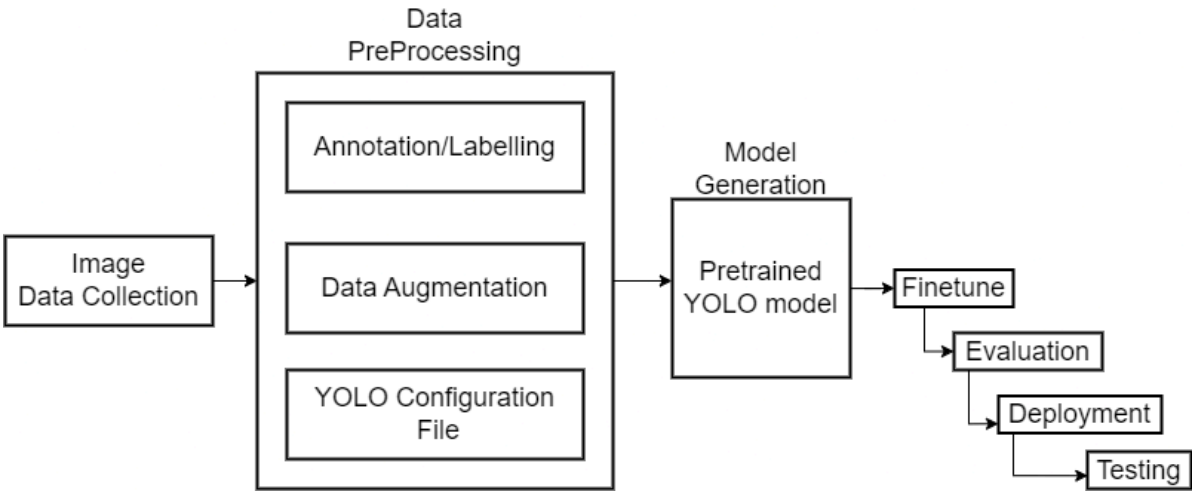

Figure 3.1: Model Workflow

3.1 Requirement Analysis and System Design

Requirement analysis and system design are pivotal stages within the methodology for developing a landmark recognition system, particularly focusing on user-system interaction and backend-frontend dynamics. Through meticulous requirement analysis, we discern the users' needs and system constraints, guiding the subsequent design phases. One crucial aspect is the creation of a detailed use case diagram, which portrays the various interactions between users and the system. This diagram delineates user actions, system responses, and external dependencies, providing a comprehensive overview of the system's functionality from a user's perspective.

3.1.1 Use Case Diagram

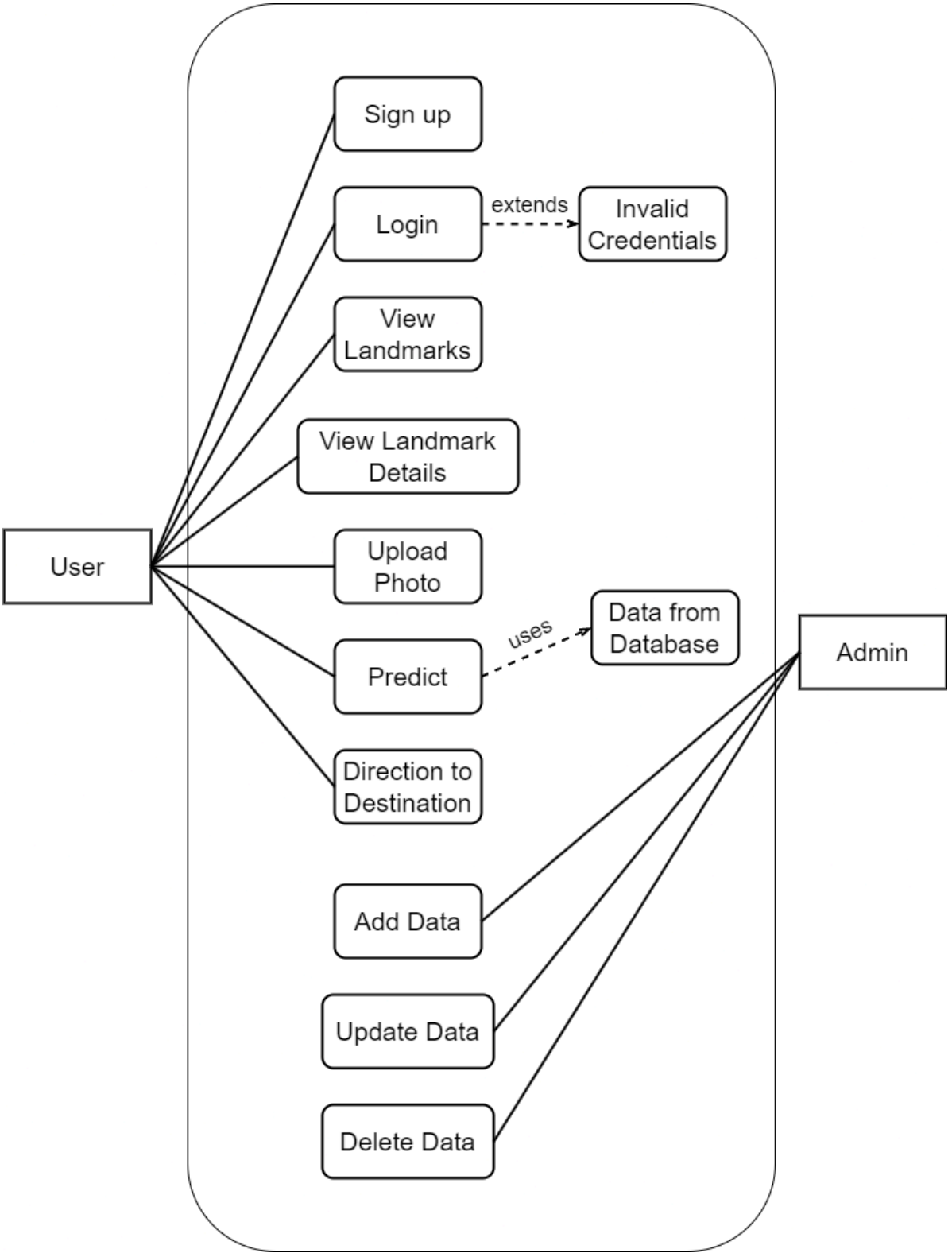

Figure 3.2: Use Case Diagram

3.1.2 User View Diagram

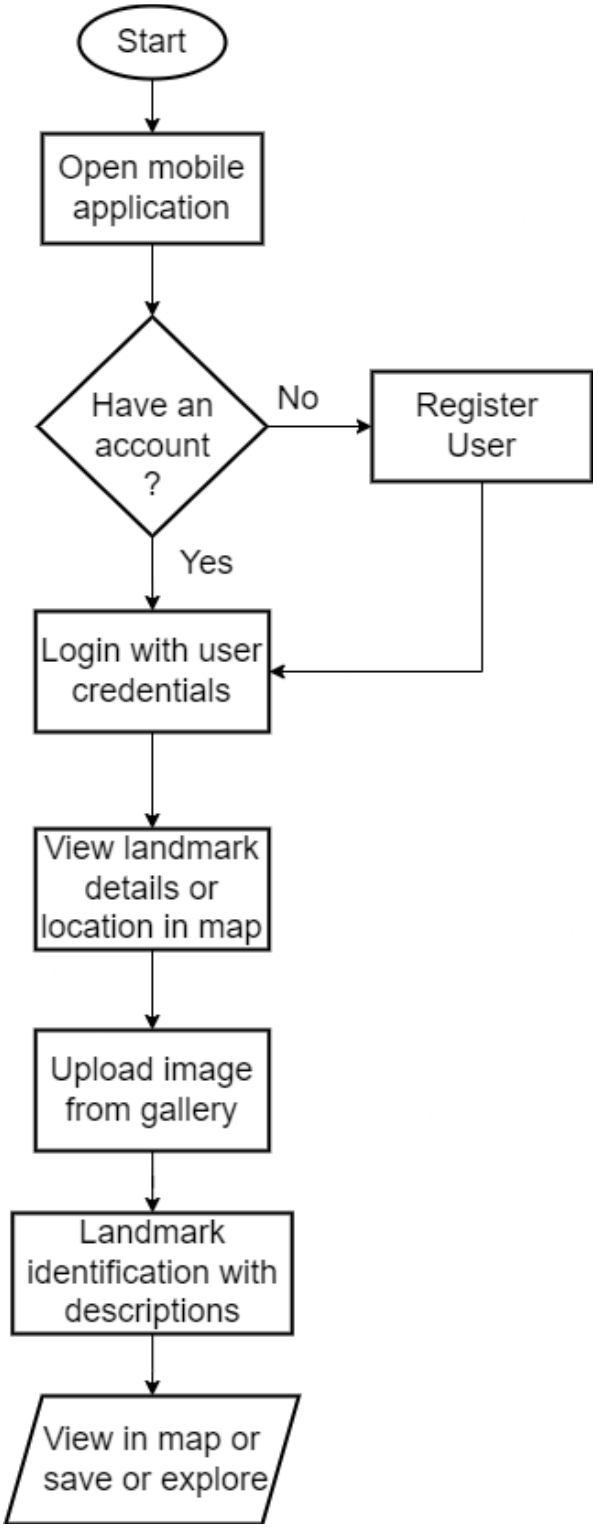

Figure 3.3: User View Diagram

3.1.3 Application Logic

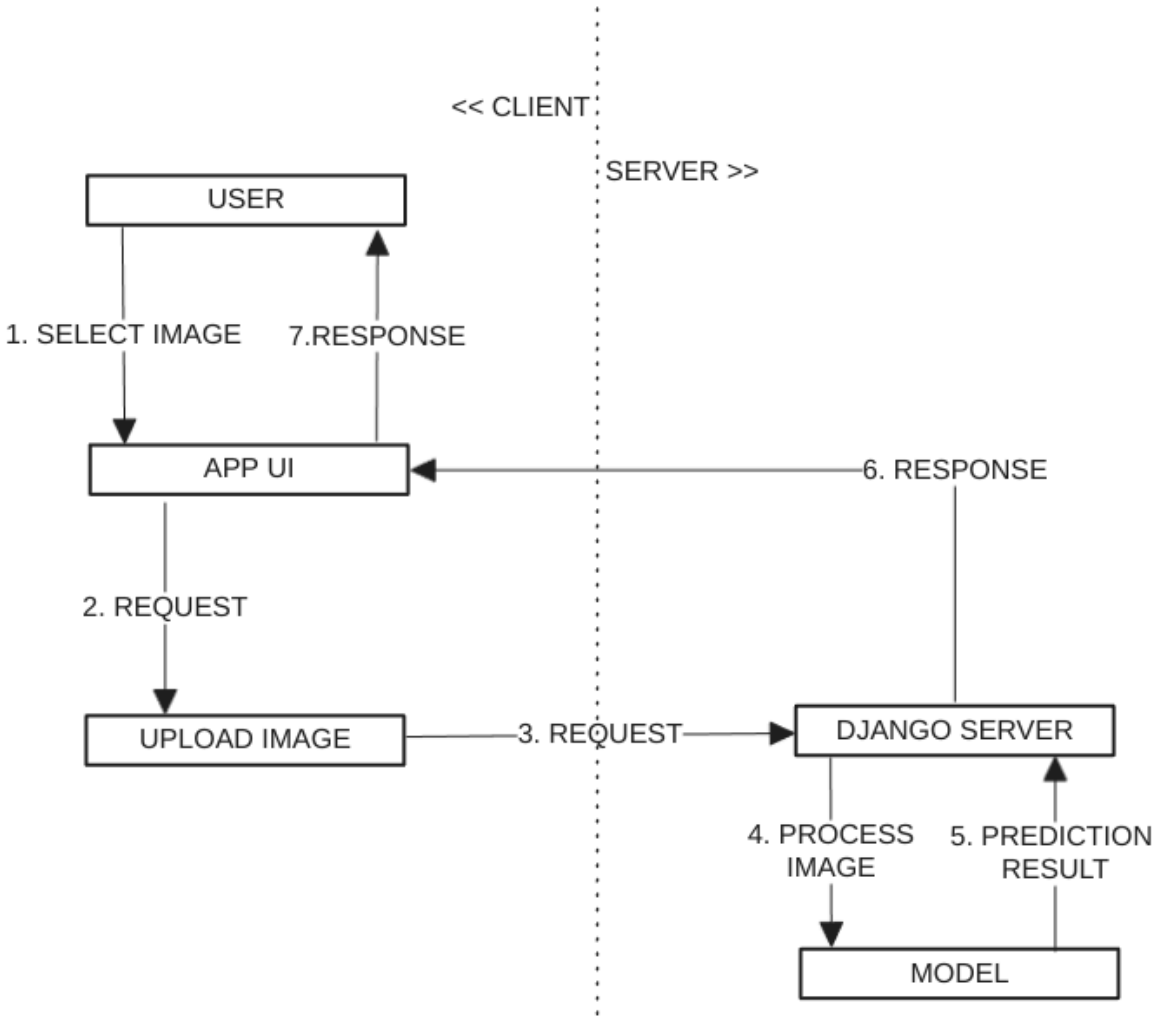

Figure 3.4: App Logic

1. Select Image: User should click the button to choose an image from the gallery.
2. Request: Selected photo to be identified is received by the application from the user's phone.
3. Request to server: Application sends the selected image to the Django backend server through a HTTP POST request by encoding the image to base64 format.
4. Process Image: The server processes the image and sends it through the trained model for inference.
5. Prediction Result: The model returns the identified landmark from the photo to the server.
6. Response to application: The server processes the prediction result and sends a JSON response with the prediction result in a readable format to the application.

3.2 Data Collection

To collect data, all four members of our group went to popular places/landmarks of cultural and historical significance. Using our phones, we took pictures of those places from various angles and proximity to create a wide range of pictures per landmark. To make more data for each type of class, we scraped data from the internet. Then all those pictures were collected in Google Drive and arranged in folders to create a variation-rich dataset. A total of around 1000 images were collected for 13 places which are listed below:

i. Pokhara International Airport
ii. Bindhyabasini Temple
iii. Bouddhanath Stupa
iv. Pema TS'AL Monastery
v. International Mountain Museum
vi. Gurkha British Memorial Museum
vii. IOE, Pulchowk Campus (ICTC Building)
viii. Pumdikot Shiva Statue
ix. Ramghat Monastery
x. IOE, Pashchimanchal Campus (RIC Building)
xi. Peace Pagoda Stupa
xii. IOE, Thapathali Campus (FSU Building)
xiii. Tribhuvan International Airport

3.3 Data Preprocessing

Data Processing is the most important step in building a machine-learning model. Data is the key factor that defines how the model will turn out to be i.e. it could make the model overfit or underfit. If data is imbalanced, it will be based on the class that has a higher number of samples. If the used data is corrupted in any form, the model will not be able to predict the user's input images accurately. We have to make sure the dataset is balanced, in proper format, and account for all the illumination conditions that the photo could be taken in by users. The entirety of the data preprocessing constitutes the following phases:

3.3.1 Collect and organize data:

We went to nearby landmarks in Pokhara, took photos in various conditions and scraped for other landmarks. Some of the data looked the same in terms of illumination, perspective, and other conditions and were ignored. Finally, more than 500 images were taken and organized in Google Drive for further processing.

3.3.2 Image Annotation/ Labeling:

An open-source library; LabelImg was used to annotate objects in our images with bounding boxes. Annotations were created containing class labels, top-left corner, and length & breadth to bottom-right corners in YOLO format created by drawing a rectangle over our desired object in each image. Now the dataset includes a text file for each image.

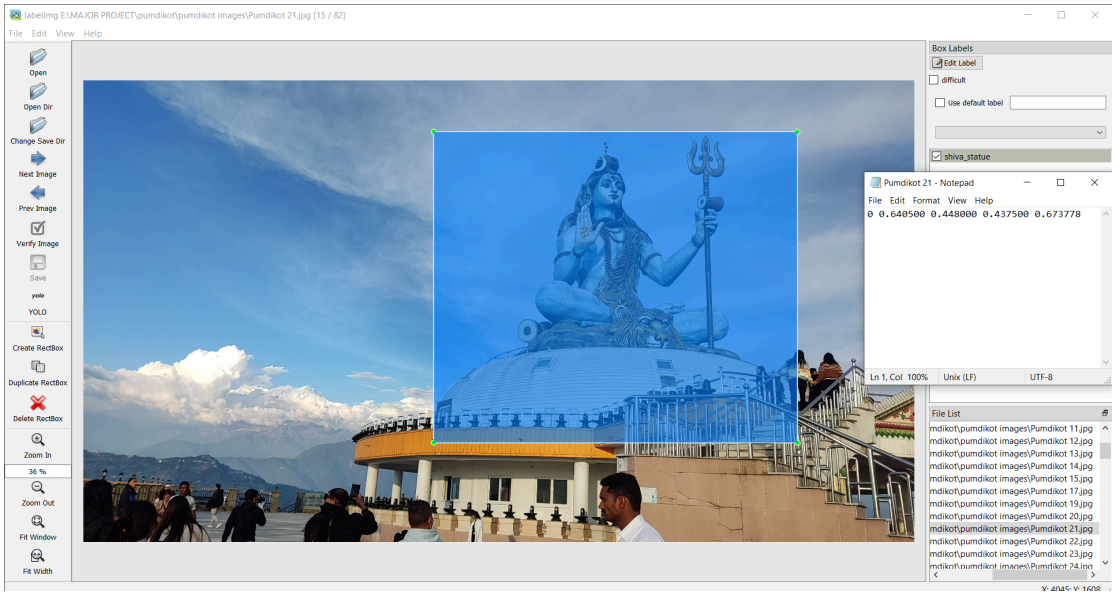

Figure 3.5: Creating Annotation of Image using LabelImg

3.3.3 Data Split

We captured most of the images of landmarks in Pokhara and scraped images for other landmarks. We didn't use duplicate images resulting in the number of images used for training being 452 and the number of images used for validation being 61.

3.3.4 Data augmentation

For this part, we used PyTorch's torchvision transform API and Pillow to grayscale and normalize training images and validation images as we wouldn't want to change images that

would not sync with their annotations. For test images, we used random horizontal flips, grayscale, normalize, rotate and other augmentation methods.

Figure 3.6: Original Image vs Augmented Images

3.3.5 Generate YOLO configuration files:

We created a YOLO configuration file (.yaml file) specifying the model architecture, training parameters, paths to our training and validation sets and many other parameters. In the YOLO (You Only Look Once) object detection model, the .yaml configuration file is crucial for defining the architecture and parameters of the model. This file included details such as the number of classes and their labels, the path to the training and validation dataset and other parameters.

3.4 Model Generation

Classification would look at the whole image to predict if any class exists or not without taking account of the object if present is in the background or foreground. Object Detection on the other hand will localize the area of occurrence of the class if any and utilize bounding boxes to predict where it exists. If there is more than one class in the same picture, it will be able to predict both classes. So, for our task of Landmark Recognition System, object detection is the better-suited approach.

Object detection identifies and locates multiple objects in images or videos, combining classification i.e. assigning class labels and localization i.e. determining bounding box coordinates. It involves predicting bounding boxes and class labels for multiple objects in an image, integrating both localization and classification tasks.

Classification distinguishes the primary object in an image by assigning a label without providing location details, while localization predicts the bounding box around a single object. Together, these tasks form the basis for object detection, enabling the identification and precise location of multiple objects within an image.

Object Detector can be mainly classified into two types:
i. Single-stage Detectors: Predict bounding boxes and classes in one pass, emphasizing speed. Examples include YOLO, SSD, etc.
ii. Double-stage Detectors: Propose regions in the first stage, refining predictions in the second for higher accuracy. Examples include R-CNN, Faster R-CNN, Mask R-CNN, etc.

Thus object detection is crucial in various applications. Single-stage detectors prioritize speed, while double-stage detectors balance accuracy and computation, depending on specific application requirements.

YOLOv8 is the latest SOTA model by Ultralytics that performs best for Classification, Segmentation, Detection, Tracking and Pose estimation tasks. Various functional parts make up the building blocks of YOLO and those building blocks finally form a working model.

The YOLOv8 builds on its predecessor versions and consists of mainly two parts; Backbone and Head.

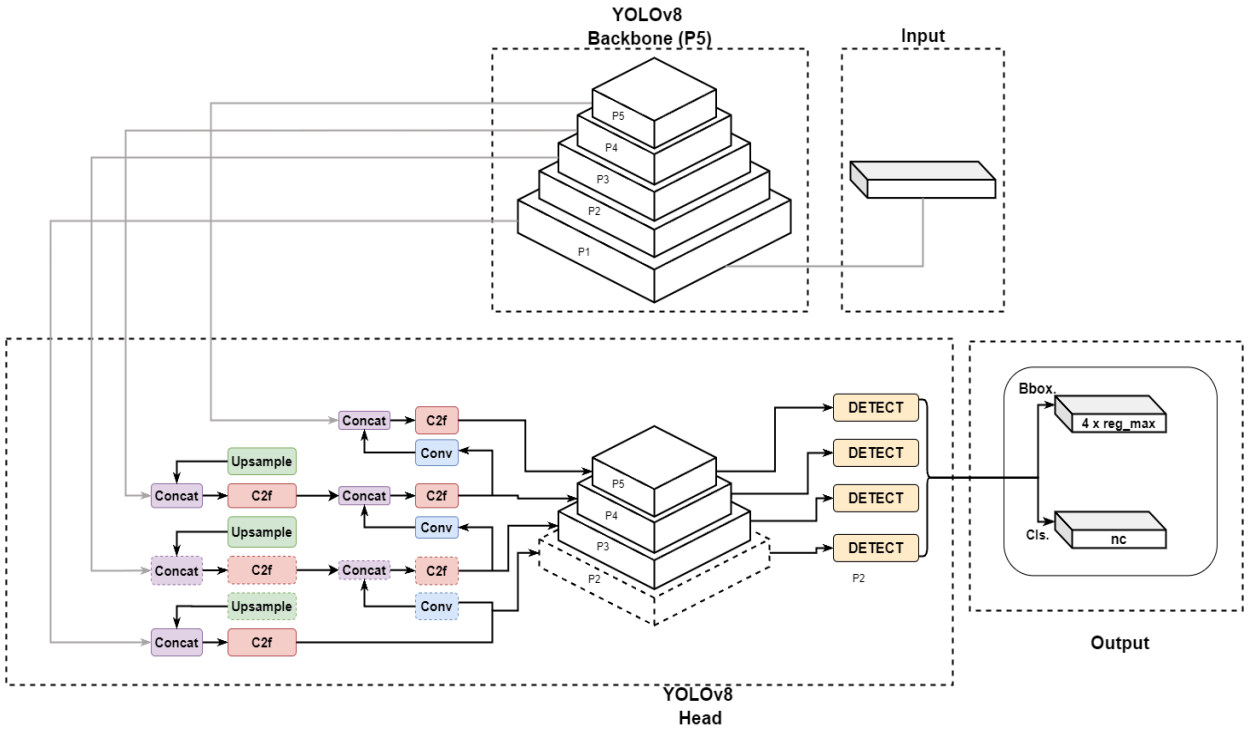

Figure 3.7: Reference Image for YOLOv8 Architecture

3.4.1 Model Architecture

3.4.1.1 Backbone

YOLOv8's backbone is a modified version of the CSPDarknet53 architecture. This backbone is composed of 53 convolutional layers, providing a deep and rich feature extraction process. YOLOv8 employs cross-stage partial connections, facilitating enhanced information flow between layers. This improves the model's ability to capture complex hierarchical features.

In short, CSPDarknet53 is a deep neural network architecture mostly used in Object Detection networks for feature extraction as it utilizes convolution blocks and residual connections with proper dimensions.

3.4.1.2 Head

The head of YOLOv8 consists of multiple convolutional layers followed by a series of fully connected layers. The layers in the head are responsible for predicting essential components,

including bounding boxes, objectness scores, and class probabilities for detected objects in an image.

YOLOv8 incorporates a self-attention mechanism within the head of the network. This mechanism enables the model to dynamically focus on different parts of the image during processing. It adjusts the importance of various features based on their relevance to the overall detection task.

3.4.1.3 Spatial Pyramid Pooling Fast

YOLOv8 excels in multi-scaled object detection through the utilization of a feature pyramid network. The feature pyramid network comprises multiple layers designed to detect objects at various scales within an image. This design allows the model to effectively detect both large and small objects within the same image, enhancing the versatility of the object detection capability. This can be considered neck as slightly mentioned in the YOLOv8 documentation.

3.4.1.4 Benchmarks

The accuracy of YOLO models is checked on the COCO dataset in which the v8 model performed better than other models including its predecessor models, making it a SOTA model. The Mean Average Precision (mAP) score is used to test the performance of object detection models. The mAP scores on the COCO validation dataset for the model were 53.9 which is around 20% more than its previous versions.

3.4.1.5 Validation

To ensure the robust performance of our Landmark Detection project using YOLOv8, a meticulous model validation process was undertaken. A custom validation dataset was curated, encompassing diverse images from each landmark class, coupled with corresponding annotations stored in .txt files. This dataset was instrumental in assessing the model's ability to accurately identify and localize landmarks. The validation process involved rigorous testing on images not seen during training, gauging the model's generalization capabilities. Data Augmentation techniques were applied to prevent model bias in the training phase. A multi-class confusion matrix was formed to find out the performance of the model in which when normalized, the model predicted each class with ease. Through this comprehensive validation strategy, we aimed to ensure the reliability and

effectiveness of our YOLOv8-based Landmark Detection system, providing users with a trustworthy tool for recognizing and learning about culturally significant places.

3.4.2 Fine Tuning

Pre-trained models are those models that are already trained on some dataset and have some parameters associated with them. Untrained models when trained on some dataset will backpropagate gradients to update parameters to make the model optimal to perform on the dataset it was trained on. This means that pre-trained models already have some parameters that can provide useful results. These models are then fine tuned as per our need and utilized for better performance without going through the trouble of complex computation and time-consuming training process. The YOLOv8 utilized is pre-trained and fine tuned on our landmark dataset and configured using the YOLO configuration .yaml file. Even while fine-tuning, YOLO will utilize the optimizers, loss function and metrics discussed below so that it results in an optimal model. Some details in finetuning the model are as:

No of layers	225
Parameters	11130615
Gradients	11140615
Speed	28.7 GFLOPs
Preprocess Time	0.6ms
Inference Time	10.2ms
Loss Calculation	0.0018 ms
Postprocess Time	4.04s
Training time	0.463hr

Table 1: Fine Tuning Results

3.4.2.1 Optimization Algorithm

AdamW is used with a learning rate of 0.000588[11]. The AdamW (Weight Decay Fix) is a modification of the Adam optimization algorithm that addresses the issue of weight decay. Traditional Adam optimization includes weight decay directly in the update step, but it has been observed that this can lead to suboptimal solutions in some cases.

The update step for AdamW is as follows:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t}+\epsilon} \cdot \hat{m}_t - \alpha \cdot \text{weight\_decay} \cdot \theta_t$$

θ_t is the parameter vector at time step t.
α is the learning rate.
$\hat{m}_t$ is the biased first moment estimate.
$\hat{v}_t$ is the biased second raw moment estimate.
ϵ is a small constant to prevent division by zero
weight_decay is the weight decay term.

Equation 1: Adam Optimization

3.4.2.2 Loss Function

BCE for classification and DFL and CIoU losses for bounding box regression are used as loss functions. The binary cross-entropy loss provides a measure of dissimilarity between the predicted probability distribution and the ground truth labels. The binary cross-entropy loss is computed as the average of the negative log-likelihood of the predicted class probabilities.

$$\text{BCE}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)]$$

Equation 2: Binary cross-entropy loss

3.4.2.3 Evaluation Metrics

Intersection over Union (IoU): IoU is a measure that quantifies the overlap between a predicted bounding box and a ground truth bounding box. It plays a fundamental role in evaluating the accuracy of object localization.

Average Precision (AP): AP computes the area under the precision-recall curve, providing a single value that encapsulates the model's precision and recall performance.

Mean Average Precision (mAP): mAP extends the concept of AP by calculating the average AP values across multiple object classes. This is useful in multi-class object detection scenarios to provide a comprehensive evaluation of the model's performance.

Precision and Recall: Precision quantifies the proportion of true positives among all positive predictions, assessing the model's capability to avoid false positives. On the other hand, Recall calculates the proportion of true positives among all actual positives, measuring the model's ability to detect all instances of a class.

F1 Score: The F1 Score is the harmonic mean of precision and recall, providing a balanced assessment of a model's performance while considering both false positives and false negatives.

3.4.2.4 Training Hyperparameters

Train Image Size	640*640
Val Image Size	640*640
No of Epochs	20
Batch Size	04
Optimizer	AdamW
Learning Rate, Momentum	0.000588, 0.9

Table 2: Training hyperparameters

During training, checkpoints were saved and the best model is saved having the minimum loss.

3.4.2.5 Training Environment

The model for the landmark recognition system was trained within the robust environment provided by Google Colab, leveraging its GPU capabilities to expedite the training process. With access to 12GB of GPU and system RAM, coupled with ample disk space totalling

78GB, Google Colab provided a conducive setting for training resource-intensive deep learning models. Moreover, the utilization of Python 3 Google Compute Engine backend with GPU support ensured efficient computation and optimization during the training phase. Google Colab's GPU support facilitated faster model convergence and scalability, enhancing the system's accuracy and robustness.

3.5 Deployment

3.5.1 Frontend Development

The user interface, built with React Native, provides a seamless experience across Android and iOS devices. Users can register and log in securely using JWT tokens for authentication. To capture images of landmarks, the app utilizes the device's gallery access. User location services are utilized to enhance the experience, allowing for features like displaying nearby landmarks or providing directions via an integrated Google Maps component. This integration showcases the identified landmarks and other registered landmarks retrieved from the backend through a RESTful API communication channel. This section emphasizes the technical aspects of frontend development, including framework choice, user interaction elements, location services, integration of external APIs, and communication protocols.

3.5.2 Backend Development

The mobile app's development necessitates a robust backend system, which, in this case, was crafted using Django REST Framework, a contemporary and swift web framework tailored for building APIs. Within this framework, an endpoint was devised to handle POST requests containing base64 encoded images in JSON format. Upon receiving such requests, the API executes a sequence of preprocessing tasks on the input images, followed by the utilization of a meticulously trained model for detecting Regions of Interest (ROIs). This model, having undergone extensive training, represents the best-performing iteration selected for this study. After the detection process, the backend system employs database queries to retrieve data regarding the predicted landmark, generating a JSON response that is then transmitted to the client soliciting the extraction of textual information from the images.

3.6 Testing

The Machine Learning model as well as the mobile application developed was tested using various types of inputs and edge cases and the errors encountered were handled using different logic implemented via code blocks.

3.7 Tools Used

For the project, the following libraries, frameworks, and tools have been used:

3.7.1 Jupyter notebook

Jupyter Notebook, an open-source web application, is an indispensable tool in the realm of data science and scientific computing. Developed as part of the Jupyter Project, it allows users to create and share documents containing live code, equations, visualizations, and narrative text. [15]

3.7.2 Google colab

Google Colab is a cloud-based platform provided by Google that enables users to write, run, and collaborate on Python code in a convenient and interactive environment. It offers a range of features and benefits that make it a popular choice for data scientists, researchers, and developers.

3.7.3 PyTorch

PyTorch is an open-source machine learning library that has gained widespread popularity for its flexibility, dynamic computation graph, and ease of use. Developed by Facebook's AI Research lab (FAIR), PyTorch provides a platform for building and training neural networks with a focus on providing a more intuitive and Pythonic interface for developers.[16]

3.7.4 TensorFlow

TensorFlow, an open-source deep learning framework developed by Google, has become synonymous with machine learning and has played a key role in advancing the field. With his rich toolset, TensorFlow enables researchers, developers, and data scientists to create and deploy state-of-the-art machine learning models.

3.7.5 NumPy

NumPy is a fundamental open-source library for numerical computing in Python. It provides a powerful array object and a collection of functions that allow for efficient manipulation of large multidimensional arrays and matrices. NumPy forms the foundation of many scientific and data-oriented Python packages and is a vital tool for numerical computations.

3.7.6 Pandas

Pandas is a powerful open-source library in Python that provides high-performance data manipulation and analysis capabilities. It is built on top of NumPy and offers a flexible and intuitive way to handle structured data, making it an indispensable tool for data scientists and analysts.

3.7.7 Matplotlib

Matplotlib is a widely used data visualization library in Python that provides a comprehensive set of tools for creating static, animated, and interactive plots. It offers a flexible and customizable interface for generating a wide range of visualizations, making it a popular choice for data scientists and researchers.

3.7.8 PIL

PIL, now known as Pillow, is a feature-rich Python library that empowers users to perform a wide array of image processing tasks. With its extensive collection of functions and classes, PIL offers a comprehensive suite of tools for opening, manipulating, and saving images in various formats.

3.7.9 OpenCV

OpenCV is a widely used open-source library for computer vision and image processing tasks. It provides a rich collection of functions and algorithms that enable users to perform a diverse range of image and video analysis tasks. OpenCV offers comprehensive support for image and video I/O operations, allowing users to read and write images and videos in various formats.

3.7.10 Django

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

3.7.11 React native

React Native, an open-source framework by Facebook, facilitates the development of

cross-platform mobile applications. It empowers developers with a rich set of tools and components for building high-quality, visually engaging apps compatible with web, iOS and Android platforms.

3.7.12 Visual studio code

Visual Studio Code is a popular code editor which was used in our project to write code for both the React Native frontend and the Django backend server. It provides a wide range of features and extensions that enable developers to code, test, and debug their applications and provides a good experience to developers.

Chapter 4: RESULTS AND DISCUSSION

4.1 Model Evaluation

Class	Images	Instances	Box(P	R	mAP50	mAP50-95)
all	61	61	0.977	1	0.995	0.799
pia	61	5	0.989	1	0.995	0.774
bindabasini	61	5	0.995	1	0.995	0.813
bouddha	61	5	0.989	1	0.995	0.379
hemja	61	5	0.974	1	0.995	0.747
mountainmuseum	61	5	0.977	1	0.995	0.835
museum	61	5	0.978	1	0.995	0.915
pulchowk	61	3	0.999	1	0.995	0.953
pumdikot	61	5	0.968	1	0.995	0.652
ramghat_gumba	61	5	0.957	1	0.995	0.873
ric	61	5	0.972	1	0.995	0.829
stupa	61	5	0.968	1	0.995	0.903
thapathali	61	4	0.961	1	0.995	0.885
tia	61	4	0.971	1	0.995	0.835

Figure 4.1: Model Validation Summary

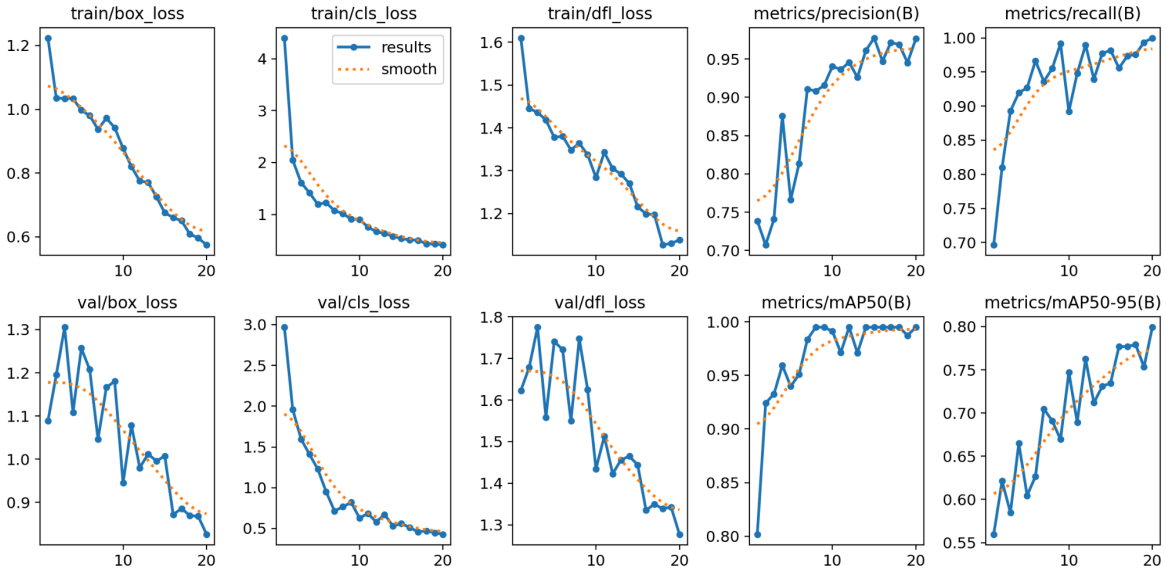

Figure 4.2: Losses and Metrics

The above figure displays diverse losses such as box loss, class loss, and distribution focal loss. Additionally, it illustrates several metrics including mAP, precision, and recall. This representation likely provides insights into the model's optimization for different

components, as well as its performance in terms of classification and localization capabilities.

Box loss: The box loss, often referred to as localization loss, is associated with the accuracy of predicting bounding box coordinates (x, y, width, height) for each object in the image. It penalizes the model for inaccuracies in predicting the position and size of bounding boxes. Common regression loss functions like Mean Squared Error (MSE) are often used to compute the box loss.

Initially, during the early stages of training, the box loss for the training and validation sets stood at 1.224 and 1.0887, respectively. As training progressed, both losses gradually decreased, approaching almost 1 by the 5th epoch for training and 1.25 for validation. However, a significant drop occurred in the training loss, while the validation loss exhibited oscillations but continued to decrease over the long run.

Class loss: Class loss pertains to the accuracy of predicting the correct class label for each bounding box. This component penalizes the model for errors in classifying objects. Cross-entropy loss is a common choice for computing the class loss in object detection tasks. It measures the dissimilarity between predicted class probabilities and the ground truth class labels.

Initially, the training class loss was 4.4037, and the validation class loss was 2.9692. As training progressed, both losses decreased, reaching approximately 1.61 for training and 1.59 for validation by the 3rd epoch. Then both the training and validation box loss continued to drop down.

Distribution Focal Loss (DFL): The Distribution Focal Loss (DFL) is a modification or enhancement to the standard focal loss used in object detection. Focal loss aims to address the issue of class imbalance in the dataset, where there are many more background (negative) examples than foreground (positive) examples.

In the early stages of training, the distribution focal loss (DFL) for the training set was 1.6103, and the validation set had a DFL of 1.6231. Subsequently, training DFL showed a gradual decline, with the training DFL reaching 1.13 at the end. On the flip side, validation DFL kept on oscillating and eventually converging at the end.

Precision: Precision is the ratio of true positive predictions to the total predicted positives, gauging the accuracy of positive predictions.

Starting at 0.73869 in epoch 1, Precision gradually improves, reaching 0.97672 by epoch 20. The most significant change occurs between epochs 5 and 10, with precision rising dramatically from 0.76616 to 0.94045, indicating a notable enhancement in correctly identifying positive instances. Overall, Precision demonstrates consistent improvement throughout training.

Recall: Recall, or sensitivity, is the ratio of true positive predictions to the total actual positives, measuring the model's ability to capture all relevant positive instances.
Starting at 0.69653 in epoch 1, Recall steadily improves, reaching 1 by epoch 20, indicating that the model achieves a perfect recall, capturing all relevant positive instances. Notably, there is a slight decline between epochs 5 and 10, where recall decreases from 0.92671 to 0.89197. Overall, Recall demonstrates an upward trend, achieving a remarkable performance by the end of the training period.

mAP (mean Average Precision): mAP is a comprehensive metric in object detection, considering precision across different confidence thresholds and providing an overall assessment of the model's detection accuracy.
Initiating at 0.80182 in epoch 1, the mAP50 metric shows a steady ascent, achieving a notable improvement in object detection precision. By epoch 8, it reaches a perfect score of 0.995, signifying the model's consistent and high performance in detecting objects at an IoU of 0.5. This exceptional level of precision is sustained throughout the later stages of training.

In graphical representation, the downward trend in any loss such as box loss, class loss and distribution focal loss shows that the model is learning and loss is decreasing with an increase in epochs while the upward trend in metrics like precision, recall, and mean average precision shows that the model is getting better with each epoch of training.

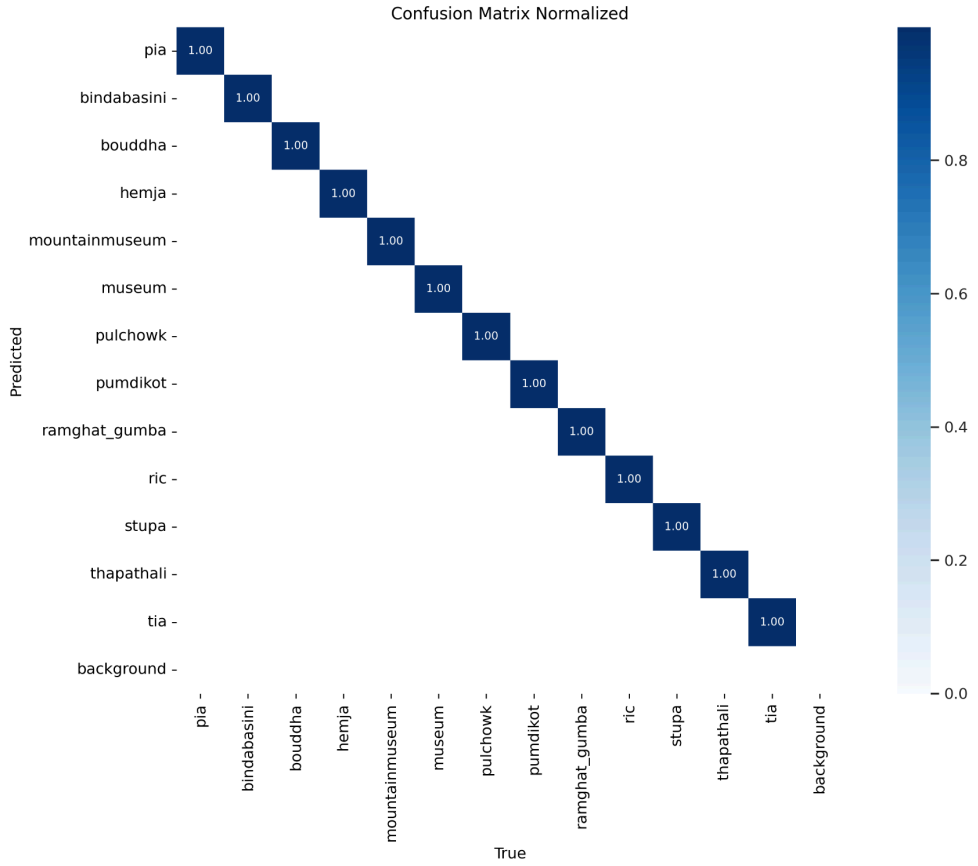

Figure 4.3: Normalized Confusion Matrix

A confusion matrix is a tabular representation employed in classification tasks to assess a model's performance by detailing the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions. This matrix offers a granular breakdown of the model's accuracy, aiding in the identification of strengths and weaknesses, and guiding improvements.

Our confusion matrix shows all the classes have TP outcomes, this shows that our model is very accurate and robust for a variety of images.

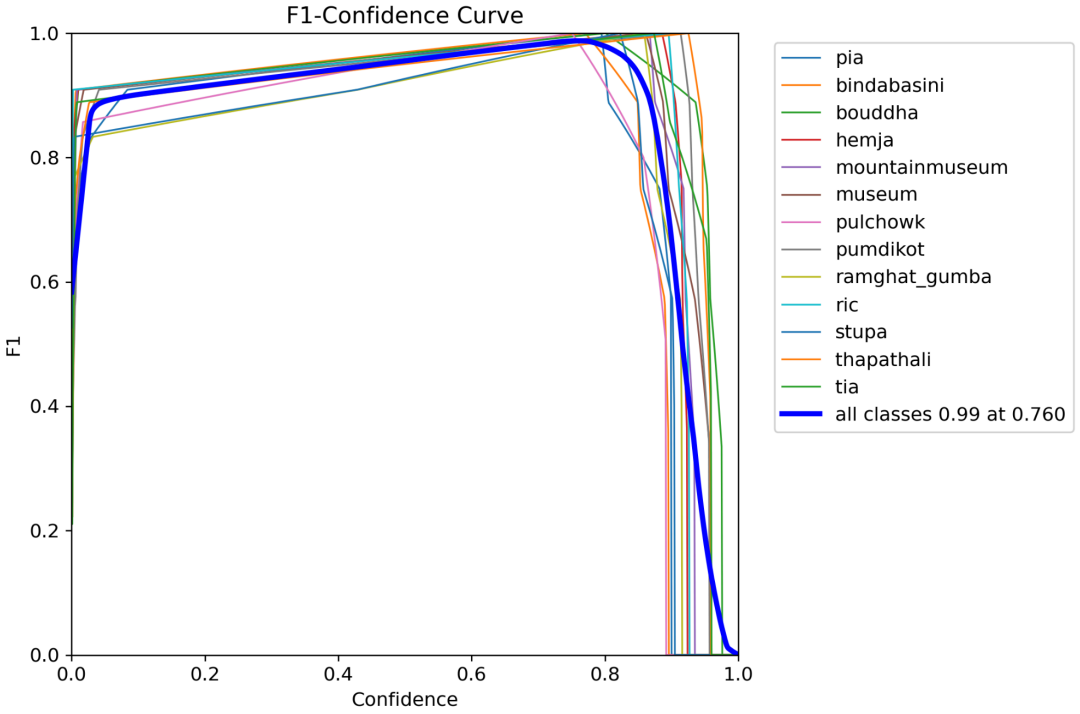

Figure 4.4: F1 Confidence Curve

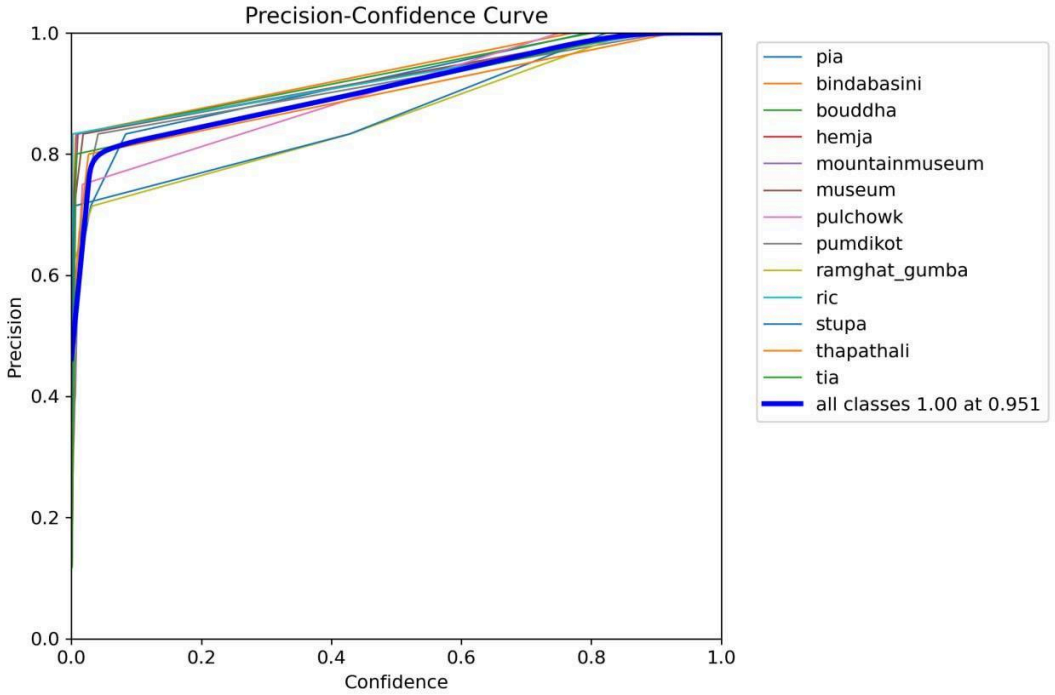

Figure 4.5: Precision-Confidence Curve

In Figure 4.4, the F1 score serves as a comprehensive metric for a model's accuracy,

incorporating both precision and recall by calculating their harmonic mean. The F1 Confidence Curve visually depicts the F1 score across varying confidence thresholds. A superior F1 score signifies improved model performance, and the confidence threshold corresponding to the maximum F1 score is often deemed the optimal threshold for prediction decisions.

In Figure 4.5, the Precision-Confidence Curve displays precision about different confidence thresholds. This visualization aids in comprehending how precision evolves with adjustments to the confidence level for categorizing a prediction as positive. Ideally, a desire is for high precision at all confidence levels, though typically precision rises with an increase in the confidence threshold.

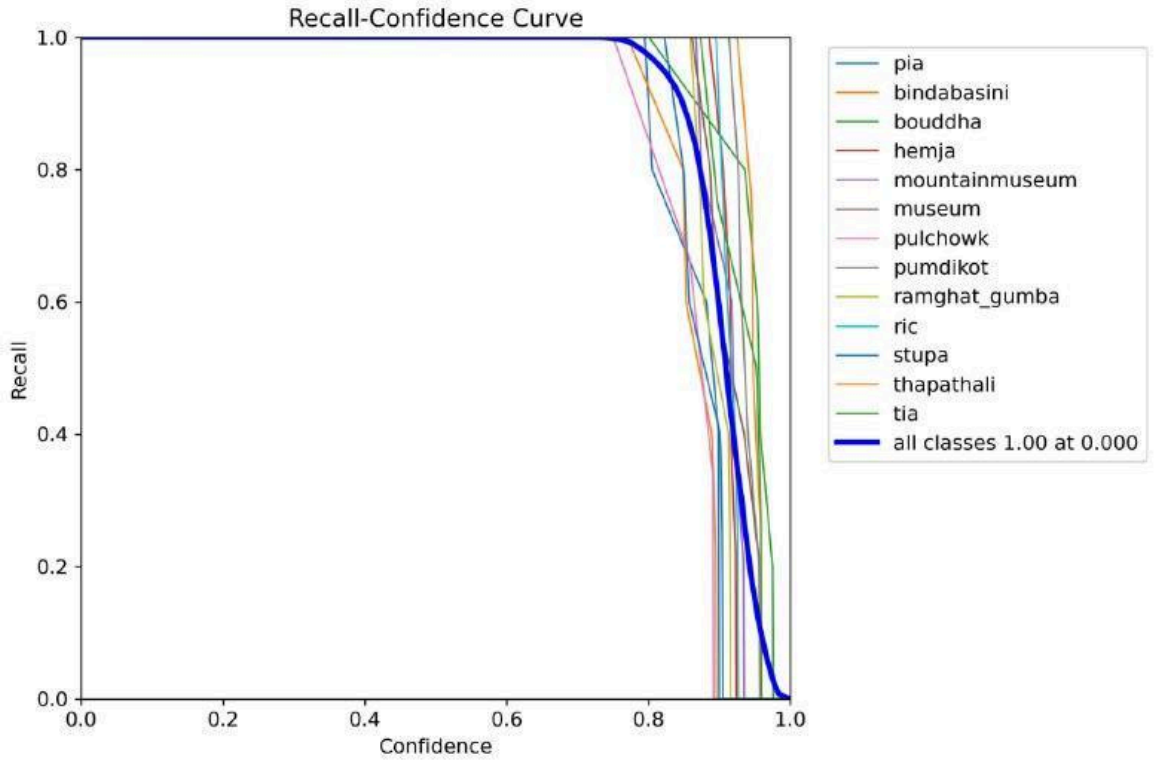

Figure 4.6: Recall-Confidence Curve

In Figure 4.6, the Recall-Confidence Curve graphically represents the relationship between recall and various confidence thresholds. It demonstrates how recall fluctuates when modifying the confidence level. Often, an increase in the confidence threshold results in a decrease in recall since a higher threshold may result in fewer positive predictions, potentially causing the model to overlook some true positive instances.

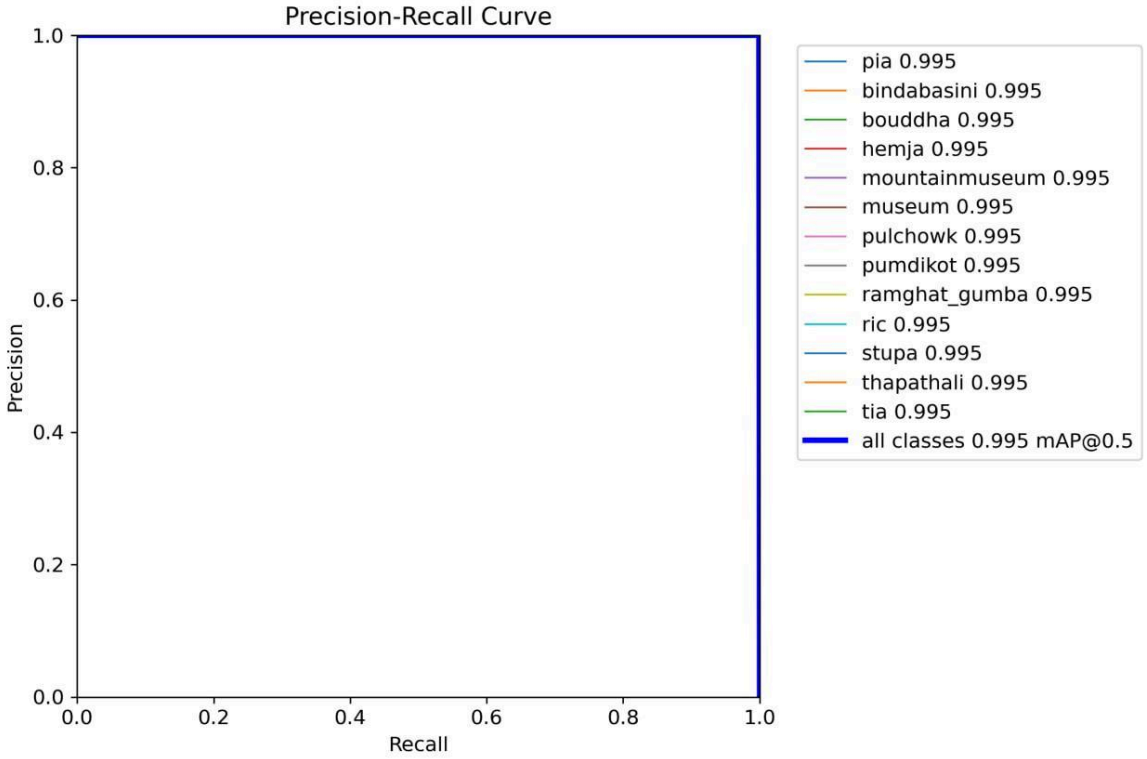

Figure 4.7: Precision-Recall Curve

The Precision-Recall Curve in Figure 4.7 illustrates the balance between precision and recall across various threshold values. Precision is defined as the proportion of accurate positive predictions relative to the total positive predictions (sum of true positives and false positives), whereas recall (also referred to as sensitivity) is the proportion of true positive predictions relative to the total actual positives (sum of true positives and false negatives).

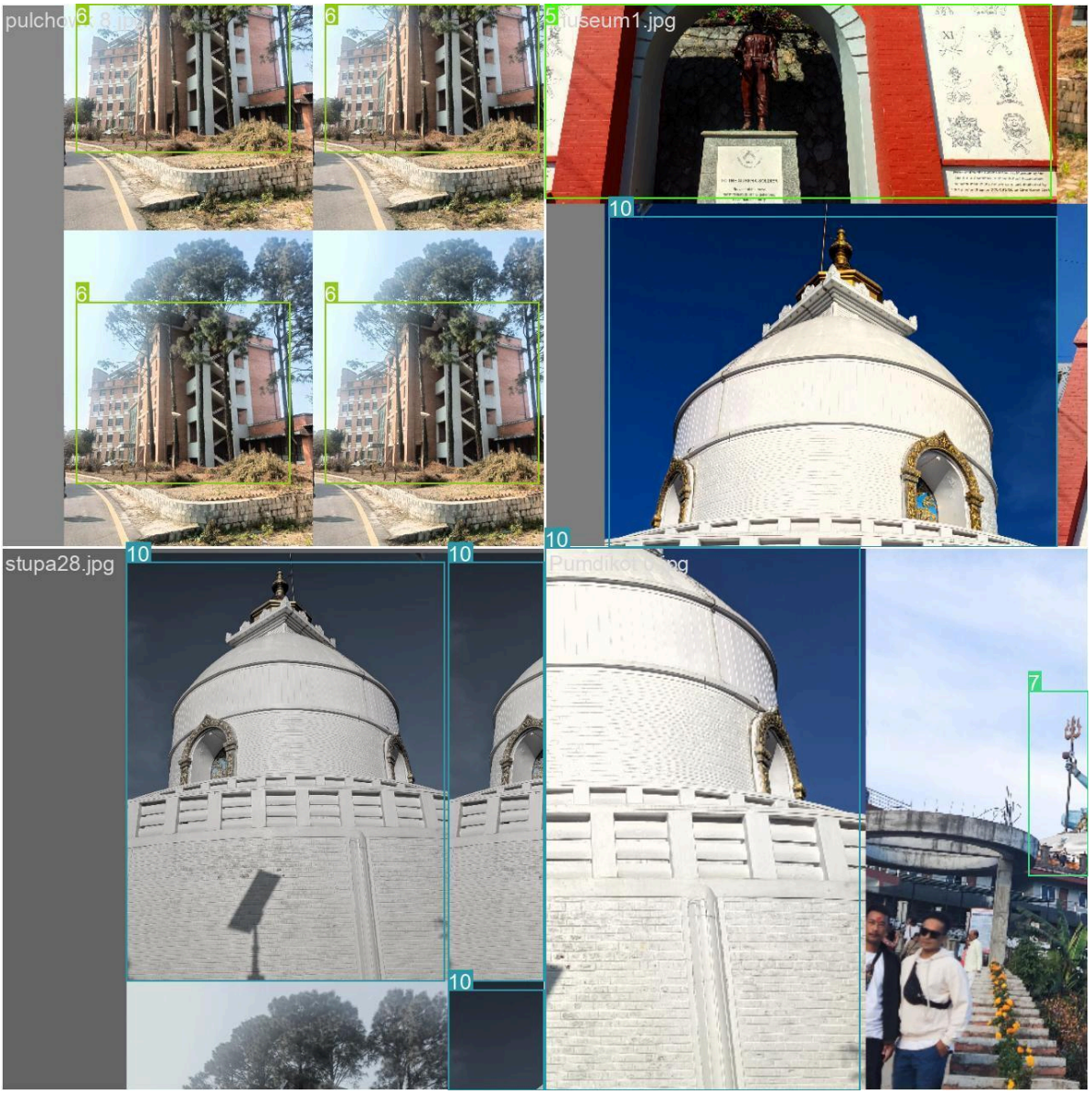

Figure 4.8: First Training Batch

The image above depicts the initial batch of training images. Given the utilization of a batch size of 4, there are 4 images in this batch. The bounding boxes in the image indicate the respective class names associated with each object.

Figure 4.9: First Validation Batch

The image above depicts the initial batch of validation images. Given the utilization of a batch size of 8, there are 8 images in this batch. The bounding boxes in the image indicate the respective class names associated with each object.

4.2 Challenges

During the full life cycle of development, several challenges emerged that required innovative solutions. Initially, the absence of an available dataset for landmarks compelled us to collect photos manually and scrape data from the internet, including contributions from friends at other IOE colleges. Additionally, the project encountered performance issues on low-end devices due to the demanding nature of the backend server and machine learning model processing. Furthermore, integrating maps for user navigation to destination landmarks posed a significant technical challenge, requiring careful implementation to ensure seamless functionality within the application. Despite these obstacles, the team persevered, employing resourcefulness and collaboration to overcome each challenge and deliver a robust and user-friendly landmark recognition system.

CHAPTER 5: EPILOGUE

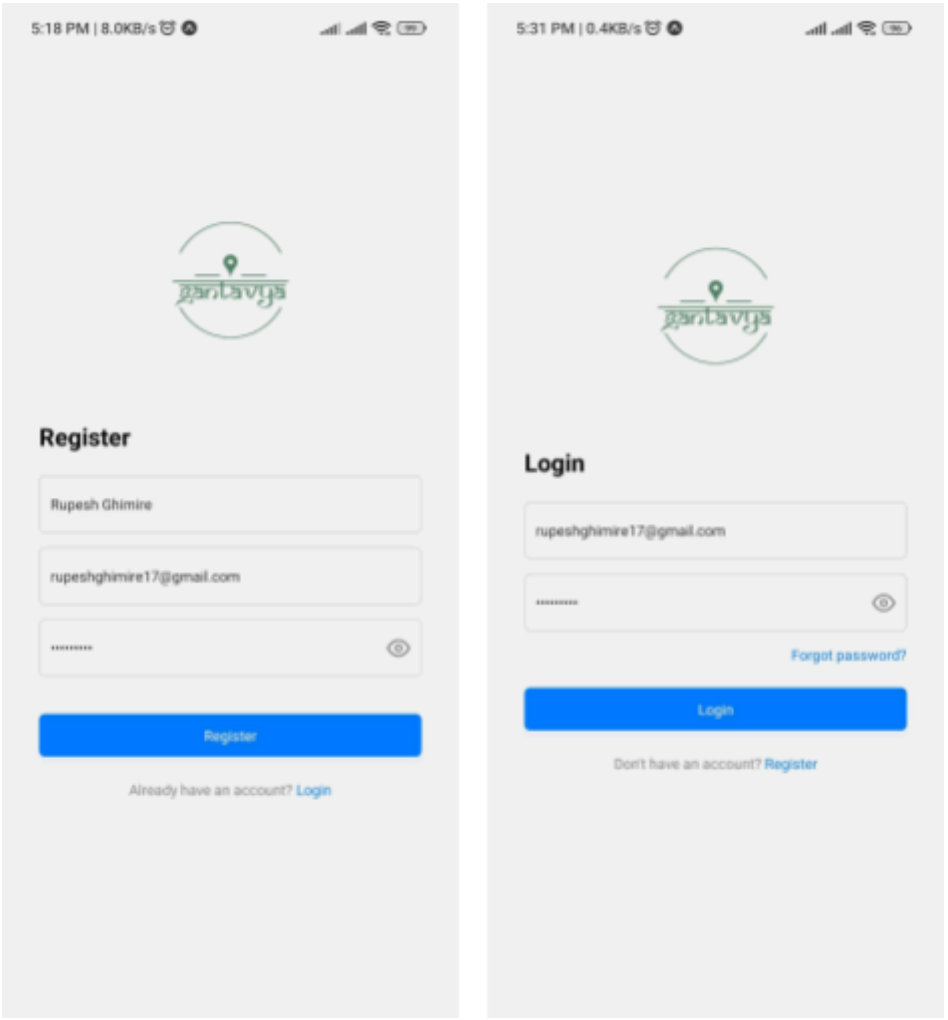

Figure 5.1: Register and Login Page

Our authentication system features a straightforward registration page requiring users to input their name, email, and password, while the login page allows access with just an email and password. Both pages utilize JWT authentication with added validation steps for credentials providing secure and efficient user authentication.

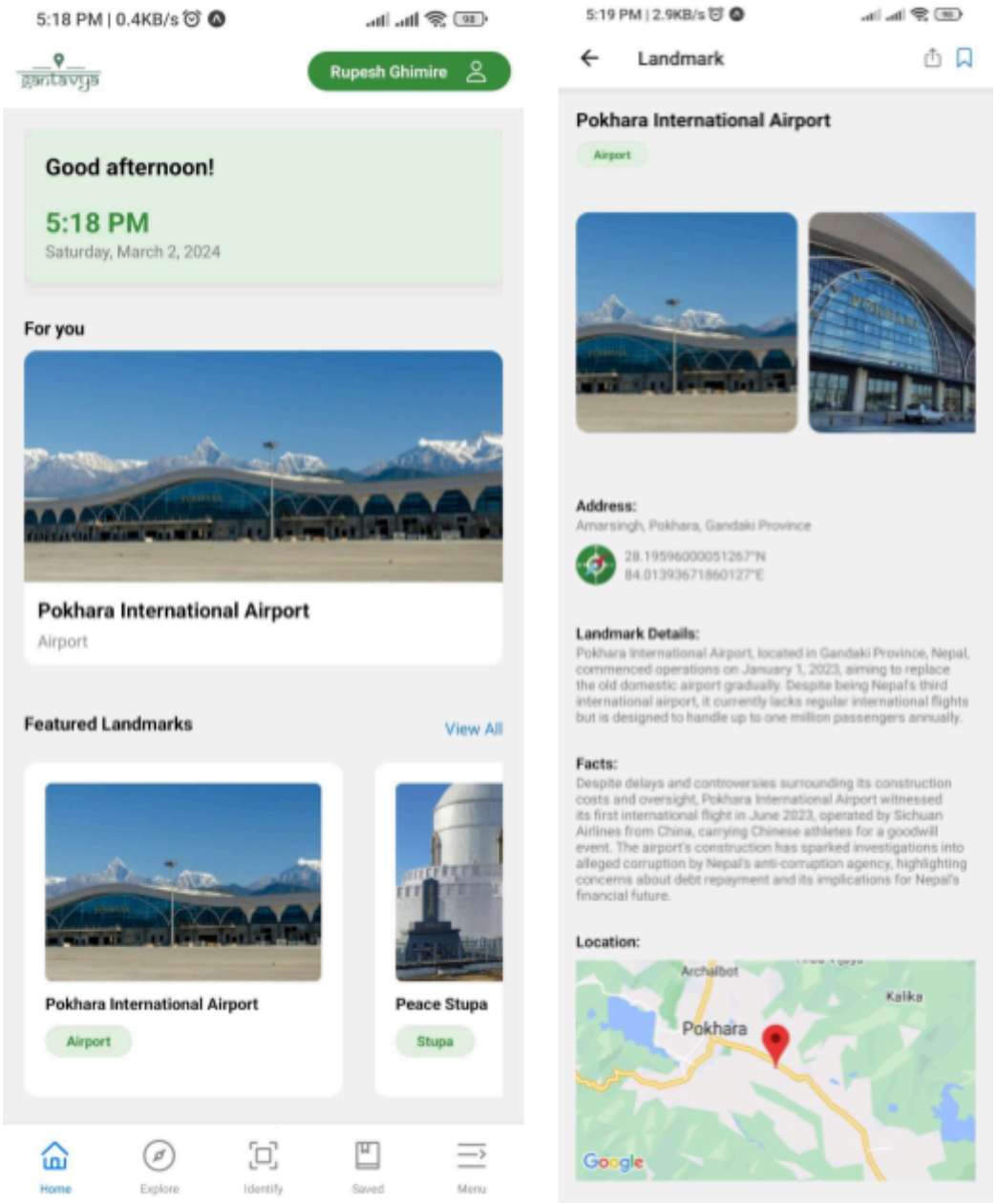

Figure 5.2: Home Page and Landmark Detail Page

Home Page: Personalized greetings and real-time display of time and date, featuring 'For You' and 'Featured Landmarks' sections, it shows users some popular and available landmarks.

Landmark Detail Page: Comprehensive details including photos, coordinates, landmark type, description, facts, and map location, providing users with in-depth information about the selected landmark.

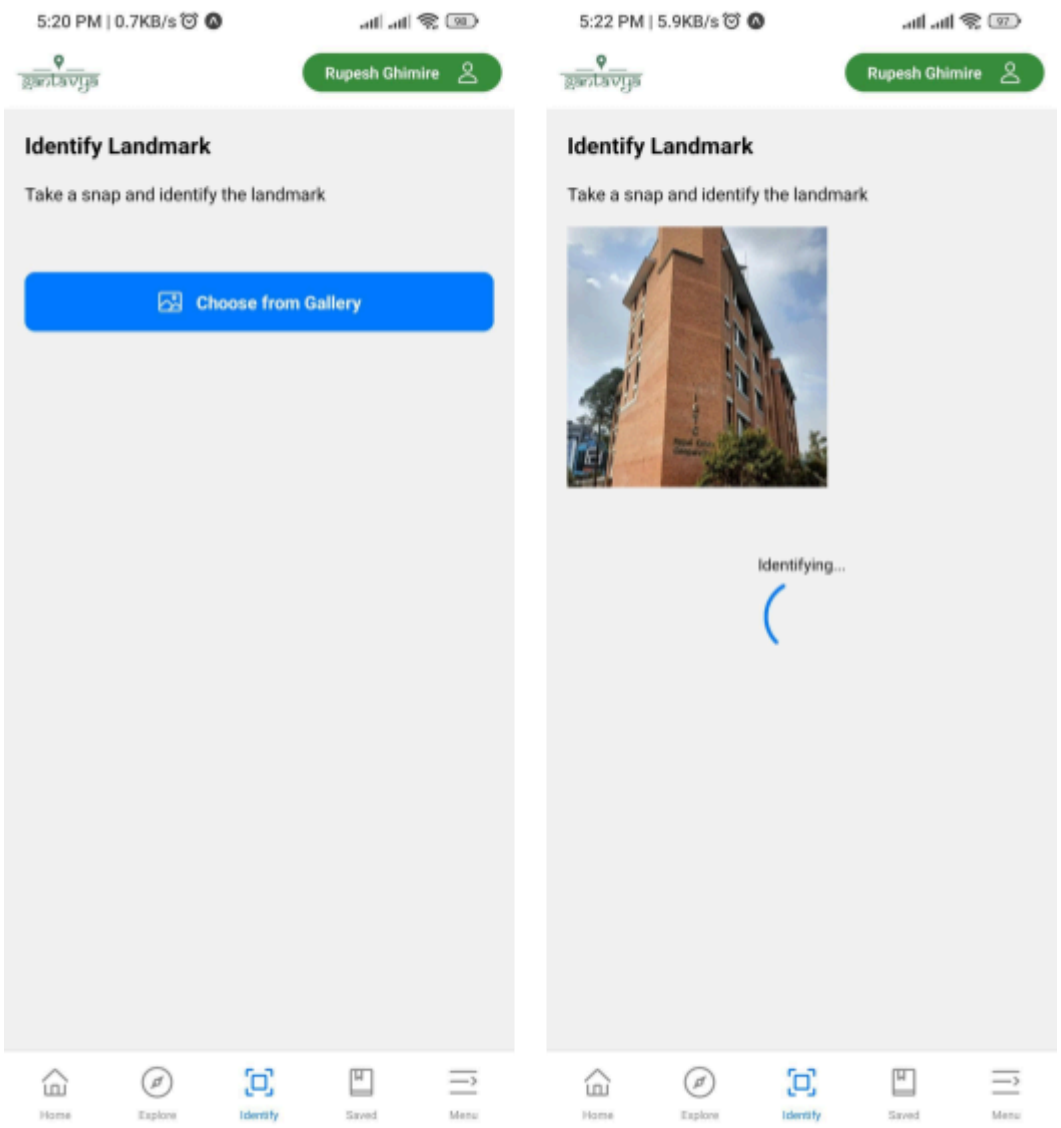

Figure 5.3: Landmark Prediction Page

Landmark Prediction Page: Allows users to select photos from their gallery, crop as necessary, and view predictions with confidence levels, displaying detailed landmark information, with options to share and save for future reference.

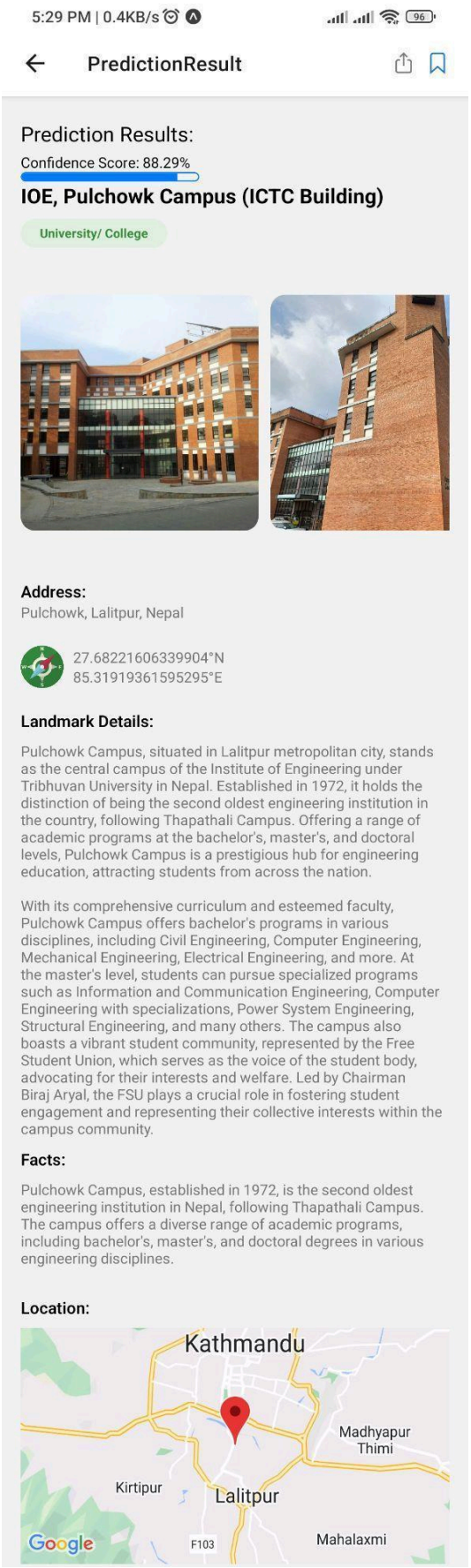

Fig 5.4: Landmark Prediction Result

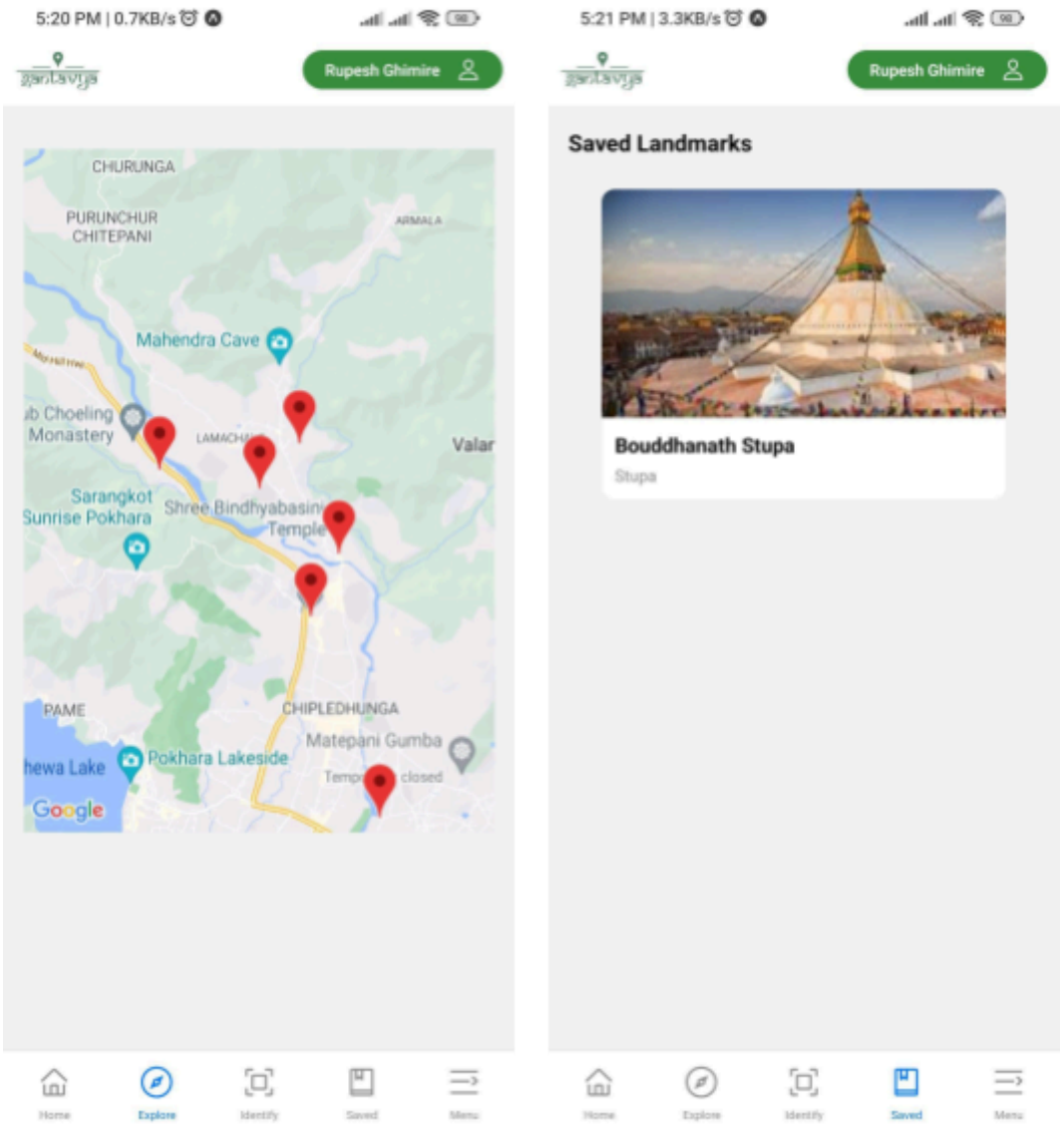

Fig 5.5: Explore Page and Saved Landmarks Page

Explore Page: Interactive map displaying pins of available landmarks, enabling users to explore various locations conveniently.

Saved Landmark Page: Access and view previously saved landmarks for quick reference and reminiscence.

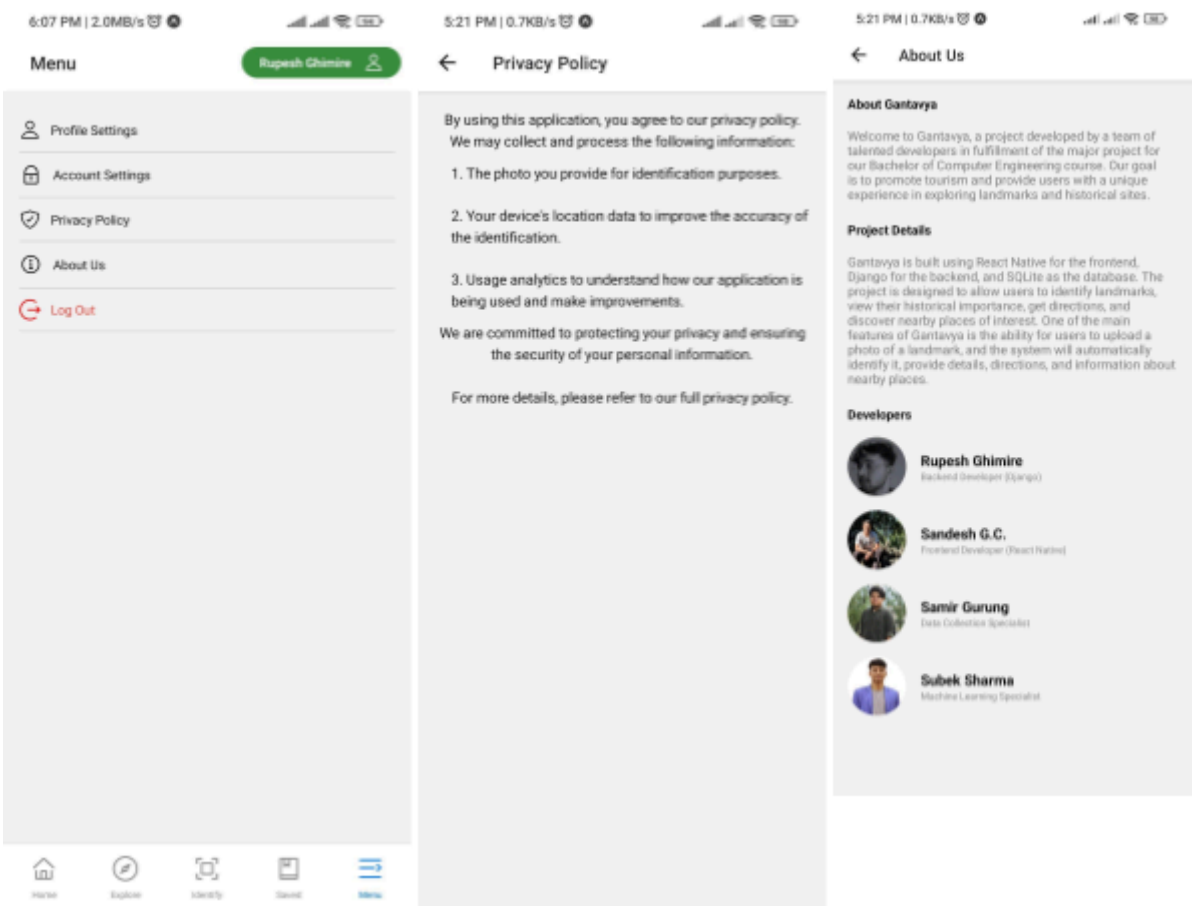

Fig 5.6: Menu Page, Privacy Policy Page and About Us

Menu Page: Provides easy access to Profile Settings, Account Settings, Privacy Policy, About Us, and Logout options for user convenience.

Privacy Policy: Details our commitment to user privacy and data protection, outlining policies governing the use and handling of user information.

About Us Page: Offers insights into our app "Gantavya" and the background of the project and team details.

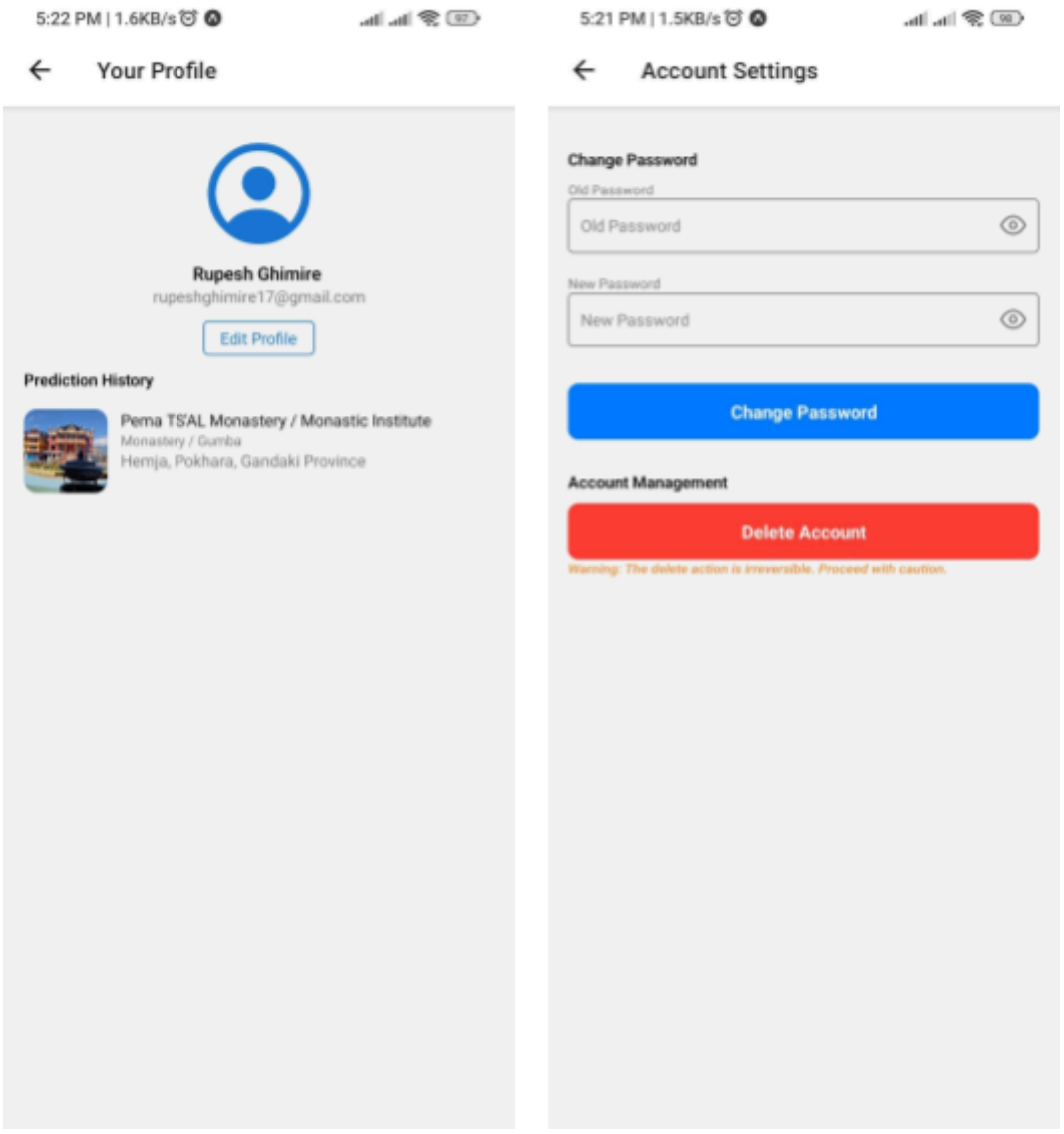

Fig 5.7: User Profile Page and Account Settings Page

User Profile Page: Users can effortlessly view their details, update name and email, and review their previous landmark identifications for a personalized experience.

Account Settings: Enables users to conveniently change passwords or delete accounts, ensuring control and security over their account information.

CHAPTER 6: CONCLUSION & FURTHER WORKS

6.1 Conclusion

In conclusion, our project has culminated in the development of a highly efficient and user-friendly Landmark Recognition System, leveraging cutting-edge technologies and methodologies. By usage of powerful libraries and frameworks such as Tensorflow, Django, and React Native, and employing advanced techniques like Data Augmentation with OpenCV and PyTorch, we have crafted a robust model capable of accurately identifying landmarks with ease.

Through our diligent efforts, we have created an application that not only excels in performance but also delivers a seamless user experience. By harnessing the power of machine learning and transfer learning, our system simplifies the process of landmark identification, offering users a convenient means to explore and learn about various landmarks primarily in Pokhara followed by some in Kathmandu.

With the elimination of manual entry and the utilization of sophisticated algorithms, our project contributes to enhancing efficiency, accuracy, and accessibility in landmark recognition. This endeavor marks a significant step forward in the realm of mobile-based landmark identification systems, promising to benefit users across diverse domains and interests.

6.2 Further enhancement

In future enhancements, expanding the training data for our models stands as a pivotal step towards improving accuracy and robustness. By incorporating a larger and more diverse dataset, we can fine-tune our algorithms to better recognize landmarks across varied conditions and environments. Additionally, implementing stemming and lemmatization techniques can further refine our model's understanding of natural language, enhancing its ability to process and interpret user inputs effectively.

Moreover, ensuring the stability and reliability of our mobile application is paramount. Conducting thorough alpha and beta testing on multiple platforms enables us to identify and address any potential bugs or performance issues, ensuring a seamless user experience for

all our users.

In line with advancing user interaction, integrating a chatbot feature for answering user inquiries about landmarks presents an exciting opportunity. This feature would empower users to engage in real-time conversations, seeking information about landmarks, such as historical significance, entry fees, and nearby attractions, enhancing their overall experience and satisfaction.

Furthermore, to enrich our application's functionality, incorporating landmark recognition through video clips adds another layer of convenience and versatility. This feature would enable users to identify landmarks in real time through video footage, expanding the application's utility and appeal to a broader audience of users.

Lastly, scaling our application by integrating more landmarks from all over Nepal with additional information sourced from scraping data and leveraging advanced NLP techniques for data processing holds promise for further enhancement. By harnessing the power of natural language processing, we can extract, clean, and utilize a wealth of information about landmarks, enriching our application's database and providing users with comprehensive and up-to-date insights into their surroundings.

REFERENCES

[1] T. Chen, K. Wu, K. -H. Yap, Z. Li, and F. S. Tsai, "A Survey on Mobile Landmark Recognition for Information Retrieval," in 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware, Taipei, Taiwan, 2009, pp. 625-630, doi: 10.1109/MDM.2009.107.

[2] Y. -T. Zheng et al., "Tour the world: Building a web-scale landmark recognition engine," 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, 2009, pp. 1085-1092, doi: 10.1109/CVPR.2009.5206749.

[3] J. Cao et al., "Landmark recognition via sparse representation," 2015 IEEE International Conference on Digital Signal Processing (DSP), 2015, pp. 1030-1034, 2015.

[4] A. S. Timmaraju and A. Chatterjee, "Monulens: Real-time mobile-based Landmark Recognition," 2014.

[5] J. Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788, 2016.

[6] N. Ketkar, “Introduction to PyTorch,” Deep Learning with Python, Apress, Berkeley, CA, vol 10, pp. 195-208, 2017.

[7] P. Panphattarasap and A. Calway, "Visual place recognition using landmark distribution descriptors," arXiv preprint arXiv:1608.04274, 2016.

[8] OpenCV. "OpenCV." OpenCV, https://opencv.org/. Accessed: June 15, 2023.

[9] A. Boiarov and E. Tyantov, "Large Scale Landmark Recognition via Deep Metric Learning," arXiv:1908.10192v3 [cs.CV], Aug. 29, 2019.

[10] J. P. Muñoz and S. Dexter, "Improving Place Recognition Using Dynamic Object Detection," arXiv:2002.04698v1 [cs.CV], Feb. 2020.

[11] S. Matsuzaki, T. Sugino, K. Tanaka, Z. Sha, S. Nakaoka, S. Yoshizawa, and K. Shintani, "CLIP-Loc: Multi-modal Landmark Association for Global Localization in Object-based Maps," arXiv:2402.04698 [cs.CV], 2024.

[12] T. Do and S. N. Sinha, "Improved Scene Landmark Detection for Camera Localization," presented at 3DV 2024, Jan. 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.18083.

[13] Ultralytics Documentation. "Getting Started with Ultralytics." Ultralytics, https://docs.ultralytics.com/. Accessed: December 9, 2023.

[14] TensorFlow. "Object Detection: TensorFlow Documentation." TensorFlow, https://www.tensorflow.org/hub/tutorials/object_detection. Accessed: June 15, 2023.

[15] PyTorch. "PyTorch Documentation." PyTorch, https://pytorch.org/docs/stable/index.html. Accessed: June 15, 2023.

[16] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," arXiv:1311.2524, 2013.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," arXiv:1512.03385 [cs.CV], Dec. 2015. [Online]. Available: https://doi.org/10.48550/arXiv.1512.03385.

[18] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," arXiv:2004.10934, 2020.

[19] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556 [cs.CV], Sep. 2014. [Online]. Available: https://doi.org/10.48550/arXiv.1409.1556

[20] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," Communications of the ACM, vol. 24, no. 6, pp. 381–395, Jun. 1981. [Online]. Available: https://doi.org/10.1145/358669.358692