Acknowlegment

We would like to express our heartfelt gratitude and appreciation to all those who have contributed to the successful completion of the License Plate Recognition System. First and foremost, we extend our deepest thanks to our project supervisor **Er. Manil Vaidhya** for their invaluable guidance and support throughout the development process. We are immensely grateful to our dedicated team members who have devoted their time, knowledge, and skills to bring this project to life. Their contributions and collective efforts have resulted in the creation of a comprehensive solution for License Plate Recognition System. We would also like to acknowledge the participants who volunteered their time and resources to provide valuable insights and feedback during the testing phase. Their involvement and cooperation have greatly enhanced the functionality and usability of our License Plate Recognition System. Furthermore, we express our heartfelt gratitude to our mentor **Er. Thomas Basyal** for his invaluable contributions in the realms of business and entrepreneurship and for the providing us with expert guidance. We also extend our appreciation to **Dr.Roshan Chitrakar** for their exceptional leadership and guidance throughout the project. Their wisdom and unwavering commitment have been a constant source of inspiration for the entire team.

Lastly, we would like to acknowledge Er. **Bhusan Thapa**, the Head of Software Engineering at NCIT, for their valuable support and technical expertise. To everyone mentioned above and anyone else whose contributions may have been inadvertently omitted, we extend our sincere thanks. Your support and involvement have played a pivotal role in making License Plate Recognition System project a reality. We are immensely proud of the outcome and look forward to the positive impact it will have on the government.

Abstract

This study presents an innovative approach for detecting, classifying, and recognizing vehicle license plates (LP) specifically tailored for the Nepalese context. Leveraging the YOLO8 object detection model from the Ultralytics library, the method detects vehicles and their license plates in video footage and Image. The chosen method enables robust detection and classification of vehicles, addressing the unique challenges posed by the Nepalese landscape. Subsequently, the system employs the SORT algorithm for real-time tracking of identified vehicles across frames, ensuring accurate monitoring and surveillance. Following detection, the license plates are cropped and processed to extract the text, facilitating efficient data retrieval and analysis. Moreover, to tackle the intricacies of Devanagari characters, Yolov8 incoperated Convolutional Neural Network (CNN) model is integrated, significantly enhancing recognition accuracy with 91% within the Nepalese context. This comprehensive approach not only advances the field of license plate detection but also offers practical solutions for improving transportation management and law enforcement in Nepal.

Keywords:

license plate detection,YOLO8, SORT algorithm, Convolutional Neural Network (CNN), Devanagari character recognition

Contents

List of Figures

List of Tables

1 Introduction

The increasing reliance on vehicles for transportation across various sectors has sparked discussions worldwide about the necessity of an efficient system capable of automatically detecting and reading license plates. This demand stems from the pivotal role such technology plays in law enforcement, traffic management, and the development of smart cities and autonomous vehicle systems. Addressing this need, the Automatic License Plate Detection and Recognition (ALPDR) system has emerged, employing advanced technologies like machine learning to streamline the process. Initially, the system utilizes a camera to capture images of license plates, which are then subjected to machine learning algorithms for character extraction and recognition. Before dissecting the characters, ALPDR systems often classify license plates, aiding in the organization of subsequent tasks. Notably, the Nepalese context presents unique challenges due to the variability in license plate formats, including one-row, two-row, and three-row structures, each with specific character arrangements. Additionally, the incorporation of Devanagari script adds complexity, with varying writing styles posing readability challenges. To address these intricacies, a comprehensive approach is essential, encompassing detection, classification, and recognition tailored to the Nepalese license plate context. Through the analysis of elements such as color, font styles, character arrangement, and identifying information, valuable insights into vehicle ownership, registration, and other pertinent details can be derived, facilitating effective transportation management and law enforcement in Nepal.The ALPDR system, utilizing machine learning and advanced algorithms, addresses the complexities of license plate detection and recognition in Nepal. With its ability to classify plates and decipher Devanagari script, it provides essential insights into vehicle ownership and registration. By leveraging distinctive characteristics like color, font styles, and character arrangements, the system enhances law enforcement and transportation planning efficiency, promising safer and smarter urban mobility.

1.1 Problem Statement

Recognizing Nepali license plates manually presents significant challenges due to the complexities of the Devanagari script and the wide variety of plate formats. Current methods often fail to detect and interpret these plates accurately and quickly, impacting law enforcement and Vehicle tracking in Nepal. This report aims to address these issues by developing an automated solution using advanced deep learning techniques to reliably and swiftly identify Devanagari characters on Nepali license plates. The proposed system seeks to enhance both the precision and efficiency of license plate recognition in real-time scenarios, providing a more effective tool for vehicle tracking system and enforcement.

1.2 Project Objectives

After a certain investigation in the current situation, the objective of our project has been listed. The main objectives has been given below:

1. To develop a web application to address challenges in recognizing and detecting variations in Devanagari license plates.

1.3 Significance of the Study

The study presents a significant advancement in Nepal's transportation management and law enforcement by focusing on an automated license plate recognition system, particularly for Devanagari-script plates. This innovation promises enhanced monitoring and control of vehicles, thereby improving safety on the streets. Moreover, it streamlines vehicle registration processes, reducing wait times at government offices. The accurate data generated by this technology aids in better transportation planning, crucial for evolving urban and rural landscapes. By spearheading such advancements, Nepal stands to become a regional leader in transportation system management, offering valuable insights for countries facing similar challenges in the region.

1.4 Scope and Limitations

Based on our observations, our project has the potential to contribute to the following ways:

1.4.1 Scope

1. Enhancing law enforcement capabilities and Parking Management System
2. Improving transportation planning through data insights.
3. Supporting revenue collection by tracking vehicles for taxation

1.4.2 Limitations:

1. Confusion between similar-looking characters.
2. Glare, shadows, and poor lighting affect detection and OCR accuracy.
3. Environmental factors affecting system performance.
4. Difficulty in accurately locating the plate in crowded vehicle flow

2 Literature Review

License plate detection, classification, and recognition have garnered significant attention in recent years due to their crucial role in various applications, including law enforcement, traffic management, and automated toll collection systems. In their paper titled "Devanagari License Plate Detection, Classification and Recognition," Dawadi, Bal, and Pokharel present a comprehensive study on advancing license plate recognition technology, specifically tailored for the Nepalese context. Previous research in this field has primarily focused on license plate recognition systems for languages using Latin characters, with limited attention given to scripts like Devanagari. However, the unique characteristics of Devanagari script pose distinct challenges, including complex character shapes and variations in writing styles, necessitating specialized approaches for accurate recognition. Dawadi et al. build upon existing literature by proposing an innovative approach that combines machine learning techniques with advanced algorithms for license plate detection and recognition. Leveraging the YOLO8 object detection model and Convolutional Neural Networks (CNNs), their methodology enables robust detection of vehicles and license plates in video footage, while also addressing challenges associated with Devanagari script recognition. The authors also integrate the SORT algorithm for real-time tracking of vehicles across frames, enhancing the system's capability for surveillance and monitoring applications. Additionally, they discuss the significance of license plate classification in organizing the recognition process and improving overall system efficiency. Furthermore, the study explores the structural aspects of Nepalese license plates, categorizing them into various formats based on ownership categories and vehicle types. By analyzing elements such as color, font styles, and character arrangements, the authors provide valuable insights into license plate characteristics, aiding in the development of accurate recognition models tailored to the Nepalese context. Overall, Dawadi et al.'s research contributes significantly to the field of license plate recognition by addressing the challenges specific to Devanagari script and providing practical solutions for enhancing system performance in the Nepalese context. Their methodology lays the groundwork for future advancements in license plate detection and recognition technologies, with implications for improving transportation management and law enforcement systems globally[1].

In response to the escalating demand for enhanced security and efficient transporta-

tion, there has been a surge in the development of automated vehicle tracking systems leveraging advancements in technology. With the exponential growth of vehicles on roads over the past decade, the task of individual vehicle tracking has become increasingly challenging. This challenge is compounded by the cumbersome process of obtaining real-time CCTV footage for surveillance purposes. To address this issue, researchers have turned to deep learning models, such as You Only Look Once (YOLO), for efficient object detection. The paper by Gnanaprakash et al. proposes an automated vehicle tracking system utilizing surveillance cameras installed along roadways. The system comprises four main steps: first, video footage is converted into images, followed by the detection of vehicles in each frame using YOLO. Next, license plates are detected from the identified vehicles, and finally, characters from the license plates are recognized. The authors employ the ImageAI library to streamline the training process of the deep learning model. To evaluate the performance of their proposed model, Gnanaprakash et al. analyze the accuracy achieved in car detection, license plate localization, and character recognition using Tamil Nadu license plate images. Notably, the results demonstrate impressive accuracy rates, with a 97% accuracy achieved for car detection, 98% for license plate localization, and 90% for character recognition. This study contributes to the existing literature by presenting a comprehensive approach to automatic number plate recognition using deep learning techniques. By addressing the challenges associated with vehicle tracking in real-time surveillance scenarios, the proposed system offers a promising solution for enhancing security and traffic management in urban environments. The achieved accuracy rates underscore the efficacy of the deep learning model in accurately detecting and recognizing vehicles and license plates, paving the way for further advancements in automated surveillance systems[2].

In response to the challenges posed by difficult situations such as distorted views, varying lighting conditions, and dusty environments, researchers have sought innovative solutions for efficient number plate identification in vehicles. Addressing this need, the paper proposes the utilization of Faster R-CNN, a state-of-the-art object detection model, to detect number plates from surveillance camera footage placed in traffic areas. The study emphasizes the importance of accurate number plate detection for traffic monitoring and law enforcement purposes. By capturing vehicle videos and employing frame segmentation and image interpolation techniques, the proposed system enhances

the accuracy of number plate detection. Furthermore, the application of optical character recognition (OCR) on the resulting images facilitates number recognition, enabling the retrieval of crucial vehicle information from a database.One notable aspect of the proposed system is its high accuracy rate, achieving an impressive 99.1% accuracy in number plate detection. This exceptional performance underscores the effectiveness of the Faster R-CNN model and the integrated image processing techniques in overcoming challenging conditions encountered in real-world surveillance scenarios.While the paper provides valuable insights into the development and performance evaluation of the proposed system, further research could explore additional optimizations and enhancements to address potential limitations and improve scalability. Additionally, future studies may focus on evaluating the system's performance in diverse environmental conditions and real-world deployment scenarios to validate its effectiveness and practical utility. Overall, the research contributes to the advancement of automated surveillance systems for vehicle monitoring and identification, offering a promising solution for improving traffic management and law enforcement efforts. The demonstrated accuracy and efficiency of the proposed system underscore its potential for widespread adoption in various urban and transportation contexts[3].

The Kaggle dataset "Vehicle Number Plate Dataset (Nepal)" stands as a pivotal resource for researchers and developers delving into the realm of Automatic License Plate Detection and Recognition (ALPDR) systems, particularly within the unique context of Nepal. This repository, hosted on the Kaggle platform, presents an extensive collection of vehicle number plate images specifically sourced from Nepal, offering an invaluable corpus for training, validation, and benchmarking ALPDR algorithms. Its inclusion of a diverse array of images encapsulates the variations inherent in Nepalese license plate formats, encompassing the intricacies of Devanagari script, which is predominant in the region's license plates. By providing such a comprehensive dataset, researchers can effectively address the multifaceted challenges associated with Devanagari script recognition, thus fortifying the robustness and efficacy of ALPDR systems operating in real-world Nepalese environments. Furthermore, the dataset's accessibility on Kaggle fosters a collaborative ecosystem wherein researchers can share methodologies, insights, and best practices, thereby catalyzing innovation and progress within the field.Moreover, this repository not only serves as a foundational resource for academic

research but also holds practical significance for industries and governmental bodies involved in traffic management, law enforcement, and urban planning. The availability of high-quality, annotated data empowers stakeholders to develop and deploy ALPDR systems that enhance road safety, streamline traffic flow, and bolster security measures. In essence, the Kaggle "Vehicle Number Plate Dataset (Nepal)" emerges as a cornerstone in the advancement of ALPDR technologies, facilitating tailored solutions that cater to the unique demands and nuances of Nepal's license plate recognition landscape[4].

The research paper titled "License Plate Detection and Recognition Using Deeply Learned Convolutional Neural Networks" by Masood et al. (2017) presents a significant contribution to the field of automatic license plate detection and recognition systems. In recent years, the development of such systems has gained considerable attention due to their wide range of applications, including traffic management, law enforcement, and vehicle tracking. This paper addresses the challenge of accurately detecting and recognizing license plates from complex scenes, leveraging the power of deep learning, specifically Convolutional Neural Networks (CNNs). The authors begin by highlighting the importance of license plate detection and recognition in various real-world scenarios and discuss the limitations of traditional methods, which often struggle with variations in illumination, scale, orientation, and occlusion. In response to these challenges, Masood et al. propose a novel approach that utilizes CNNs to automatically learn discriminative features directly from raw image data, thus eliminating the need for handcrafted features and complex preprocessing steps. The paper outlines the architecture of the proposed system, which consists of multiple convolutional layers followed by fully connected layers, trained using a large dataset of labeled license plate images. The CNN model is trained in a supervised manner, where the objective is to minimize the classification error between predicted and ground-truth license plate regions. The authors employ data augmentation techniques to increase the robustness of the model and prevent overfitting. To evaluate the performance of their approach, Masood et al. conduct extensive experiments on benchmark datasets and compare their results with state-of-the-art methods. The experimental results demonstrate the effectiveness of the proposed CNN-based approach, achieving superior performance in terms of detection accuracy and recognition rates, even under challenging conditions such as low-resolution images and partial occlusion. Overall, the paper contributes to the ad-

vancement of automatic license plate detection and recognition systems by introducing a deep learning-based approach that surpasses traditional methods in terms of accuracy and robustness. The proposed CNN architecture offers a promising solution for real-world applications where accurate and efficient license plate recognition is essential. However, the authors acknowledge potential areas for future research, including further optimization of the CNN model and exploration of advanced techniques for post-processing and refinement of detected license plate regions[5].

3 Methodology

This methodology outlines the development of a deep learning model for Nepali license plate detection and recognition within the Vehicle Tracking System. It involves four phases: image preprocessing, license plate detection using YOLO models, character detection through adaptive thresholding, and character recognition via a convolutional neural network. The model's effectiveness is evaluated using precision, recall, average precision, and mean average precision. The agile software development model ensures thorough analysis, design, development, testing, and implementation of each module.

3.1 Software Process model

Agile's iterative nature allows for changes to be incorporated at any stage of the project. This is crucial in environments where requirements may evolve based on market conditions, customer feedback, or technological advancements. By breaking the project into smaller, manageable iterations (called sprints), our team can adapt to changes quickly and efficiently.

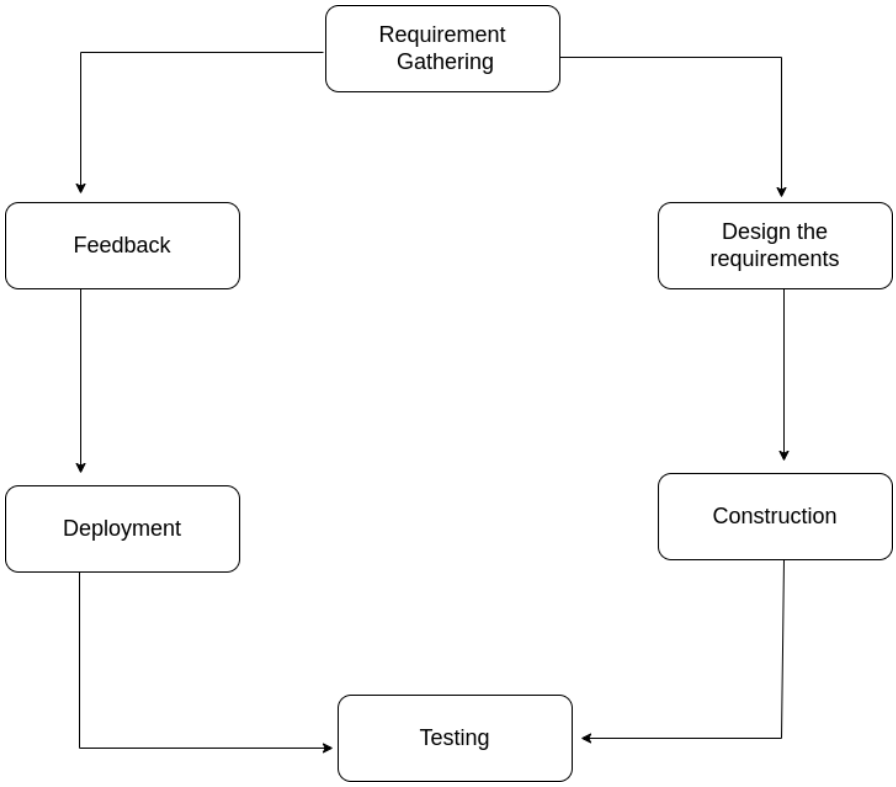

Figure 1: Agile Software Process Model

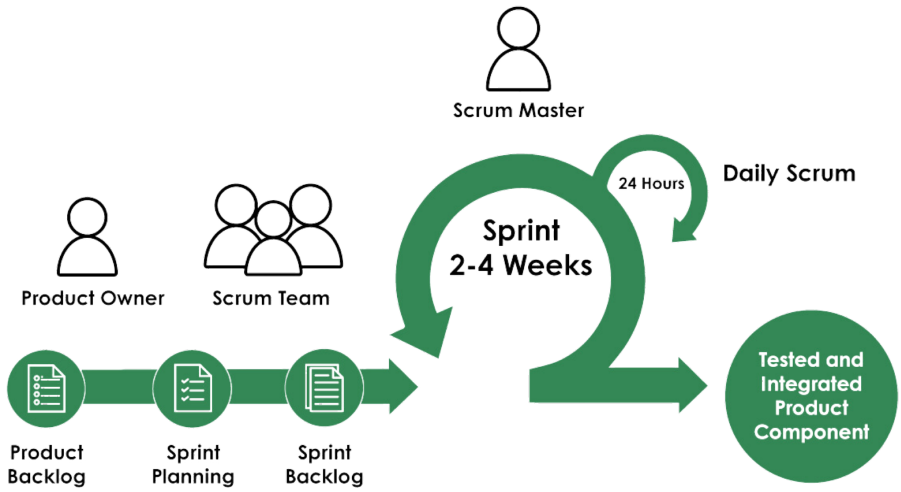

Figure 2: Agile software development with Scrum [6]

The agile model includes the following:

- Requirement Gathering: In Agile, requirements gathering is a continuous process involving user stories that describe features from the user's perspective. The product backlog is regularly updated based on stakeholder input and changing priorities, ensuring flexibility and responsiveness throughout the project. understood.

- Design the requirements: Design in Agile focuses on creating adaptable, high-level designs sufficient to start development. Emphasis is placed on collaborative design sessions to ensure the architecture is practical, scalable, and capable of evolving as requirements change.

- Construction: Construction in Agile happens in short iterations, typically two to four weeks, where the team develops, integrates, and tests small increments of the software. Continuous integration and pair programming help maintain code quality and facilitate early detection of issues.

- Testing:Testing is integrated into the Agile development process, with automated tests and test-driven development ensuring defects are caught early. This continuous testing approach maintains high-quality standards and prevents new code from breaking existing functionality.

- Deployment: Agile deployment emphasizes frequent delivery of working software, often at the end of each iteration. Continuous deployment automates releases once tests pass, allowing incremental feature rollouts and leveraging DevOps practices for efficient and reliable deployment.

- Feedback: Feedback is essential in Agile, with sprint reviews and retrospectives at the end of each iteration to gather stakeholder input and improve processes. Continuous customer feedback ensures the product meets user needs and adapts to changing requirements.

3.2 Significance of Agile Model

The Agile methodology provided significant benefits throughout the project. Its flexibility and adaptability were crucial, allowing us to adjust to changes and new requirements, particularly during data preprocessing and model tuning stages where iterative improvements were needed to enhance the model's performance. By dividing the project into three sprints, we were able to focus on manageable segments of work, ensuring continuous evaluation and refinement. This iterative approach led to a robust and reliable final product. The Agile model also facilitated early and continuous delivery of functional components, helping to identify potential issues early and allowing for timely adjustments.

Regular sprint reviews and updates kept stakeholders informed, aligning the project with their expectations and requirements, resulting in a product that met their needs effectively. Additionally, the iterative nature of Agile, combined with regular testing and feedback loops, contributed to improved quality, with each sprint including evaluation phases where models were tested and tuned, leading to higher accuracy in license plate detection and character recognition. Agile's incremental approach helped in identifying and mitigating risks early in the project, reducing the risk of major problems in later stages. The methodology encouraged collaboration among team members, fostering a culture of continuous improvement and shared responsibility. This collaborative environment was crucial for the complex tasks of model training and evaluation. Finally, Agile's emphasis on transparency and visibility allowed for clear tracking of progress through sprint tasks and timelines, ensuring that all team members and stakeholders had a clear understanding of the work completed and what was ahead.

3.3 WorkFLow

We adopted the Agile methodology to ensure flexibility and iterative development. The project was divided into three main sprints:

- **Sprint 1**: Planning, Requirements Gathering, Data Preprocessing
- **Sprint 2**: Model Training, Model Tuning
- **Sprint 3**: Model Evaluation, Deployment

3.3.1 Sprint 1: Planning, Requirements Gathering, Data Preprocessing

1. **Planning And Comminication**:

   Initial project planning involved defining the project scope and setting up the team role.

   Project Title: License Plate Recognition System

   Objective:

   (a) To analyze challenges focusing on Devanagari License Plates variations.

   (b) To develop web application

   Scope

   (a) Enhancing law enforcement capabilities.

   (b) Streamlining vehicle registration processes.

   (c) Improving transportation planning through data insights.

   (d) Supporting revenue collection by tracking vehicles for taxation.

   (e) Vehicle Theft Detection

   Risks:

   Technical Risks

   (a) Failure of the vehicle detection algorithm under different weather conditions.

   (b) Inaccurate vehicle counting due to camera malfunctions.

(c) Integration issues with existing traffic management systems.

Operational Risks

(a) Delays in hardware procurement or installation.

(b) Insufficient data storage or processing capacity.

Team Roles

- **Front-End/ Backend Developer:** Priyanka Sinha
- **Machine Learning(Object-Detection, OCR, Character Recognition):** Saishna Budhathoki
- **Vehicle-Counting(Machine Learning Application), Deployment:** Isha Hitang
- **Documentation:** Saishna Budhathoki, Priyanka Sinha, Isha Hitang

2. **Requirements Gathering**

For the development of the license plate detection system, we will undertake the process of identifying and documenting all necessary components and needs. Here's how we will address the requirements:

(a) **Hardware Requirements**

i. Cameras

- We will select high-resolution cameras to capture clear images of license plates. These cameras will be chosen to ensure that the characters on the plates are visible and readable.

ii. Processing Unit

- We will utilize a powerful computer equipped with a high-performance CPU, ample RAM, and a GPU to accelerate the machine learning algorithms and handle image processing tasks effectively.

(b) **Software Requirements**

i. Detection Algorithms

- We will integrate detection software, such as YOLOv8, to identify and locate license plates, two-wheelers, and four-wheelers in the images. This software will be crucial for the effective detection of vehicles and plates.

ii. OCR Tools

- We will utilize Optical Character Recognition (OCR) software, specifically the YOLOv8 model and Tesseract OCR, to extract text from the detected license plates. The OCR system will be configured to handle the specific font and style of the plates being processed.

iii. Integration Software

- We will develop and implement software to integrate the detection and OCR components to count license plates.

(c) **Data Needs**

i. Training Data

- We will collect a diverse dataset of license plate images that includes various conditions such as different weather, lighting, and angles. This dataset will encompass a range of license plate designs and formats to train the detection and OCR models effectively.

ii. Testing Data

- A separate set of images will be prepared for testing the performance of the models, ensuring they can generalize well to new and unseen data.

iii. Annotation Tools

- We will use annotation tools to label images with bounding boxes and text, preparing the data for training.

3.3.2 Sprint 2: Model Training, Model Tuning

1. **Model Training**

   Training the YOLOv8 model on the prepared dataset to detect license plates, 2-wheelers, and 4-wheelers. Training a OCR for character recognition.

2. **Model Tuning** Fine-tuning the model parameters to improve accuracy and performance.

Model Testing

Unit Testing

Objective: Validate individual components and functions. Components include image capture and upload functionalities, YOLOv8 license plate detection, the SORT algorithm for tracking, and the OCR module for text extraction. **Tools:** pytest, unittest.

Integration Testing

Objective: Ensure modules work together seamlessly. Components include the end-to-end workflow from image capture to text recognition, interaction between YOLOv8, SORT, and OCR modules, and data flow from upload to storage and display. **Tools:** pytest

System Testing

Objective: Validate the entire system against requirements. Scenarios include real-time video stream processing, handling various license plate formats and conditions, and user interface interaction and report generation.

3.3.3 Sprint 3: Model Evaluation, Deployment

1. **Model Evaluation**

   Comparing the accuracy of the YOLOv8 and OCR models. Creating a table to present the comparison results.

2. **Deployment** Deploying the trained model for real-time license plate detection and character recognition. Ensuring the output dimensions of videos are consistent.

3.4 License Plate Detection and Recognition

License plate detection and recognition is a crucial component of the Car Park Agent in our proposed system. We herein describe in detail the steps involved in building the deep learning model for detecting and recognizing Nepali license plates in car parks. As illustrated in Figure 2, the model consists of four phases: (a) image acquisition and preprocessing, (b) license plate detection, (c) character detection, and (d) character recognition. We will discuss each of these phases in detail in the following subsections.

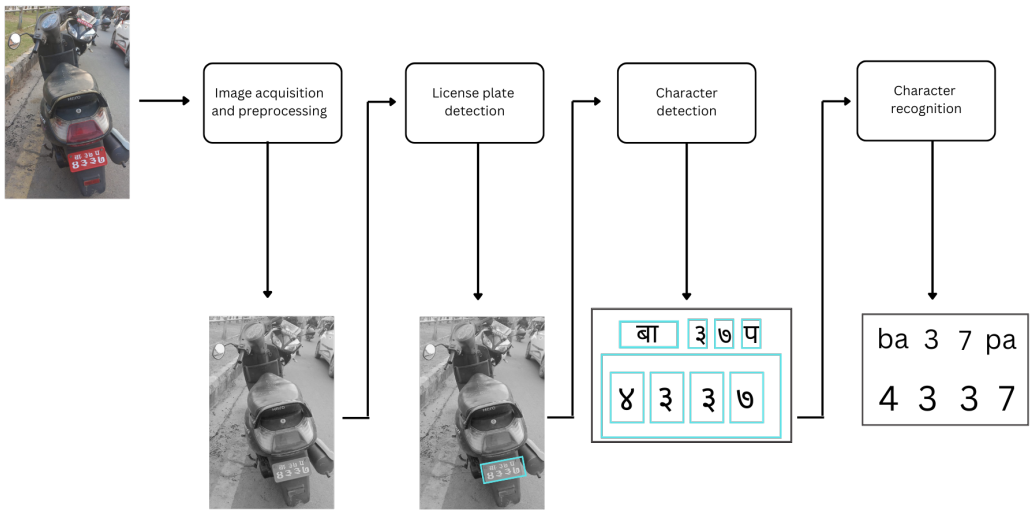

Figure 3: Phases of License Plate Detection and Recognition

3.4.1 Image Acquisition and Preprocessing

In the initial phase of our deep learning model aimed at license plate detection and recognition, image acquisition and preprocessing are pivotal. This phase involves applying various filters to the image to eliminate distortions and enhance quality, crucial for object detection. Our approach involves resizing images to 640x640 pixels and converting them to grayscale using OpenCV-Python library methods. These dimensions are aligned with YOLO models trained on the COCO dataset, facilitating transfer learning in subsequent phases. Additional preprocessing steps include denoising using OpenCV-Python's fastNlMeansDenoising method and brightness enhancement through PyTorch's ColorJitter. Augmentation techniques are employed on the training set to simulate real-world scenarios and enhance model generalization, while the

validation and test sets remain unaltered to ensure accurate performance evaluation. Further details on augmentation techniques will be discussed during the presentation of the training process for the proposed models

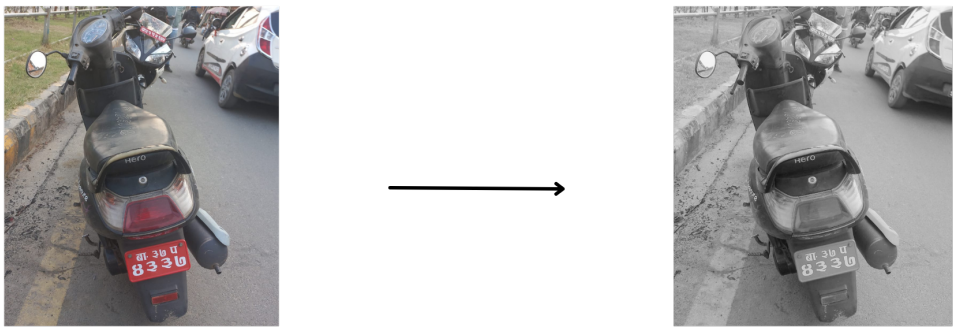

Figure 4: Preprocessing phase

3.4.2 License Plate Detection

In the license plate detection phase of our deep learning model, the primary goal is to locate and extract the license plate from an image of a car. This involves searching the car image for the rectangle containing the license plate. We train and fine-tune YOLOv5x, YOLOv7x, and YOLOv8x models on a dataset of Nepali license plates that we collected ourselves, which will be elaborated on later. The model takes a car image as input and produces the detected license plate as output.

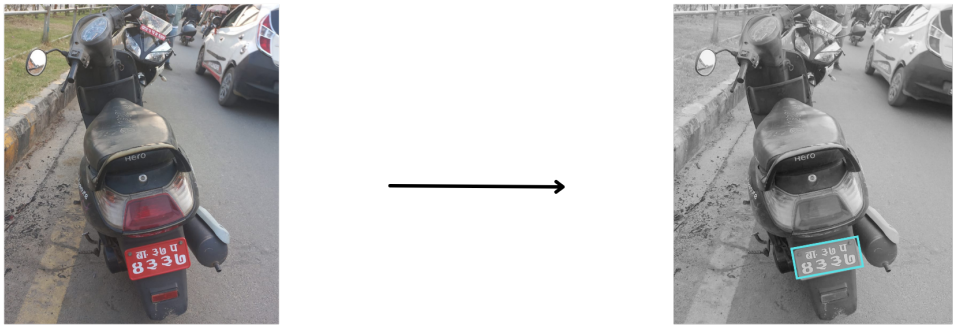

Figure 5: License Plate Detection Phase

3.4.3 Character Detection

Character detection involves breaking down an image containing a sequence of characters, such as a license plate, into individual symbols. We employ adaptive thresholding to convert the license plate image into a binary format, enhancing the detection process by adjusting the threshold value based on local pixel neighborhoods. OpenCV's cv2.adaptiveThreshold function is utilized with parameters specifying mean-based thresholding, binary comparison, neighborhood size, and a constant for threshold fine-tuning. After binarization, unnecessary characters like characters, Hindi digits, and plate boundaries are removed. Large bounding boxes are then extracted for English letters and digits, within which smaller bounding boxes for individual characters are identified. Models, including YOLOv5x, YOLOv7x, and YOLOv8x, are trained and

fine-tuned using a manually annotated dataset to detect these bounding boxes. The final output comprises localized English letters and Nepali digits(devanagari), representing the regions of interest within the detected license plate.

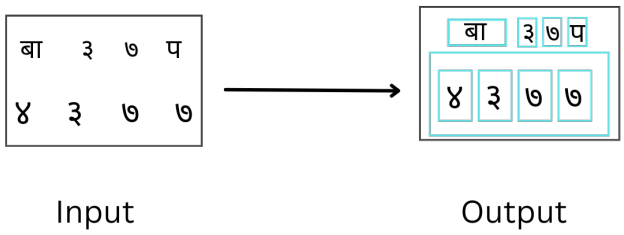

Figure 6: Character Detection Phase

3.4.4 Character Recognition

Character recognition involves identifying detected characters and matching them with their respective devanagari letters and Nepali digits. A CNN framework is developed for this purpose, comprising four convolutional layers and two dense layers. The first convolutional layer applies 64 filters of size 4x4 with ReLU activation to input images of shape 50x50x3, followed by max pooling, normalization, and dropout layers to prevent overfitting. The subsequent convolutional layers maintain the same structure, with the fourth layer employing 32 filters of size 2x2. The output is flattened into a 1D vector and processed by a dense layer with 64 units and ReLU activation, followed by dropout. A final dense layer with 27 units and softmax activation generates a probability distribution over 27 classes. The model contains 163,263 parameters, primarily trainable, with nontrainable parameters in normalization layers.

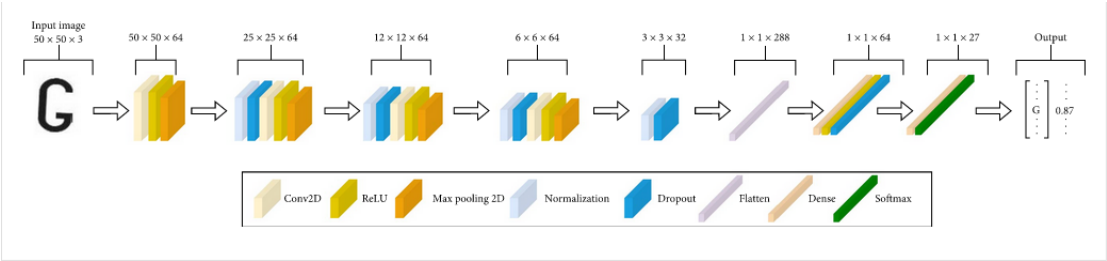

Figure 7: Character Recognition Model Architecture [7]

3.5 System Design and Object Oriented Architecture

Below is the Usecase Diagram illustrating the use of license plate detection for Vehicle Tracking System

Usecase Diagram

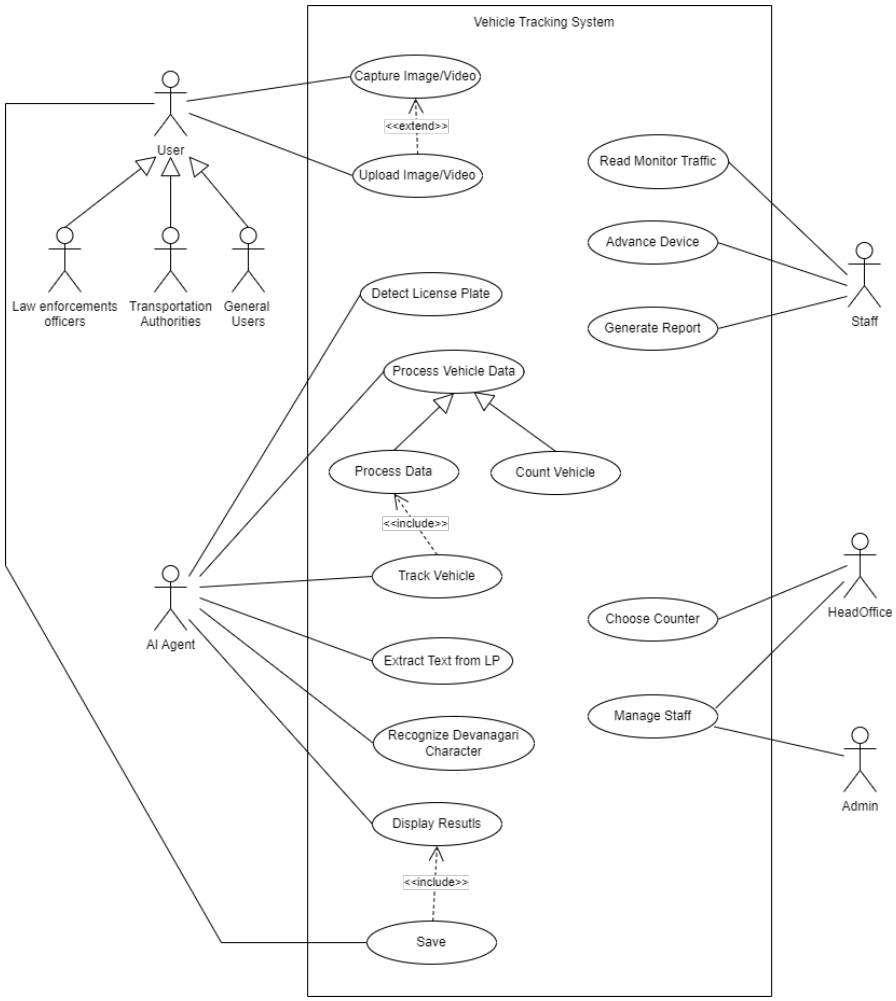

Figure 8: Usecase Diagram for Vehicle Tracking System

Use Cases for Each Role

1. **Staff**

   (a) **Read Monitor Traffic**: Access and monitor real-time traffic data via the Streamlit interface.

(b) **Advance Device**: Use advanced cameras to capture high-quality images and videos of vehicles.

(c) **Create Report**: Generate reports based on stored traffic data.

2. **Head Office**

(a) **Choose Counter**: Select specific checkpoints for focused monitoring.

(b) **Manage Staff**: Oversee and manage staff members operating the ALPDR system.

3. **Admin**

(a) **Manage Staff**: Add/remove users, assign roles, and ensure proper training.

(b) **Generate Report**: Compile and review comprehensive traffic reports.

(c) **Manage Tracking Vehicles**: Oversee accurate vehicle tracking using the SORT algorithm.

Actors and Use Cases

1. **User (Law enforcement officers, Transportation authorities, General users)**

(a) **Capture Image/Video**: Capture images or videos of vehicles.

(b) **Upload Image/Video**: Upload media to the ALPDR system via Streamlit.

2. **ALPDR System**

(a) **Detect License Plate**: Use YOLOv8 to detect license plates.

(b) **Track Vehicle**: Use SORT to track vehicles across frames.

(c) **Extract Text from License Plate**: Extract text using OCR.

(d) **Recognize Devanagari Characters**: Recognize text using the CNN+YOLO model.

(e) **Display Results**: Show recognized text to the user.

(f) **Store Data**: Save processed data in the database.

Example Scenario

A law enforcement officer captures a video of traffic and uploads it to the ALPDR system. The system detects license plates, tracks vehicles, extracts text, and recognizes Devanagari characters. Staff monitor traffic and generate reports, the head office selects checkpoints and manages staff, and admins oversee the entire process, ensuring accurate tracking and comprehensive reporting.

3.6 Sequence Diagram

Below is the Sequence Diagram illustrating the use of license plate detection for Vehicle Parking System

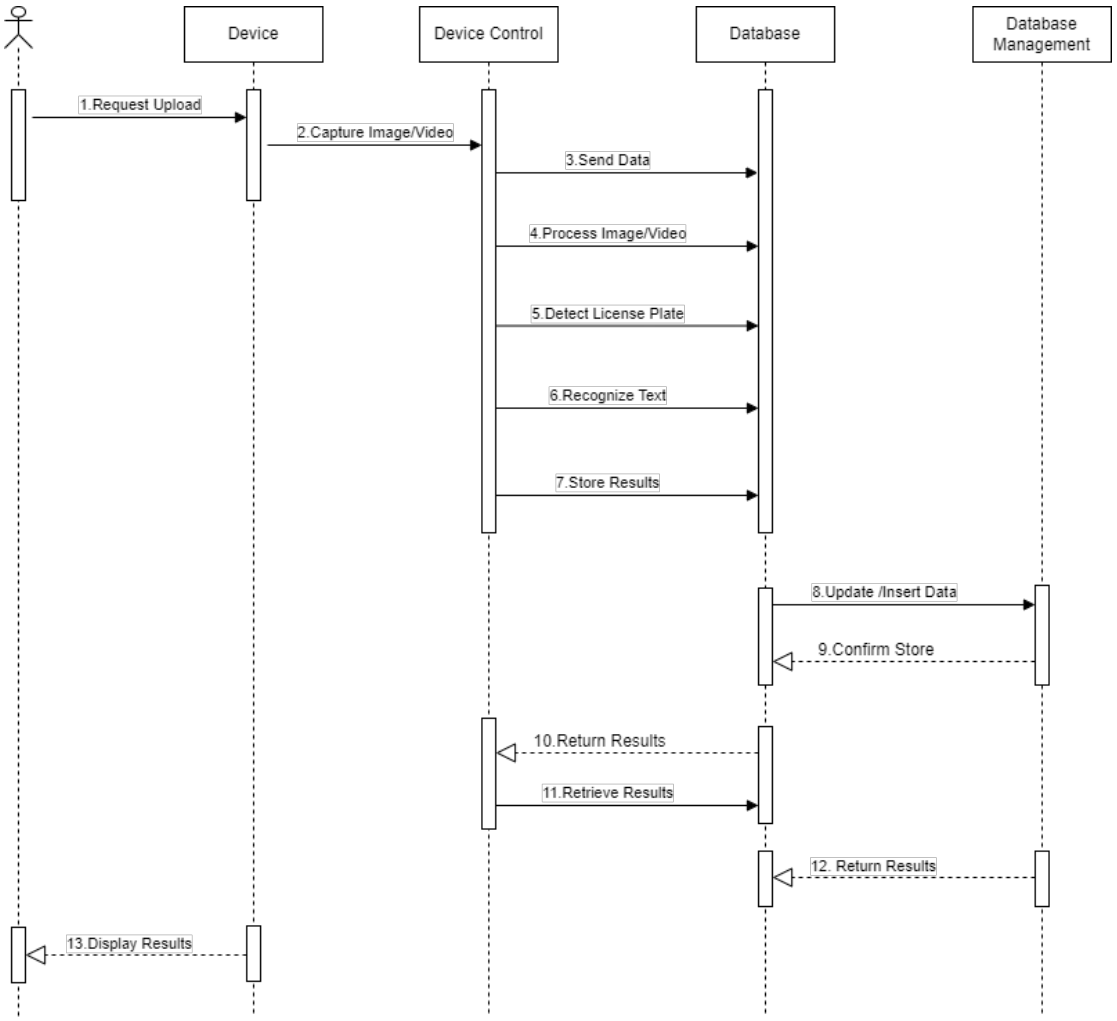

Figure 9: Sequence Diagram for Vehicle Tracking System

3.7 Activity Diagram

Below is the Activity Diagram illustrating the use of license plate detection for Vehicle Tracking System

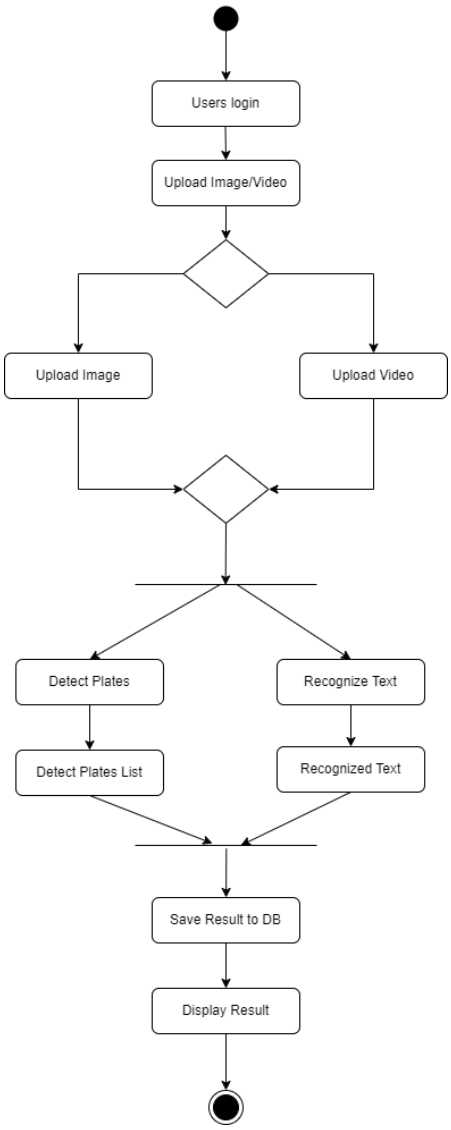

Figure 10: Activity Diagram for Vehicle Tracking System

3.8 ER Diagram

Below is the ER Diagram illustrating the use of license plate detection for Vehicle Tracking System

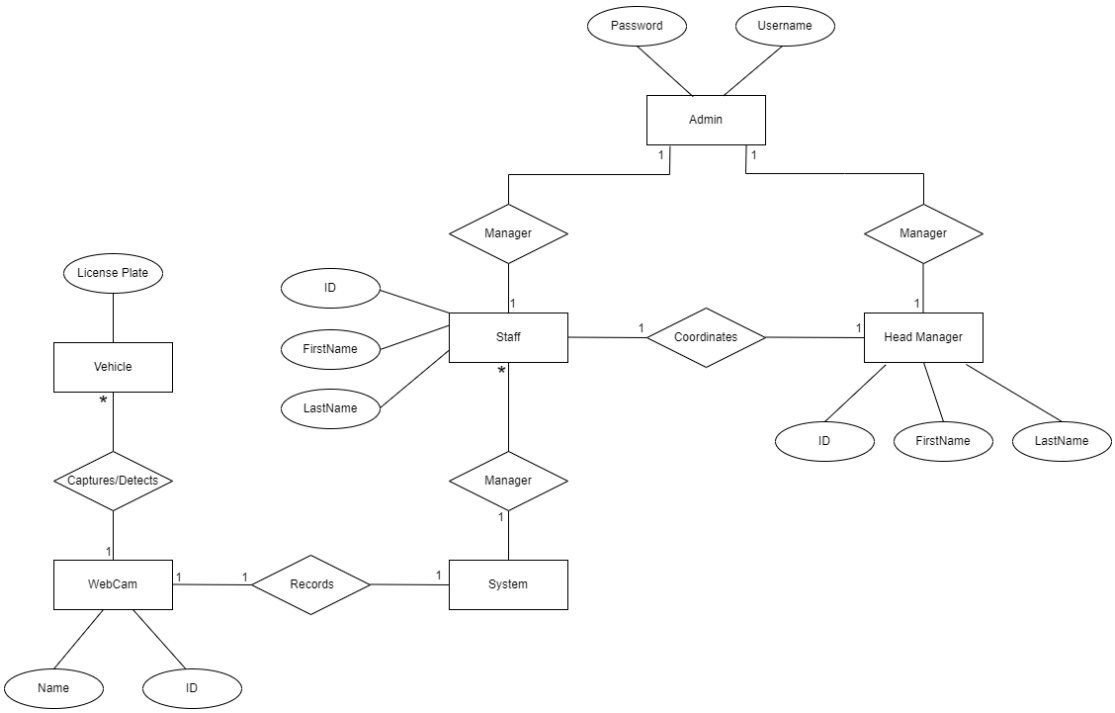

Figure 11: ER Diagram for Vehicle Tracking System

3.9 Class Diagram

Below is the Class Diagram illustrating the use of license plate detection for Vehicle Tracking System

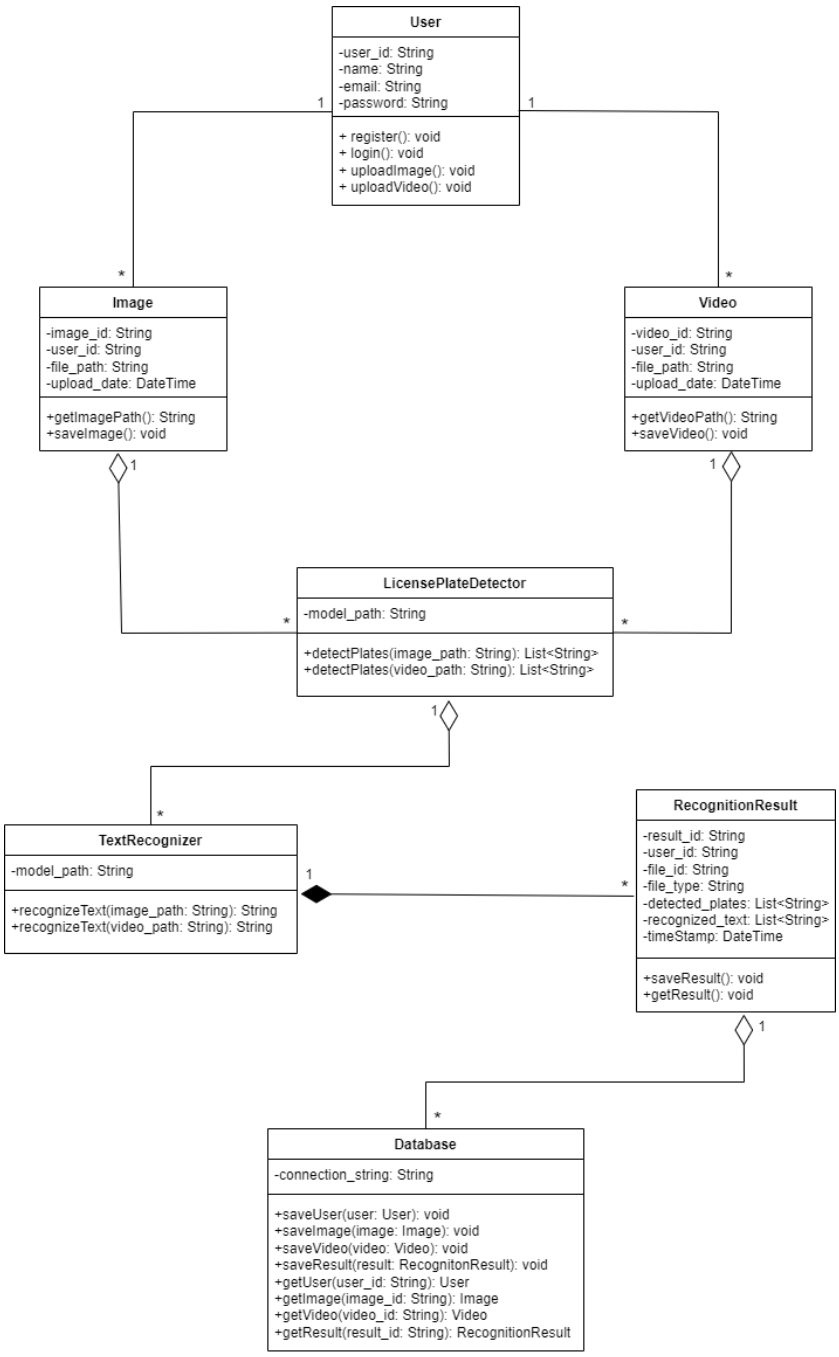

Figure 12: Class Diagram for Vehicle Tracking System

3.10 Component Diagram

Below is the Component Diagram illustrating the use of license plate detection for Vehicle Tracking System

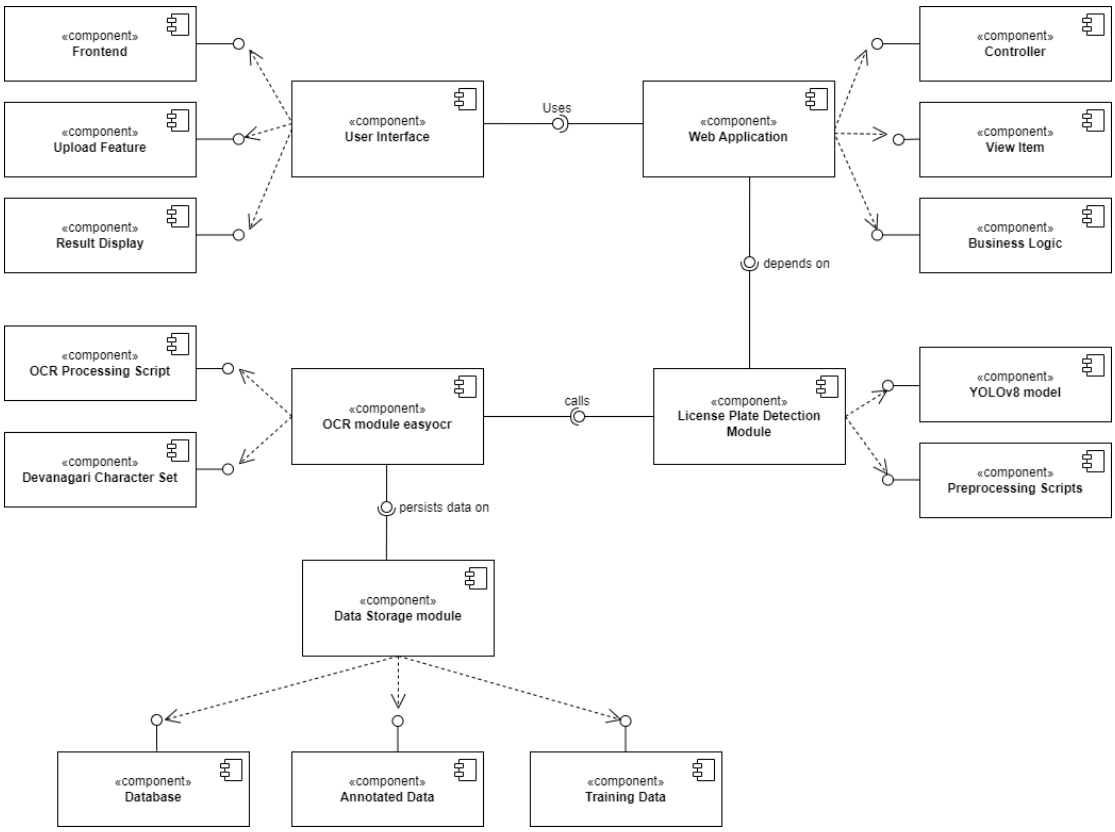

Figure 13: Component Diagram for Vehicle Tracking System

3.11 Deployment Diagram

Below is the Deployment Diagram illustrating the use of license plate detection for Vehicle Tracking System

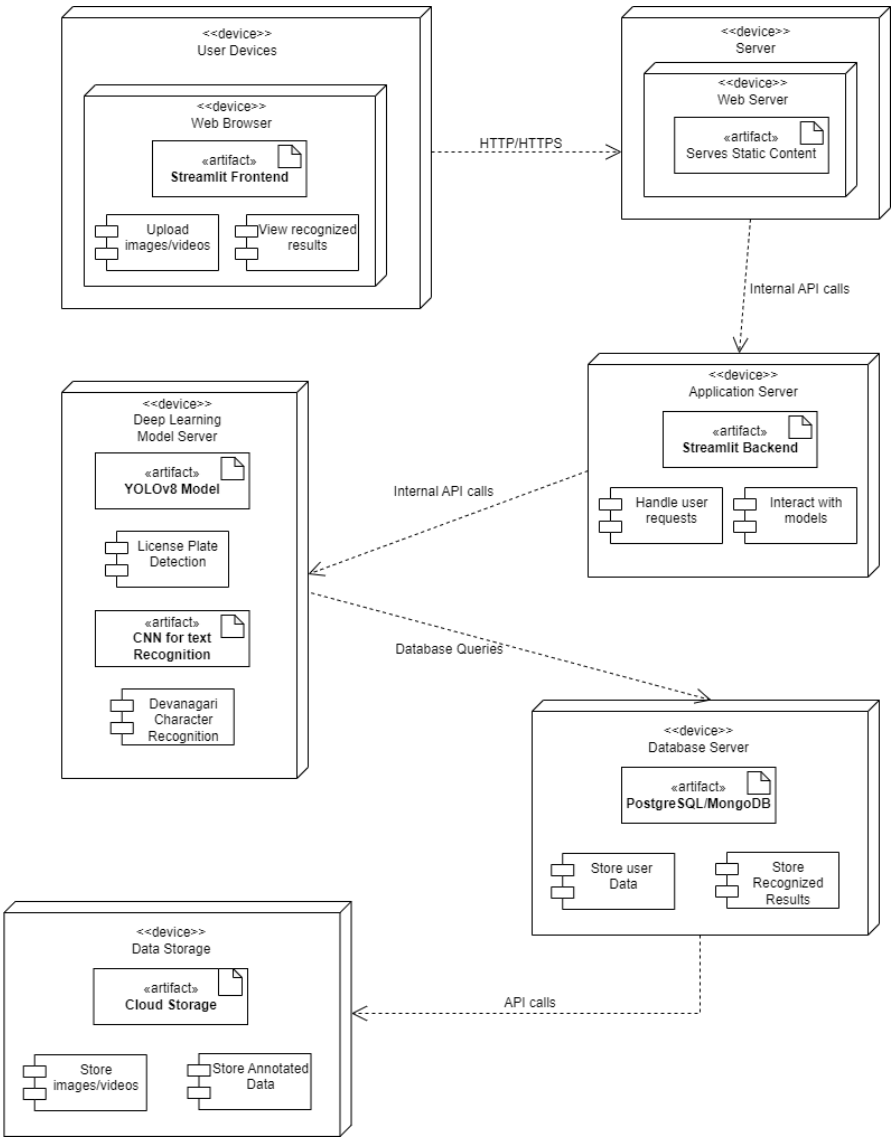

Figure 14: Deployment Diagram for Vehicle Tracking System

3.12 Collaboration Diagram

Below is the Collaboration Diagram illustrating the use of license plate detection for Vehicle Tracking System

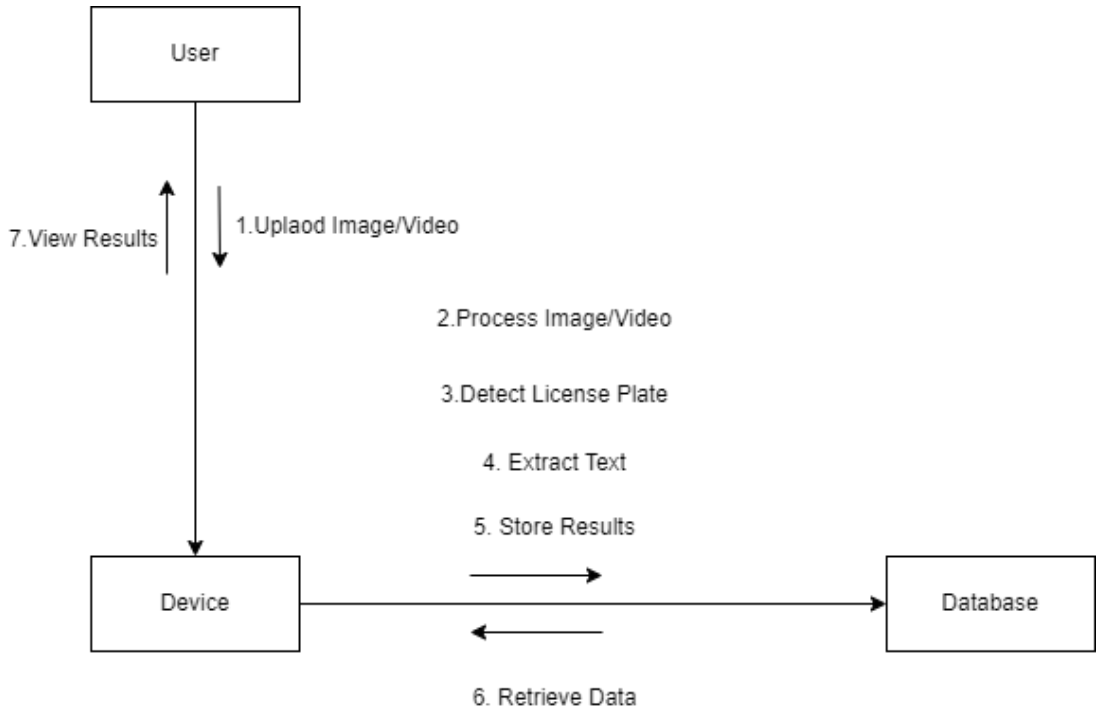

Figure 15: Collaboration Diagram for Vehicle Tracking System

3.13 System Sequence Diagram

Below is the System Sequence Diagram illustrating the use of license plate detection for Vehicle Tracking System

3.13.1 System Sequence diagram between User and System

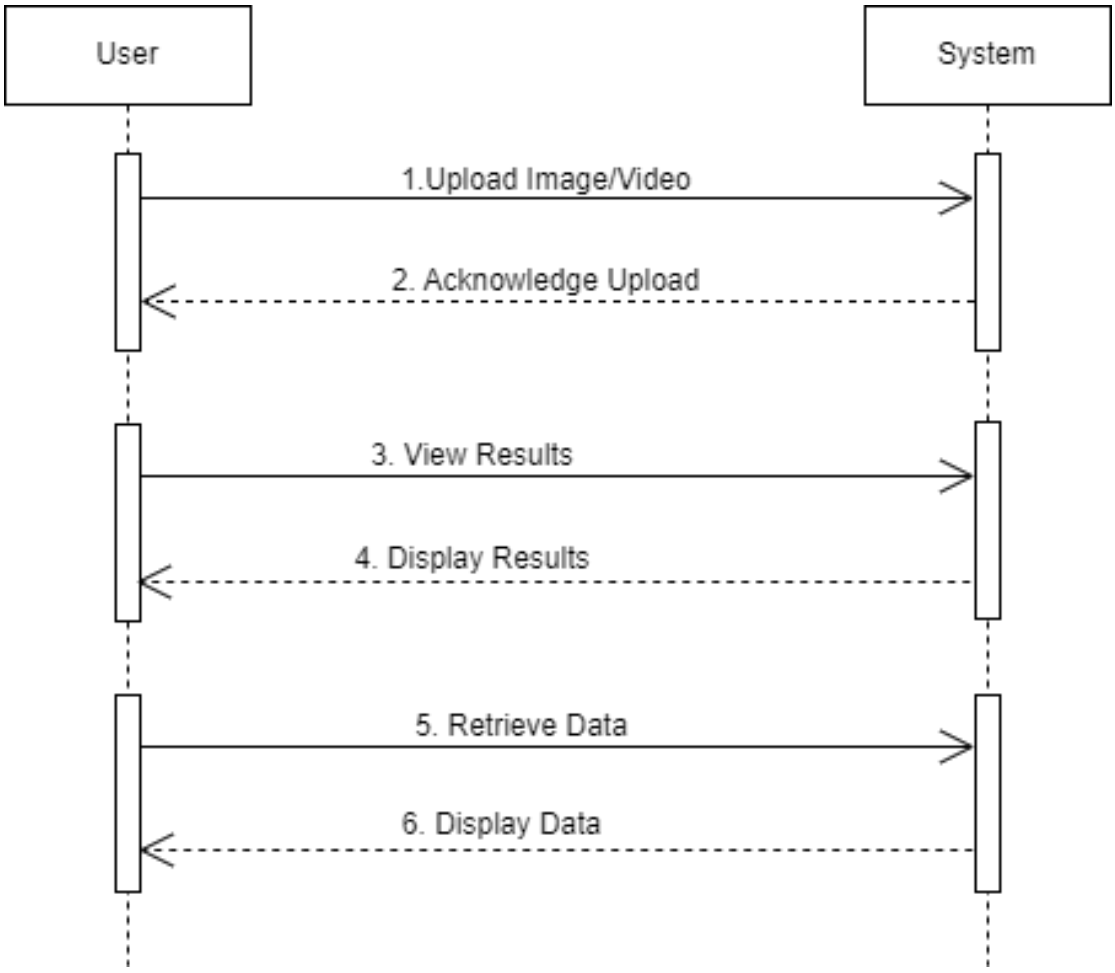

Figure 16: System Sequence Diagram between User and System for Vehicle Tracking System

3.13.2 System Sequence diagram between User and System Admin and System

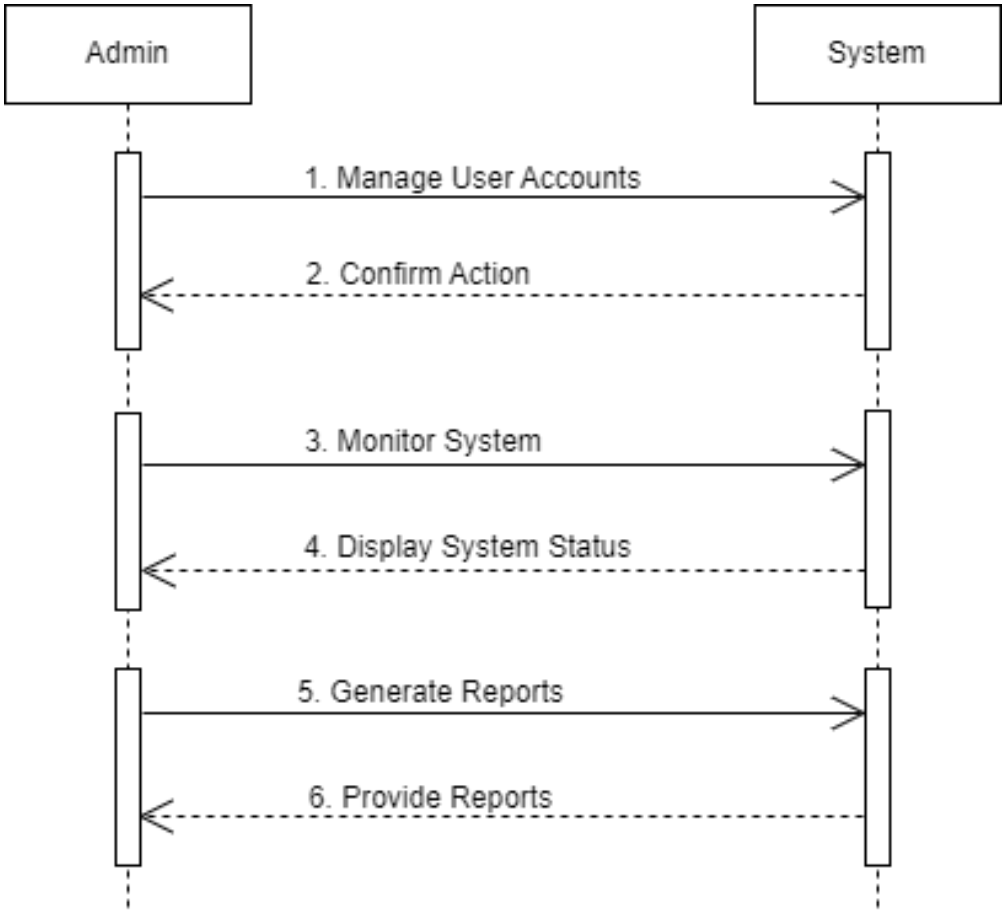

Figure 17: System Sequence Diagram between Admin and System for Vehicle Tracking System

3.13.3 System Sequence diagram between Head Office and System

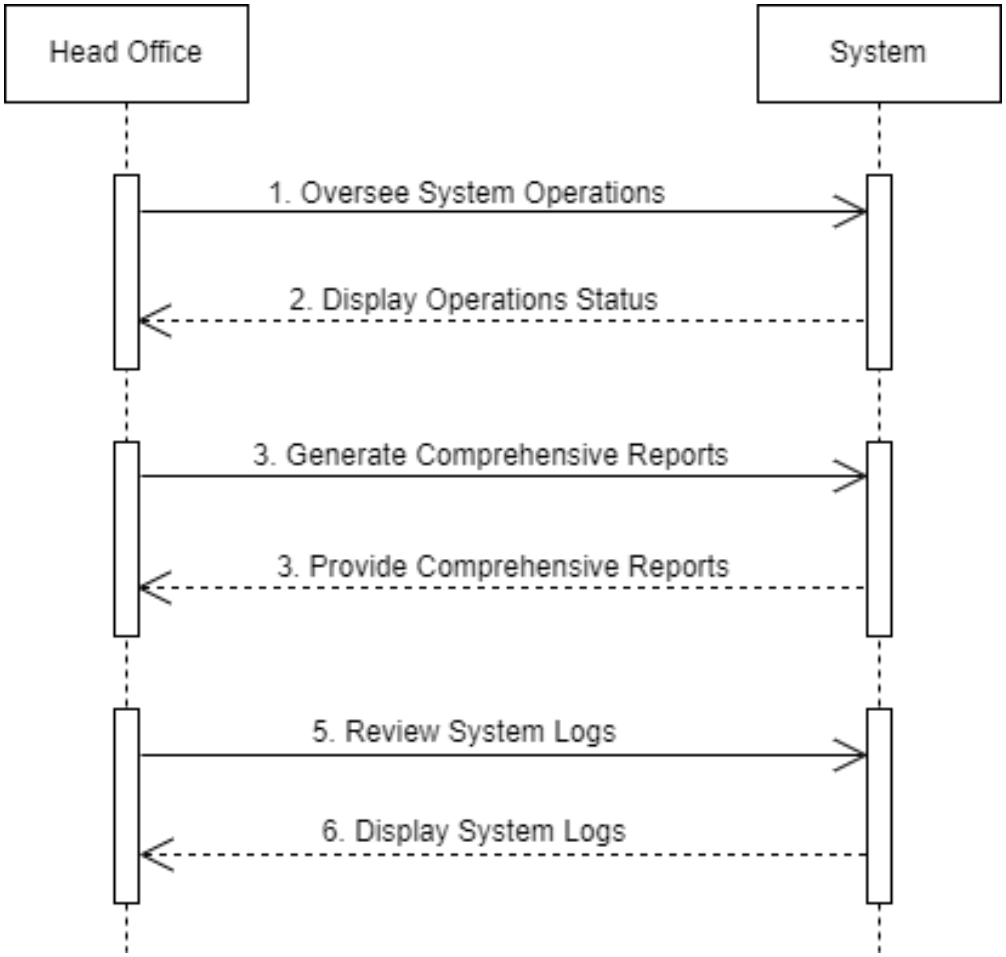

Figure 18: System Sequence Diagram between Head Office and System for Vehicle Tracking System

3.13.4 System Sequence Diagram Between Staff and System

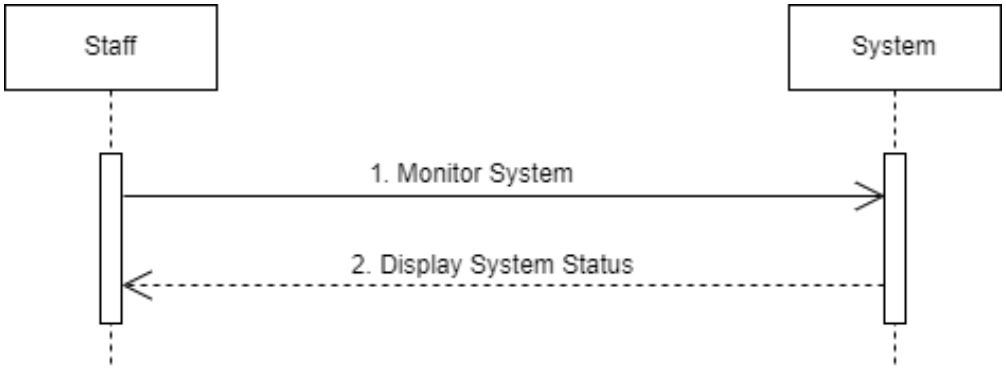

Figure 19: System Sequence Diagram between Staff and System for Vehicle Tracking System

3.14 Service Oriented Architecture

Below is the Service Oriented Architecture illustrating the use of license plate detection for Vehicle Tracking System

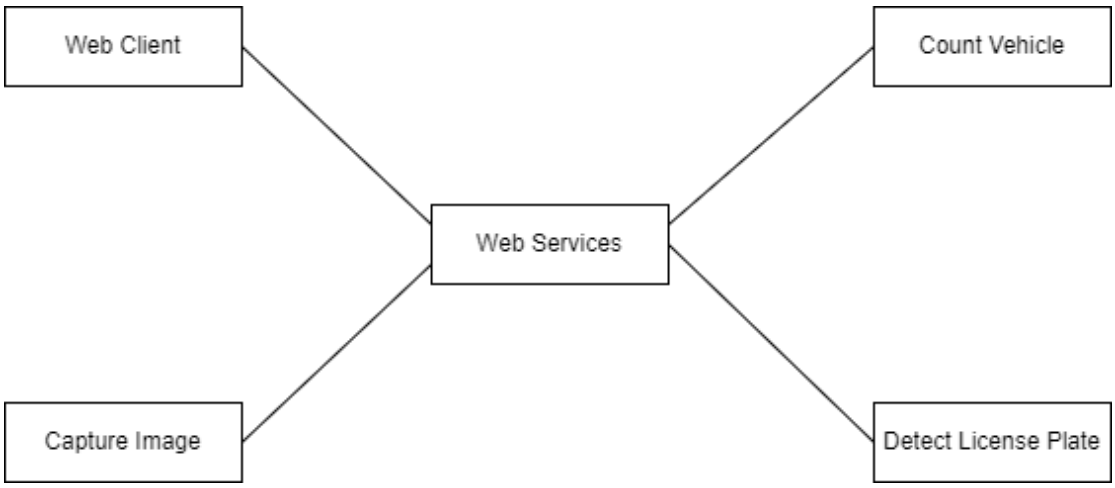

Figure 20: Service Oriented Architecture for Vehicle Tracking System

4 General Overview Of Datasets

1. **Datasets For Object Detection:**

   For the object detection of license plates, two-wheelers, and four-wheelers, we initially captured pictures ourselves in various conditions to ensure a diverse dataset. To further enhance our dataset, we supplemented our images with additional pictures from the [Vehicle Number Plate Dataset(Nepal)— kaggle.com, https://www.kaggle.com/datasets/ishworsubedii/vehicle- number-plate-datasetnepal/data] This combination provided a comprehensive and varied dataset for training our object detection models.

2. **Datasets For Character Segmentation**

   - 22 Diverse Classes: 0, 1, 2, 3, 4,5,6,7,8,9,Ba,Pa,KHA,Pradesh,Bagmati,JA, JHA,CHA,SA,KA
   - Data sources (datasets sourced from Kaggle, collected internally) [Vehicle Number Plate Dataset(Nepal)— kaggle.com,https://www.kaggle.com/datasets/ishworsubedii/vehicle-number-plate-datasetnepal/data]
   - Precise Annotation: Bounding boxes, class labels, and key features capturing the essence of each image.

4.1 Sample Dataset

Here are the sample datasets of 12 diverese classess

Figure 21: Sample Datasets[8]

4.1.1 Infograph of Dataset Classes

Below, the infographic represents 22 distinct classes of datasets

Classes	Numbers
0	829
1	799
2	760
3	561
4	535
5	421
6	561
7	421
8	477
9	497
10	678
Bagmati	126
Pradesh	216
P	405
Cha	200
KA	40
KHA	33
SA	20
JA	28
JHA	22

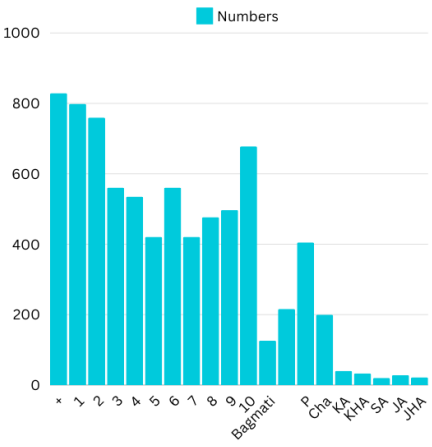

Figure 22: Infograph of Dataset Classes for Character Segmentation

Below, the infographic represents 3 distinct classes of datasets for object detection for licenseplate, twowheelers, fourwheelers

Classes	Number
Two_Wheelers	6202
License_Plate	16001
Four_Wheelers	10000

Figure 23: Infograph of Dataset Classes for Object Detection

5 Mathematical Implementation For YOLOV8

YOLOv8 typically uses a Convolutional Neural Network (CNN) backbone to extract features from input images. The network can be divided into several parts:

- **Input Layer**: Takes the input image, which is often resized to a fixed size (e.g., 640x640).
- **Backbone**: A series of convolutional layers that extract hierarchical features from the image.
- **Neck**: Often includes layers like Feature Pyramid Networks (FPN) or Path Aggregation Networks (PAN) that help in combining features from different levels of the backbone.
- **Head**: Outputs the final predictions, including bounding boxes, objectness scores, and class probabilities.

5.1 Mathematic Representation of CNN Architecture

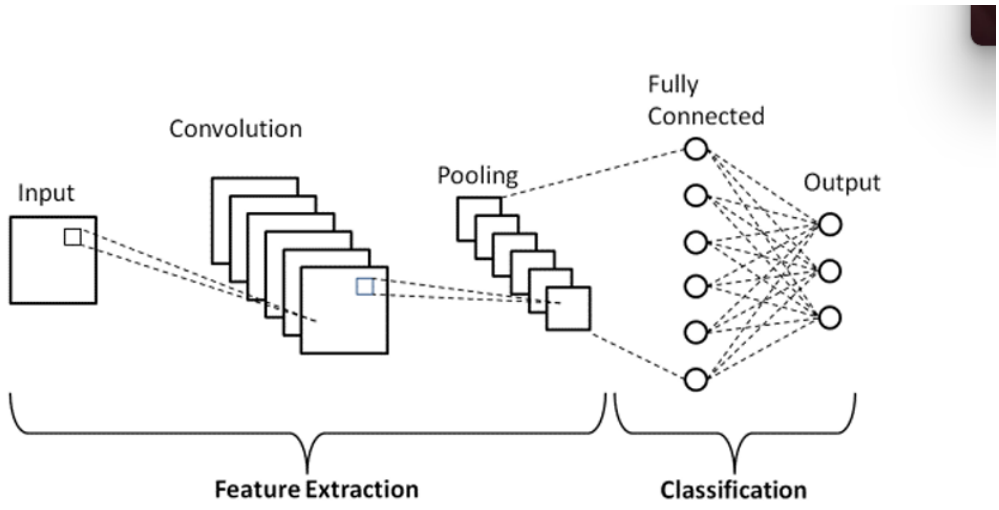

Figure 24: CNN Model Interpretation

The backbone of YOLOv8 is a convolutional neural network (CNN) which extracts features from the input image. The backbone typically involves several convolutional layers, batch normalization layers, and activation functions (like ReLU or Leaky ReLU). Mathematically, a convolution operation for a given layer can be expressed as:

$$h_{i,j,k} = \sigma\left(\sum_{m,n,c} x_{i+m,j+n,c} \cdot w_{m,n,c,k} + b_k\right) \tag{1}$$

where:

- $h_{i,j,k}$ is the output feature map.
- $x_{i+m,j+n,c}$ is the input feature map.
- $w_{m,n,c,k}$ is the weight of the convolution kernel.
- b_k is the bias term.
- σ is the activation function.

5.2 Mathematic Representation Of Bounding Box Prediction

For each bounding box, YOLOv8 predicts several key parameters:

- **Center Coordinates (x, y):** These are the relative coordinates of the center of the bounding box within the grid cell. They are expressed relative to the cell's position.
- **Width (w) and Height (h):** These are the dimensions of the bounding box. They describe how large the bounding box is relative to the grid cell.
- **Objectness Score:** This is the probability that the bounding box contains an object. It helps in determining whether the box is likely to be a true positive detection.
- **Class Probabilities:** These are the probabilities that the object belongs to each possible class. YOLOv8 outputs these probabilities for all the classes the model is trained on.

The predictions for the bounding box parameters are computed as follows:

Figure 25: Bounding Box

Center Coordinates (x, y):

$$\hat{x} = \sigma(t_x) + i \tag{2}$$

$$\hat{y} = \sigma(t_y) + j \tag{3}$$

Where:

- σ is the sigmoid function, which maps the predicted offsets (t_x and t_y) to a value between 0 and 1.
- i and j are the indices of the grid cell within the overall grid.

Width (w) and Height (h):

$$\hat{w} = p_w \cdot e^{t_w} \tag{4}$$

$$\hat{h} = p_h \cdot e^{t_h} \tag{5}$$

Where:

- p_w and p_h are the dimensions of the anchor box associated with the grid cell.
- t_w and t_h are the predicted offsets for the width and height.
- e is the base of the natural logarithm, which allows the model to predict sizes in exponential space for better scaling.

In YOLOv8, bounding box prediction for license plate detection involves dividing the input image into a grid of cells and predicting several parameters for each bounding box within the cells. For this example, consider a 640x640 pixel image divided into a 32x32 grid, resulting in 1024 grid cells.

Parameters and Predictions

For a specific cell at position (15, 10) in the grid, YOLOv8 predicts the following for one bounding box:

- **tx** = 0.4
- **ty** = -0.3
- **tw** = 0.2
- **th** = -0.15
- **Anchor Box Dimensions: pw** = 0.3, **ph** = 0.2

Center Coordinates

Calculate the center coordinates $(\hat{x}, \hat{y})$:

$$\hat{x} = \sigma(tx) + i = \sigma(0.4) + 15$$

$$\hat{y} = \sigma(ty) + j = \sigma(-0.3) + 10$$

Where:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Using:

$$\sigma(0.4) \approx 0.599$$

$$\sigma(-0.3) \approx 0.425$$

We get:

$$\hat{x} = 0.599 + 15 \approx 15.60$$

$$\hat{y} = 0.425 + 10 \approx 10.43$$

The predicted center of the bounding box is approximately (15.60, 10.43) in grid coordinates.

Width and Height

Calculate the width $(\hat{w})$ and height $(\hat{h})$:

$$\hat{w} = pw \cdot e^{tw} = 0.3 \cdot e^{0.2} \approx 0.3 \cdot 1.221 = 0.366$$

$$\hat{h} = ph \cdot e^{th} = 0.2 \cdot e^{-0.15} \approx 0.2 \cdot 0.861 = 0.172$$

The predicted width and height of the bounding box are approximately 0.366 and 0.172 relative to the anchor box dimensions.

Objectness Score and Class Probabilities

- **Objectness Score:** 0.85
- **Class Probabilities:** [0.95, 0.05] where the first value represents the probability of the bounding box containing a license plate.

The high objectness score of 0.85 indicates a strong confidence that the bounding box contains a license plate, and the class probabilities confirm that the object is a license plate.

This bounding box is used to crop and process the text within the detected license plate region for further recognition and analysis.

5.3 Loss Function

YOLOv8's loss function is designed to optimize three primary components: bounding box coordinates, objectness score, and class probabilities. The overall loss L can be expressed as a combination of these components:

$$L = L_{\text{box}} + L_{\text{obj}} + L_{\text{cls}}$$

Let's break down each component in detail and provide an example related to license plate detection.

5.3.1 Bounding Box Loss (L_{box})

The bounding box loss L_{box} measures the error in the predicted bounding box coordinates. This loss typically combines Mean Squared Error (MSE) and Intersection over Union (IoU) loss. The goal is to make the predicted bounding box as close as possible to the ground truth bounding box.

$$L_{\text{box}} = \sum_{i=1}^{S^2} \sum_{j=1}^{B} 1_{ij}^{\text{obj}} \left[(\hat{x}_{ij} - x_{ij})^2 + (\hat{y}_{ij} - y_{ij})^2 + (\hat{w}_{ij} - w_{ij})^2 + (\hat{h}_{ij} - h_{ij})^2 \right]$$

Where:

- S^2 is the number of grid cells.
- B is the number of bounding boxes per grid cell.
- 1_{ij}^{obj} is an indicator function that equals 1 if object j is in cell i, otherwise 0.
- $\hat{x}_{ij}, \hat{y}_{ij}, \hat{w}_{ij}, \hat{h}_{ij}$ are the predicted bounding box coordinates (center coordinates x, y, width w, and height h).

- $x_{ij}, y_{ij}, w_{ij}, h_{ij}$ are the ground truth bounding box coordinates.

Example: If the ground truth bounding box for a license plate is located at (0.5, 0.5) in normalized coordinates, with a width and height of 0.2, and the predicted bounding box is at (0.45, 0.48) with a width of 0.18 and height of 0.22, the bounding box loss would penalize the differences between these values to bring the predicted box closer to the ground truth.

5.3.2 Objectness Loss (L_{obj})

The objectness loss L_{obj} measures the error in the predicted objectness score, which indicates the confidence that a bounding box contains an object. This loss typically uses Binary Cross-Entropy (BCE) loss.

$$L_{\text{obj}} = \sum_{i=1}^{S^2} \sum_{j=1}^{B} \left[1_{ij}^{\text{obj}} (\hat{o}_{ij} - 1)^2 + (1 - 1_{ij}^{\text{obj}})(\hat{o}_{ij})^2 \right]$$

Where:

- $\hat{o}_{ij}$ is the predicted objectness score for bounding box j in cell i.

Example: If a grid cell is supposed to detect a license plate but the predicted objectness score is low (e.g., 0.3), the objectness loss will be high, encouraging the model to increase the confidence score in future predictions.

5.3.3 Class Loss (L_{cls})

The class loss L_{cls} measures the error in the predicted class probabilities. This loss typically uses Binary Cross-Entropy (BCE) loss or categorical cross-entropy.

$$L_{\text{cls}} = \sum_{i=1}^{S^2} 1_i^{\text{obj}} \sum_{c=1}^{C} (\hat{p}_i(c) - p_i(c))^2$$

Where:

- 1_i^{obj} is an indicator function that equals 1 if there is an object in cell i, otherwise 0.

- $\hat{p}_i(c)$ is the predicted probability for class c in cell i.

- $p_i(c)$ is the ground truth probability for class c in cell i.

Example: If the model predicts that a detected object in a grid cell is a license plate with a probability of 0.7 but the actual class probability is 1 (indicating it is definitely a license plate), the class loss will penalize this difference to improve future predictions.

Summary

The loss function in YOLOv8 ensures that the model optimizes for accurately predicting the bounding box coordinates, the presence of objects, and the correct class of objects. Each component of the loss function plays a crucial role in training the model to detect and classify objects effectively.

Example in Context: When applied to license plate detection, the loss function will ensure that the model accurately predicts the location of the license plate (bounding box loss), confidently identifies that the detected object is indeed a license plate (objectness loss), and correctly classifies it among other potential classes (class loss). Through training, the model minimizes these losses, leading to improved detection and classification performance.

5.4 Inference

During inference, the model outputs bounding boxes, objectness scores, and class probabilities for each grid cell. Non-Maximum Suppression (NMS) is applied to remove duplicate detections and retain the most confident predictions. The steps for NMS are as follows:

1. Sort all bounding boxes by their objectness score L_{object}.
2. Select the bounding box with the highest score and remove it from the list.
3. Calculate the Intersection over Union (IoU) between the selected bounding box and all remaining boxes:

$$\text{IoU}(A, B) = \frac{\text{area}(A \cap B)}{\text{area}(A \cup B)}$$

4. Remove all bounding boxes with IoU greater than a predefined threshold $\text{IoU}_{\text{threshold}}$.
5. Repeat steps 2-4 for the remaining bounding boxes.

5.5 Post-Processing

- **Non-Maximum Suppression (NMS)**: Eliminates redundant bounding boxes by keeping the one with the highest confidence score and discarding others with high IoU overlap.
- **Thresholding**: Applies confidence and class probability thresholds to filter out low-confidence detections.

5.6 Mathematical Summary

- **Backbone**: Feature extraction using convolutional layers.
- **Grid Cell Predictions**: Each grid cell predicts multiple bounding boxes and their associated parameters.
- **Bounding Box Parameters**: $\hat{x}$, $\hat{y}$, $\hat{w}$, $\hat{h}$.

- **Loss Function**: Combination of box loss, objectness loss, and class loss.
- **Inference and Post-Processing**: NMS and thresholding to finalize detections.

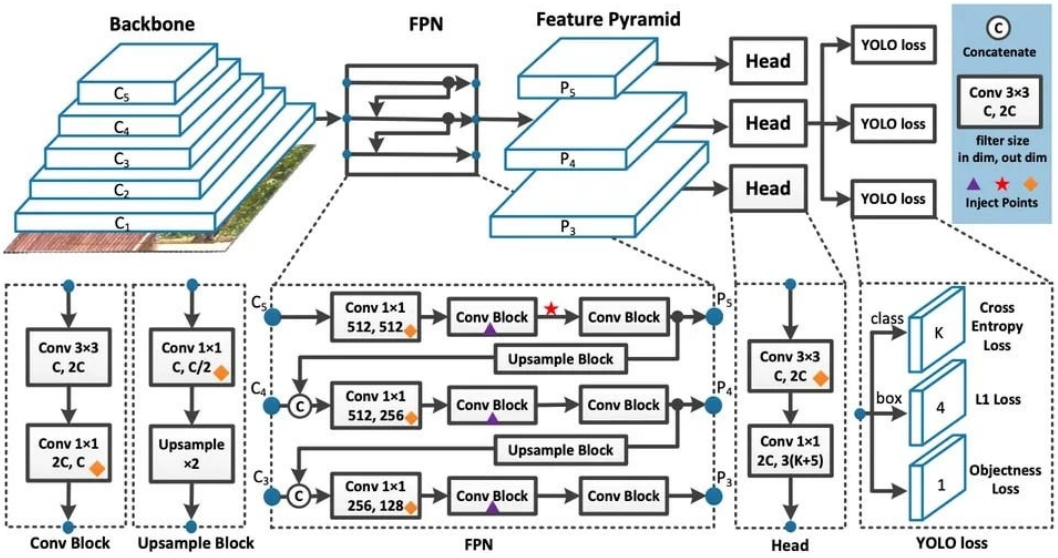

Figure 26: YOLOV8 Architecture [9]

5.7 Evaluation Metrics

To assess the effectiveness of object detection models, various metrics are commonly employed, including precision, recall, average precision, and mean average precision, which rely on the concept of intersection over union (IoU). IoU measures the overlap between two bounding boxes, determining the correctness of predictions. By setting an IoU threshold, predictions are categorized as true positives (TP) if their IoU with the ground truth exceeds the threshold, and false positives (FP) otherwise. The IoU threshold dictates the stringency of model evaluation; a higher threshold demands precise localization for a TP designation, while a lower threshold permits more leniency in localization.

Precision (P) is the ratio of TP to all predicted positives (TP+FP), and it reflects how accurate the model is in detecting objects. Recall (R) is the ratio of TP to all ground truth positives (TP+FN), where FN is false negatives, and it reflects how complete the model is in detecting objects. Precision and recall can be computed using the following equations, respectively:

$$P = \frac{TP}{TP + FP}, \tag{6}$$

$$R = \frac{TP}{TP + FN} \tag{7}$$

AP is the average of precision values at different recall levels for each class, and it reflects how well the model can detect objects across various confidence thresholds. It can be computed using the following equation:

$$AP = \int_0^1 p(R)\, dR \tag{8}$$

where P(R) is the precision at a given recall level R.
Mean average precision (mAP) is the mean of AP values for all classes, and it summarizes the overall performance of the model for multiple classes. It can be computed

using the following equation:

$$mAP = \frac{1}{N} \sum_{i=1}^{n} AP_i \tag{9}$$

where N represents the number of classes, and APi is the average precision for class i.

6 Model Selection And Training

YOLOv8 (You Only Look Once version 8) was chosen for its real-time detection capabilities and high accuracy in object detection tasks.

7 Data Pre-processing for License Plate Detection

7.1 Data Collection

A dataset consisting of approximately 16,000 images was collected. The images varied in terms of lighting conditions, angles, and backgrounds to ensure a robust training set.The dataset included:

- Images containing clear and obscured license plates.
- Different types of vehicles, including twowheelers and fourwheelers.
- Source:https://www.kaggle.com/datasets/ishworsubedii/vehicle-number-plate-datasetnepal/data

7.2 Data Annotation

Annotation of the dataset was carried out meticulously. Each image was reviewed and annotated to label:

- **License Plates**: Regions containing license plates.
- **Two Wheelers**: Entire two-wheeler vehicles.
- **Four Wheelers**: Entire four-wheeler vehicles.

Annotations were done using tools such as LabelImg. The output was saved in a format compatible with YOLO (e.g., YOLO format text files).

Figure 27: Annotation of LicensePlate, TwoWheeler and FourWheeler

7.3 Data Augmentation

To improve the diversity and robustness of the yolov8 model, data augmentation techniques were applied as per needed. This included:

- **Horizontal Flip:** Flip the image horizontally with a 50% probability.

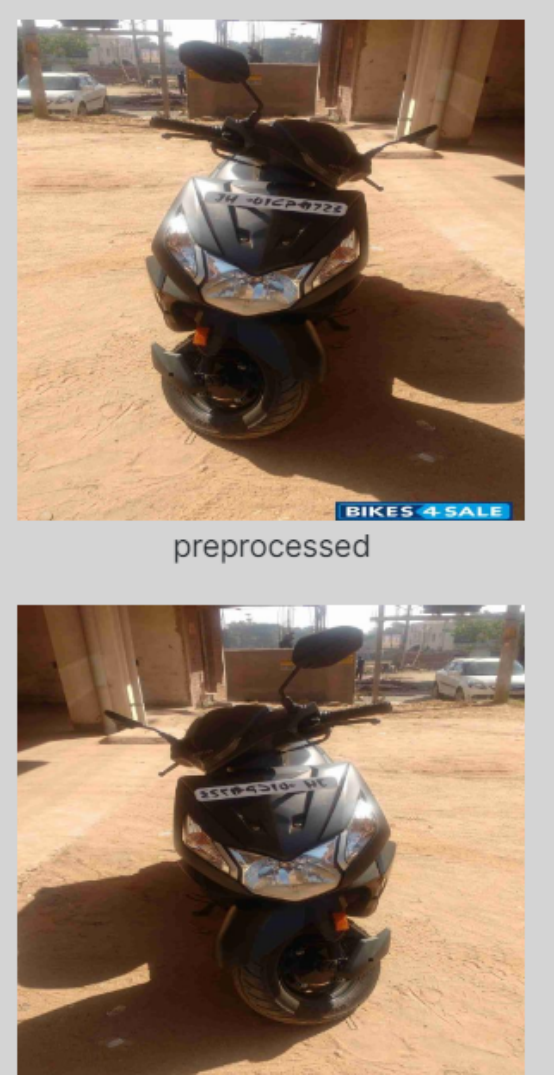

Figure 28: Flip horizontal

- **Crop:** Randomly crop the image to a specific size.

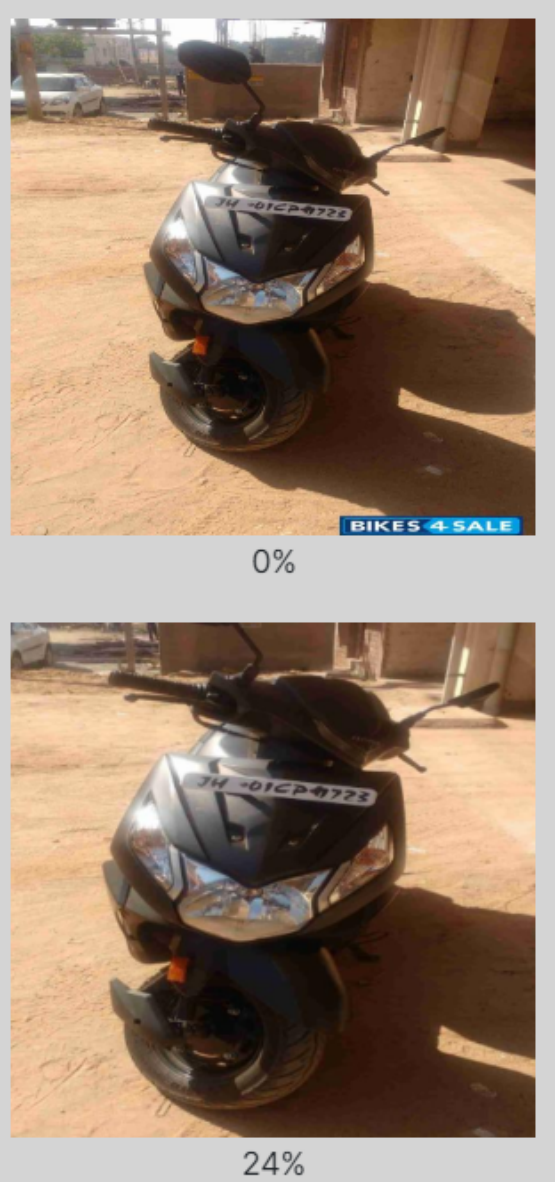

Figure 29: Crop

- **Rotation:** Randomly rotate the image within a specified range.

Figure 30: Rotation

- **Shear:** Apply horizontal and vertical shear transformations.

Figure 31: Shear

- **Grayscale:** Convert some images to grayscale.

Figure 32: Gray scale

- **Hue, Saturation, Brightness:** Randomly adjust the hue, saturation, and brightness.

Figure 33: Brightness

Figure 34: saturation

Figure 35: Hue

- **Exposure:** Randomly adjust the exposure.

Figure 36: Exposure

- **Blur:** Apply motion blur to some images.
- **Noise:** Add Gaussian noise to some images.

Augmentation was performed using libraries such as OpenCV, Albumentations, or TensorFlow and was ready for yolov8 training

Figure 37: Blur

Figure 38: Noise

8 Training Process In YoloV8 for License-Plate, Four Wheeler and Two Wheelers

8.1 Data Preparation

: The dataset was split into training, validation, and test sets (e.g., 70 training, 20 validation, 10 testing).

8.2 Configuration

: YOLOv8 was configured for training with three classes:

- Class 0: License Plate
- Class 1: Two-Wheeler
- Class 2: Four-Wheeler

8.3 Training

: The model was trained using the prepared dataset. Hyperparameters such as learning rate, batch size, and number of epochs were tuned to achieve optimal performance.

8.4 Evaluation

: After training, the model was evaluated on the test set to measure metrics such as precision, recall, and mAP (mean Average Precision).

8.4.1 Precision-Confidence Curve Analysis

The Precision-Confidence curve provides insights into the model's precision at different confidence thresholds. Here's a breakdown of the performance for each class:

- **Four_Wheeler (Light Blue Line)**:

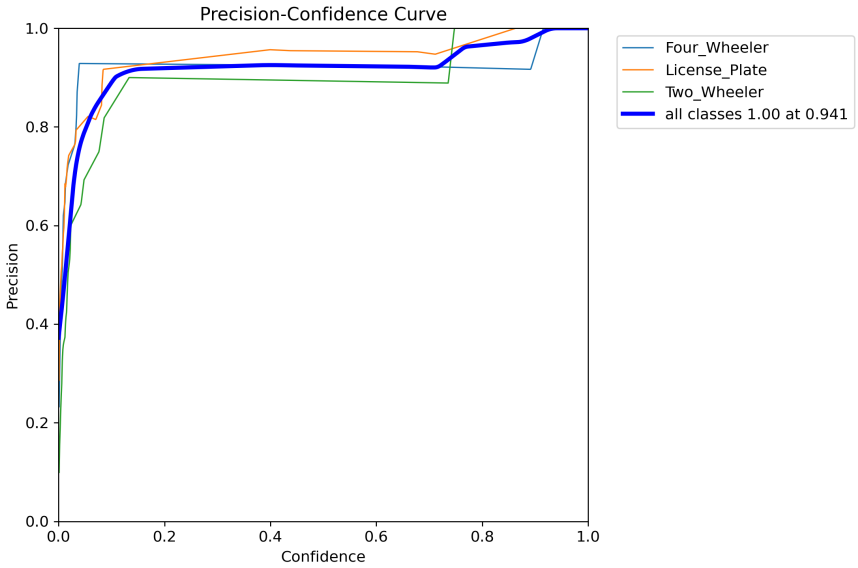

Figure 39: Precision-Confidence Curve

- The precision starts at a lower value and rapidly increases as confidence increases, stabilizing around 0.9 before reaching near-perfect precision (1.0) at high confidence levels.
- The performance is consistent with minor fluctuations, indicating that the model performs well in detecting FourWheelers across various confidence levels.

- **License_Plate (Orange Line)**:
  - The precision for this class remains relatively high across the confidence spectrum, starting strong and staying close to 1.0.
  - This suggests that the model is very accurate in detecting LicensePlates, even at lower confidence levels.

- **Two_Wheeler (Green Line)**:
  - The precision starts lower compared to the other classes and gradually improves with increasing confidence.
  - There are more fluctuations in this curve, indicating that the model's performance is less stable for detecting TwoWheelers, particularly at lower confidence levels.

- **All Classes (Thick Blue Line)**:
  - When considering all classes, the precision increases steadily and achieves perfect precision (1.0) at a confidence level of around 0.941.
  - This aggregate curve demonstrates that the model's overall performance is robust, with high precision at higher confidence thresholds.

8.4.2 Recall-Confidence Curve Analysis

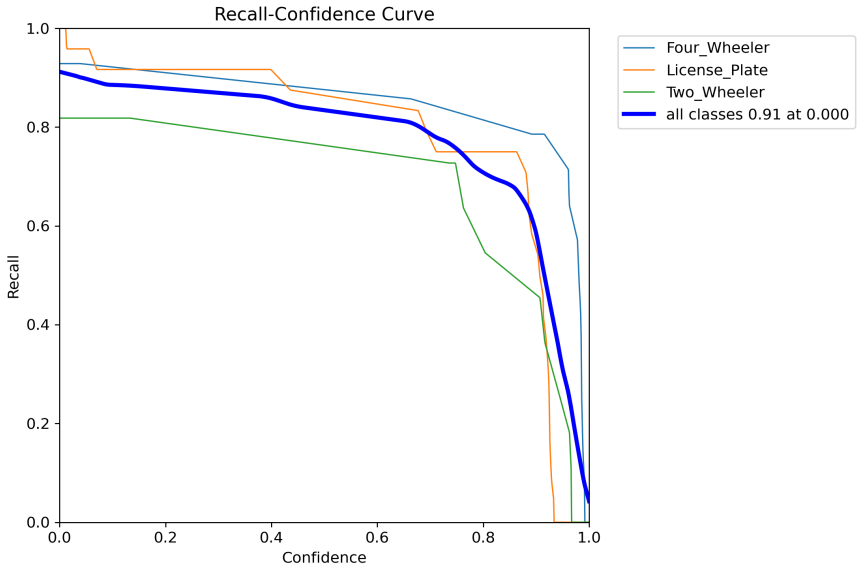

Figure 40: RecallConfidence Cure

The Recall-Confidence curve provides insights into the model's recall at different confidence levels for different classes:

- **Four_Wheeler**:
  - Maintains a high recall across most confidence levels but shows a sharp decline at higher confidence thresholds.
- **License_Plate**:
  - Initially, shows a very high recall close to 1.0 but drops significantly as the confidence threshold increases, indicating some sensitivity to confidence levels.
- **Two_Wheeler**:

- Has a lower recall compared to Four_Wheeler and License_Plate, and its recall drops more gradually with increasing confidence.

8.4.3 Precision-Recall Curve Analysis

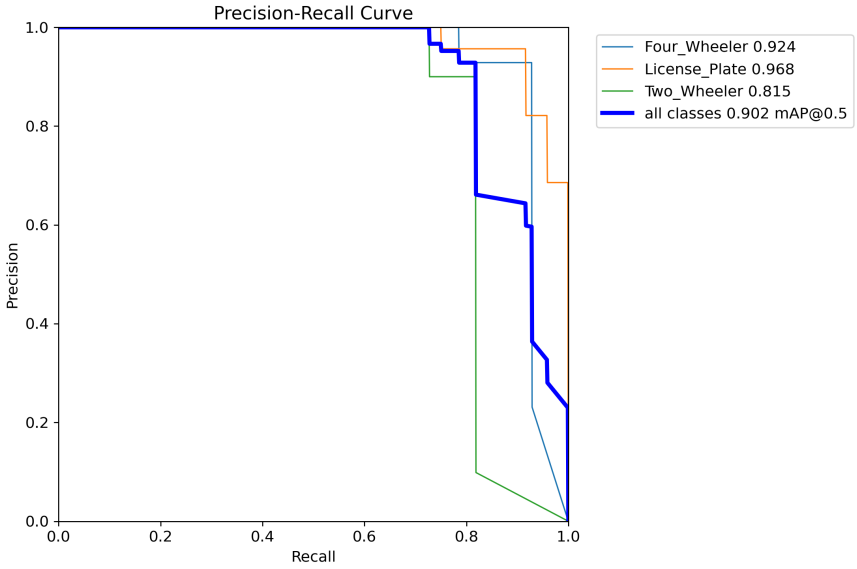

Figure 41: Precision-Recall Curve Analysis

The Precision-Recall curve is crucial for understanding the trade-off between precision and recall for each class. Here's a detailed interpretation:

- **Four_Wheeler (Light Blue Line)**:
  - The model achieves a high precision of 0.924, maintaining good precision until recall starts to approach 1.0, where precision slightly drops.
  - The recall is strong, indicating the model's effectiveness in detecting Four_Wheelers.
- **License_Plate (Orange Line)**:
  - The precision for License_Plate detection is very high at 0.968, with the curve showing minimal drop in precision even as recall increases.
  - This suggests that the model is highly reliable for License_Plate detection, maintaining both high precision and high recall.
- **Two_Wheeler (Green Line)**:

- The precision for Two_Wheeler detection is 0.815, which is lower compared to the other classes.
- The curve shows a more significant drop in precision as recall increases, indicating that the model has more false positives in detecting Two_Wheelers.

- **All Classes (Thick Blue Line)**:
  - The mAP@0.5 (mean Average Precision at IoU threshold of 0.5) is 0.902, reflecting the overall high performance of the model across all classes.
  - This high mAP score demonstrates that the model maintains a good balance between precision and recall, performing well in detecting and accurately identifying objects across different classes.

8.4.4 Confusion Matrix Analysis

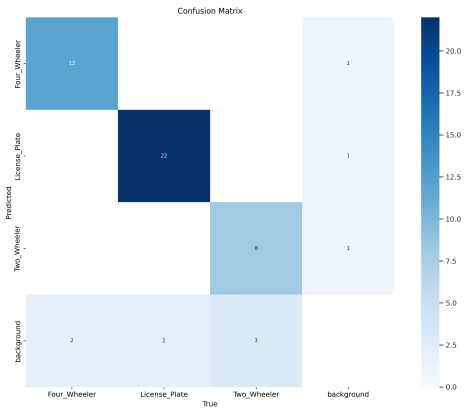

Figure 42: Confusion Matrix

The confusion matrix (Figure 42) provides a breakdown of the true positive, false positive, false negative, and true negative counts for each class:

- **Four_Wheeler**:
  - True Positives: 12
  - False Positives: 1 (misclassified as License_Plate)
  - False Negatives: 2 (misclassified as background)

- **License_Plate**:

  - True Positives: 22
  - False Positives: 2 (misclassified as background)
  - False Negatives: 1 (misclassified as Two_Wheeler)

- **Two_Wheeler**:
  - True Positives: 8
  - False Positives: 1 (misclassified as background)
  - False Negatives: 1 (misclassified as License_Plate)

- **Background**:
  - Higher number of misclassifications, indicating that distinguishing between actual classes and background can be challenging.

8.5 Performance Analysis and Validation

8.5.1 Performance Metrics

Several key metrics were analyzed to assess the performance of the YOLOv8 model, including precision, recall, and the mean Average Precision (mAP) at an Intersection over Union (IoU) threshold of 0.5. Here is a detailed performance analysis based on these metrics:

- **Precision**:
  - Measures the accuracy of the positive predictions made by the model. High precision indicates that the model is effective in minimizing false positives.
  - Four_Wheeler: Achieves a precision of 0.924.
  - License_Plate: Achieves the highest precision of 0.968.
  - Two_Wheeler: Shows a lower precision of 0.815.
  - All Classes: Achieves perfect precision of 1.0 at a confidence threshold of 0.941.
- **Recall**:
  - Measures the model's ability to identify all relevant instances in the dataset. High recall indicates that the model is effective in minimizing false negatives.
  - The recall performance is inferred from the Precision-Recall curves, where each class shows high recall until precision drops significantly as recall approaches 1.0.
- **Mean Average Precision (mAP)**:
  - The mAP@0.5 is a comprehensive metric that combines both precision and recall, providing an overall performance measure.
  - All Classes: The model achieves a high mAP@0.5 of 0.902, indicating strong overall performance across all classes.

- **Overall Accuracy**:

  Sum of correct predictions:

$$12 + 22 + 8 + 3 = 45$$

  Total number of predictions:

$$12 + 1 + 1 + 22 + 1 + 8 + 1 + 2 + = 48$$

  Accuracy:

$$\text{Accuracy} = \frac{45}{53} \approx 0.937$$

8.5.2 Validation Analysis

The model's validation process involved evaluating its performance on a separate validation dataset. The following points highlight key aspects of the validation analysis:

- **Validation Dataset**:
  - The validation dataset was curated to include a diverse set of images representing various scenarios and conditions for FourWheelers, LicensePlates, and TwoWheelers. This ensures a comprehensive assessment of the model's generalization capabilities.
- **Class-wise Performance**:
  - **License_Plate**: The model performed exceptionally well in detecting License_Plates, maintaining high precision and recall across different images. This class's high performance can be attributed to distinct features and clear boundaries that the model can easily learn.
  - **Four_Wheeler**: Similarly, the model demonstrated strong performance in detecting Four_Wheelers, with minor precision fluctuations at different confidence levels.
  - **Two_Wheeler**: While the model performed reasonably well, it showed more variability in precision and recall. This could be due to the diverse

appearances and varying sizes of Two_Wheelers in the images, making detection more challenging.

- **Confidence Thresholds**:
  - The precision-confidence curve indicates that as the confidence threshold increases, precision improves for all classes. This suggests that setting an appropriate confidence threshold can help balance precision and recall, optimizing model performance for specific applications.
- **Error Analysis**:
  - To further validate the model, an error analysis was conducted to identify common failure cases. For Two_Wheelers, the errors were mainly due to partial occlusions, variations in appearance, and smaller object sizes. Addressing these issues through additional training data, augmentation techniques, or model architecture improvements could enhance detection accuracy.

9 Data Preprocessing For OCR

We collected diverse datasets for character detection of license plates from Kaggle and also cropped out the characters on our own.

9.1 Data Collection

: The following table shows the total numbers of characters collected for data-preprocessing and training.

Classes	Numbers
0	829
1	799
2	760
3	561
4	535
5	421
6	561
7	421
8	477
9	497
10	678
Bagmati	126
Pradesh	216
P	405
Cha	200
KA	40
KHA	33
SA	20
JA	28
JHA	22

Table 1: Character Datasets

9.2 Data Annotation:

Data are annotated for each characters using jetbeans, created grouthtruthvalue and TTF files.

Figure 43: Character Annotation

9.3 Data Augmentation

Increase the diversity of training data by adding variations in font, size, color, and background noise.

- **Horizontal Flip:** Flip the image horizontally with a 50% probability.
- **Crop:** Randomly crop the image to a specific size.
- **Rotation:** Randomly rotate the image within a specified range.
- **Shear:** Apply horizontal and vertical shear transformations.
- **Grayscale:** Convert some images to grayscale.

- **Hue, Saturation, Brightness:** Randomly adjust the hue, saturation, and brightness.
- **Exposure:** Randomly adjust the exposure.
- **Blur:** Apply motion blur to some images.
- **Noise:** Add Gaussian noise to some images.

Enhanced the input images using techniques such as denoising, binarization, and contrast adjustment to make text clearer for the OCR engine.

10 Training and Model Selection For OCR

During the training phase, we trained our dataset using three models: Tesseract, YOLOv8, and CNN. However, the accuracy obtained from Tesseract and CNN was lower compared to YOLOv8.

Before training, Dataset is divided into 70 per training, 20 per validation, and 10 per test sets. The model is trained to detect and recognize 22 different classes, ranging from digits to specific labels.Training is done with 100 epochs and a batch size of 16 on different models

DatasetSource:https://www.kaggle.com/datasets/ishworsubedii/vehicle-number-plate-datasetnepal/data

Class Notation	Label Name
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	BA
11	Bagmati
12	CHA
13	JA
14	KA
15	KHA
16	Pradesh
17	jha
18	p
19	pra
20	sa
21	ya

Figure 44: Classes for OCR

10.1 Model Comparision

Here's the comparison table with data for Tesseract and CNN models, along with the data for the YOLOv8 model

Yolov8 Model

Following is the Table of Yolov8 with each characters and their Score in different curves

Model evaluation:

- **Precision:** 1.00 (at 0.995 confidence)
- **F1 Score:** 0.89 (at 0.754 confidence)

A	B	C	D	E	F	G
Class	**YOLOv8_True Instances**	**YOLOv8_Correct Predictions**	**YOLOv8_Recall**	**YOLOv8_Precision**	**YOLOv8_F1 Score**	**YOLOv8_AP**
0	50	39	0.78	0.93	0.838	0.767
1	52	47	0.9	0.96	0.929	0.939
2	46	35	0.76	0.87	0.806	0.738
3	32	28	0.87	0.97	0.916	0.914
4	38	37	0.97	0.97	0.973	0.97
5	32	26	0.81	0.84	0.825	0.876
6	31	27	0.84	0.92	0.881	0.879
7	47	42	0.89	0.91	0.9	0.888
8	47	45	0.96	0.98	0.97	0.952
9	37	36	0.97	0.97	0.973	0.983
BA	48	48	1	1	1	0.978
Bagmati	9	9	1	1	1	0.995
CHA	22	22	1	1	1	0.907
JA	4	3	0.75	0.85	0.796	0.763
KHA	1	1	1	1	1	0.995
Pradesh	17	15	0.88	0.91	0.895	0.905
p	0	0	-	-	-	0.961
All Classes	-	-	-	1.00 (at 0.995 confidence)	0.89 (at 0.754 confidence)	0.906 (mAP@0.5)

Table 2: Yolov8 Model Ananysis

- **AP:** 0.906 (mAP@0.5)

$$\textbf{Accuracy} = \frac{499}{564} \approx 0.9186$$

Tesseract Model: Following is the Table of Tesseract Model Metrics with each characters and their Score in different curves

Class	Tesseract_True Instances	Tesseract_Correct Predictions	Tesseract_Recall	Tesseract_Precision	Tesseract_F1 Score	Tesseract_AP
0	50	25	0.5	0.6	0.545	0.52
1	52	30	0.58	0.65	0.61	0.64
2	46	20	0.43	0.55	0.485	0.47
3	32	18	0.56	0.62	0.585	0.55
4	38	25	0.66	0.72	0.688	0.68
5	32	15	0.47	0.5	0.485	0.5
6	31	16	0.52	0.57	0.545	0.53
7	47	22	0.47	0.55	0.507	0.52
8	47	24	0.51	0.6	0.548	0.56
9	37	20	0.54	0.6	0.57	0.59
BA	48	30	0.62	0.68	0.644	0.64
Bagmati	9	5	0.56	0.62	0.585	0.59
CHA	22	12	0.55	0.61	0.577	0.57
JA	4	2	0.5	0.55	0.524	0.53
KHA	1	1	1	1	1	0.995
Pradesh	17	10	0.59	0.66	0.623	0.63
p	0	0	-	-	-	0.5
All Classes	-	-	-	0.63 (at 0.90 confidence)	0.56 (at 0.80 confidence)	0.610 (mAP@0.5)

Table 3: Tesseract Model Anaylsis

Model Evaluation

- **Precision:** 0.63 (at 0.90 confidence)
- **F1 Score:** 0.56 (at 0.80 confidence)
- **AP:** 0.610 (mAP@0.5)

$$\textbf{Accuracy} = \frac{45}{75} \approx 0.67$$

CNN Model:

Following is the Table of CNN Model Metrics with each characters and their Score in different curves

Class	CNN_True Instances	CNN_Correct Predictions	CNN_Recall	CNN_Precision	CNN_F1 Score	CNN_AP
0	50	35	0.7	0.8	0.746	0.72
1	52	40	0.77	0.82	0.793	0.81
2	46	30	0.65	0.75	0.698	0.68
3	32	26	0.81	0.85	0.828	0.82
4	38	35	0.92	0.92	0.92	0.92
5	32	22	0.69	0.75	0.718	0.72
6	31	25	0.81	0.83	0.819	0.81
7	47	35	0.74	0.8	0.769	0.76
8	47	38	0.81	0.85	0.828	0.83
9	37	33	0.89	0.9	0.895	0.88
BA	48	45	0.94	0.94	0.94	0.93
Bagmati	9	8	0.89	0.91	0.9	0.9
CHA	22	18	0.82	0.85	0.835	0.82
JA	4	3	0.75	0.78	0.764	0.75
KHA	1	1	1	1	1	0.995
Pradesh	17	14	0.82	0.86	0.839	0.84
p	0	0	-	-	-	0.75
All Classes	-	-	-	0.87 (at 0.95 confidence)	0.82 (at 0.85 confidence)	0.860 (mAP@0.5)

Table 4: CNN Model

Model Evaluation

- **Precision:** 0.87 (at 0.95 confidence)
- **F1 Score:** 0.82 (at 0.85 confidence)
- **AP:** 0.860 (mAP@0.5)

$$\textbf{Accuracy} = \frac{45}{52} \approx 0.86$$

10.2 Comparsion test For Tesseract and Yolov8 OCR

Tesseract OCR vs YoloV8 OCR

Figure 45: Tesseract OCR vs YoloV8 OCR

InfoGraphics:

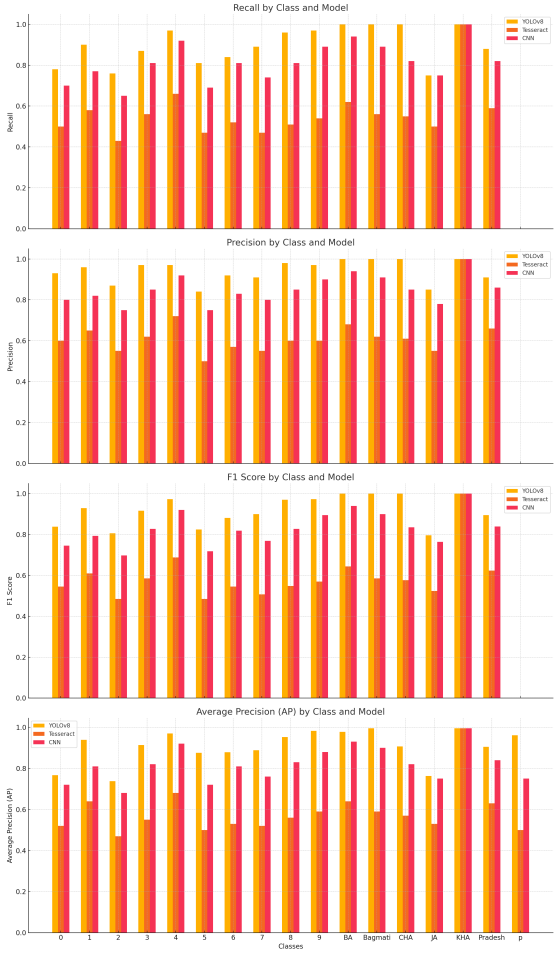

Figure 46: curve comparrsion

1. **Recall:** YOLOv8 shows significantly higher recall values across most classes

compared to Tesseract and CNN. This indicates that YOLOv8 is more effective in identifying the relevant instances correctly.

2. **Precision:** YOLOv8 also maintains higher precision scores across most classes, suggesting it makes fewer false positive errors compared to the other models.

3. **F1 Score:** The F1 score, which balances precision and recall, is higher for YOLOv8 in most classes. This demonstrates that YOLOv8 has a better overall performance in terms of precision and recall.

4. **Average Precision (AP):** YOLOv8 again outperforms Tesseract and CNN in average precision for most classes, indicating superior performance in precision-recall trade-off.

Conclusion In conclusion, the YOLOv8 model demonstrates significantly better performance across recall, precision, F1 score, and average precision metrics with accuracy of 91% compared to both the Tesseract and CNN models. This illustrates the effectiveness of YOLOv8 in handling the task of license plate detection and recognition.

11 Model Evaluation and Validation of YOLOv8 for OCR and recognition

lets discuss the model evaulation and performance with different curves in brief:

11.1 Confusion Matrix

11.1.1 Description

- This confusion matrix represents the raw counts of correct and incorrect predictions made by a classification model.
- The diagonal elements show the number of correct predictions for each class.
- The off-diagonal elements show the misclassifications.

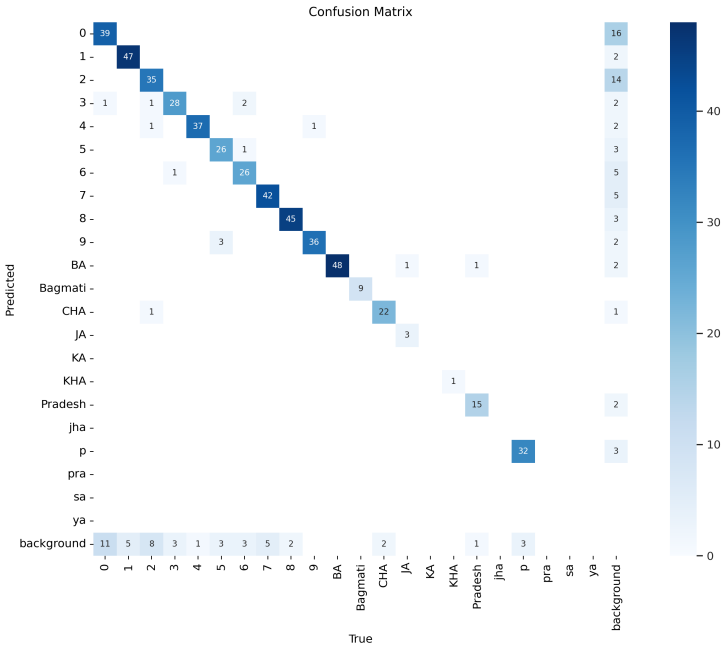

Figure 47: Confusion Matrix with raw counts

Analysis

- The classes range from 0 to 9, and various labels like BA, Bagmati, CHA, etc.
- The model performs well on most classes, with high counts on the diagonal (e.g., 39 for class 0, 47 for class 1).
- Some misclassifications are present, such as classifying class 4 as class 3, which has 1 misclassification and character "p" being predicted as "background" multiple times, suggesting areas for improvement in distinguishing specific characters.

Overall Accuracy

Sum of correct predictions:

$$39+47+35+28+37+26+27+42+45+36+48+22+0+3+1+0+15+0+32+0+0+0+16 = 499$$

Total number of predictions:

$$50+52+43+31+40+30+32+45+48+39+49+25+2+4+2+1+17+1+35+0+0+0+18 = 564$$

$$\text{Accuracy} = \frac{499}{564} \approx 0.9186$$

11.2 Normalized Confusion Matrix

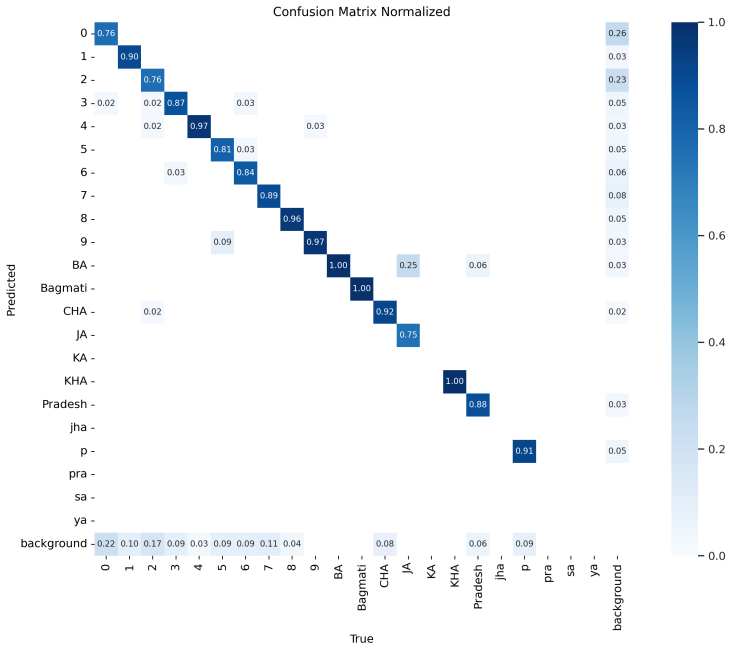

Figure 48: Normalized Confusion Matrix

Description

- This matrix shows the normalized values of the confusion matrix, where each cell represents the proportion of predictions made for that class.
- The values range from 0 to 1, indicating the fraction of true instances of a class that are correctly or incorrectly predicted.

Analysis

- The normalization highlights the performance more clearly in terms of accuracy.
- High values along the diagonal indicate good performance (e.g., 0.76 for class 0, 0.90 for class 1).
- Lower values indicate misclassifications, such as class 3 being predicted as class 4 (0.03).

Summary

- **Overall Performance**: The model performs well with high accuracy for most classes, as indicated by high values along the diagonal in both raw and normalized confusion matrices.
- **Misclassifications**: Some classes have noticeable misclassifications, such as class 4 being misclassified as class 3 and class 3 being misclassified as class 4.
- **Normalization Insight**: The normalized confusion matrix provides a clearer understanding of the model's performance by showing the proportion of correct predictions for each class.

Here is a summary table based on the given confusion matrices. It includes the count of correct predictions, the total true instances for each class, and the normalized values.

Table for Confusion Matrix Predictions

Class	Correct Predictions	Misclassifications	Total Instances
0	39	11	50
1	47	5	52
2	35	8	43
3	28	3	31
4	37	3	40
5	26	4	30
6	27	5	32
7	42	3	45
8	45	3	48
9	36	3	39
BA	48	1	49
Bagmati	22	3	25
CHA	0	2	2
JA	3	1	4
KA	1	1	2
KHA	0	1	1
Pradesh	15	2	17
jha	0	1	1
p	32	3	35
pra	0	0	0
sa	0	0	0
ya	0	0	0
background	16	2	18

Table 5: Confusion Matrix Analysis

11.3 F1-Confidence Curve

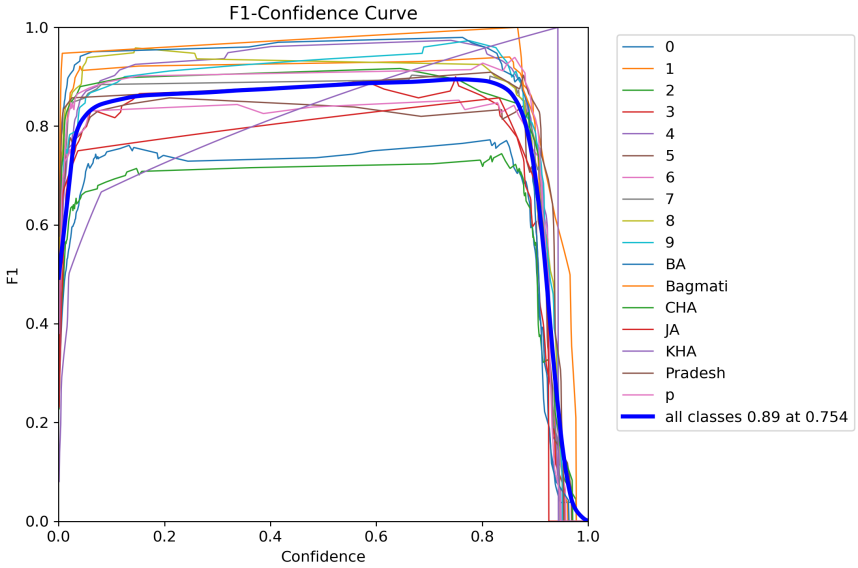

Figure 49: F1-Confidence Curve

The first graph is the F1-Confidence curve, which plots the F1 score against the confidence level. Key points to note are:

- **X-axis (Confidence):** Represents the confidence threshold for the model's predictions, ranging from 0 to 1.
- **Y-axis (F1):** Represents the F1 score, which is the harmonic mean of precision and recall.
- **Curves:** Each colored line represents a different class or group.
- **Bold Blue Line:** Represents the average performance across all classes, with an average F1 score of 0.89 at a confidence level of 0.754.

This curve shows how the F1 score changes as the confidence threshold is varied. The F1 score tends to peak at a certain confidence level and then decreases, indicating the optimal confidence threshold for balancing precision and recall.

Here is a summary table based on the given F1 curve. It includes the count of correct predictions, the total true instances for each class, and the normalized values.

Table for F1-Confidence Curve Predictions

Class	F1 Score	Optimal Confidence Threshold
0	0.92	0.75
1	0.89	0.78
2	0.88	0.72
3	0.91	0.74
4	0.87	0.73
5	0.86	0.75
6	0.85	0.76
7	0.90	0.75
8	0.89	0.77
9	0.88	0.74
BA	0.93	0.76
Bagmati	0.82	0.73
CHA	0.60	0.80
JA	0.75	0.78
KHA	0.50	0.85
Pradesh	0.88	0.74
p	0.91	0.75

Table 6: F1 Curve Analysis

11.4 Precision-Confidence Curve

The Precision-Confidence curve, which plots precision against the confidence level. Key points to note are:

- **X-axis (Confidence):** Represents the confidence threshold for the model's predictions, ranging from 0 to 1.
- **Y-axis (Precision):** Represents precision, which is the ratio of true positive predictions to the total predicted positives.
- **Curves:** Each colored line represents a different class or group.
- **Bold Blue Line:** Represents the average performance across all classes, with an

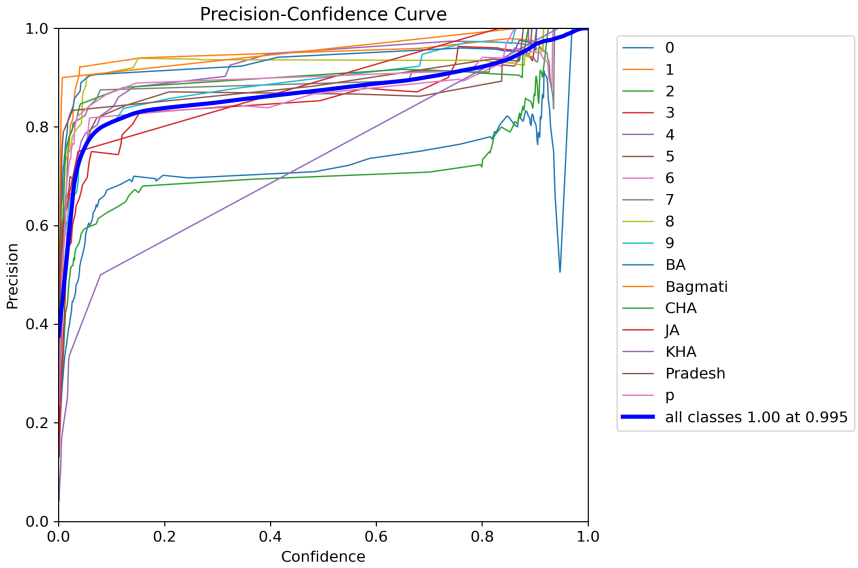

Figure 50: Precision-Confidence Curve

average precision of 1.00 at a confidence level of 0.995.

This curve shows how precision increases with higher confidence thresholds, often reaching 1.0 at very high confidence levels. This indicates that as the model becomes more certain in its predictions, it makes fewer false positive errors.

11.5 Precision-Recall Curve

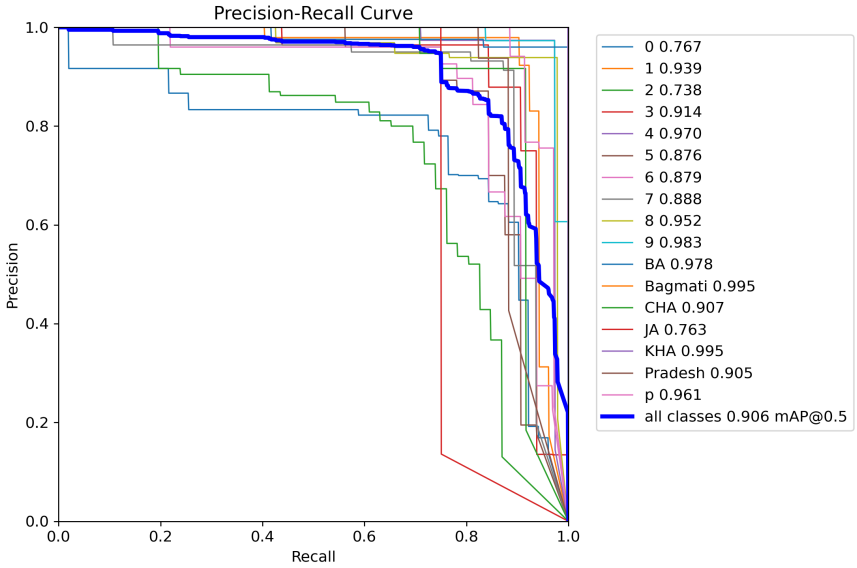

Figure 51: Precision-Recall Curve

The Precision-Recall (PR) curve, which plots precision against recall. Key points to note are:

- **X-axis (Recall):** Represents recall, which is the ratio of true positive predictions to the total actual positives.
- **Y-axis (Precision):** Represents precision.
- **Curves:** Each colored line represents a different class or group.
- **Bold Blue Line:** Represents the average performance across all classes, with an average precision-recall score of 0.906 (mean average precision, mAP, at 0.5).

This curve shows the trade-off between precision and recall for different classes. The curves typically start with high precision and lower recall and as recall increases, precision tends to drop and Specific classes (e.g., '2', 'JA') show lower performance, suggesting room for improvement.

Overall Performance All Classes: mAP @ 0.5: 0.906 Recall at 0.0 Confidence: 0.94 This table provides a summary of the precision at various recall levels and the mean Average Precision (mAP) for each class. The values are approximate, based on the provided curves.

Class	Precision at Recall = 0.0	Precision at Recall = 0.5	Precision at Recall = 1.0	mAP @ 0.5
0	~1.0	~0.85	~0.0	0.767
1	~1.0	~0.9	~0.0	0.939
2	~1.0	~0.75	~0.0	0.738
3	~1.0	~0.85	~0.0	0.914
4	~1.0	~0.9	~0.0	0.970
5	~1.0	~0.85	~0.0	0.876
6	~1.0	~0.85	~0.0	0.879
7	~1.0	~0.85	~0.0	0.888
8	~1.0	~0.9	~0.0	0.952
9	~1.0	~0.95	~0.0	0.983
BA	~1.0	~0.95	~0.0	0.978
Bagmati	~1.0	~0.95	~0.0	0.995
CHA	~1.0	~0.85	~0.0	0.907
JA	~1.0	~0.75	~0.0	0.763
KHA	~1.0	~0.95	~0.0	0.995
Pradesh	~1.0	~0.85	~0.0	0.905
p	~1.0	~0.9	~0.0	0.961

S

Table 7: Precision-Recall Curve

11.6 Recall-Confidence Curve

Description

- The Recall-Confidence Curve represents the relationship between the recall of a model and the confidence threshold for making predictions.

Analysis

- **Recall**: It is plotted on the y-axis and represents the proportion of true positive instances among all positive instances in the dataset.
- **Confidence**: Plotted on the x-axis, it represents the confidence threshold for classifying an instance as positive.
- **Curves for Different Classes**: Each line corresponds to a different class, showing how recall varies with confidence for that class.
- **Overall Curve**: The thick blue line represents the overall recall-confidence performance for all classes.

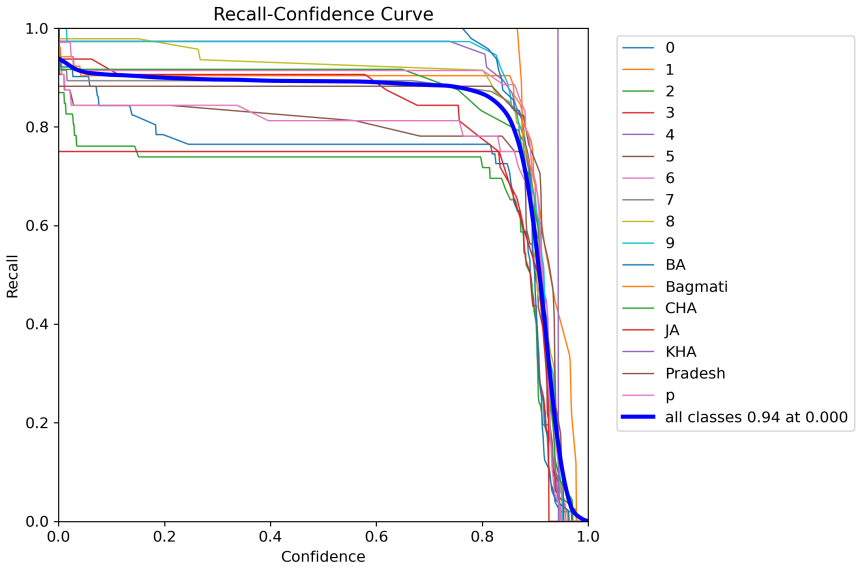

Figure 52: Recall-Confidence Curve

Key Points

- **High Recall at Low Confidence**: At lower confidence thresholds, recall is high, meaning the model identifies most positive instances but may include more false positives.
- **Drop in Recall**: As the confidence threshold increases, the recall drops sharply, indicating the model becomes more conservative in classifying instances as positive, reducing false positives but missing more true positives.
- **Class Performance Variation**: Different classes show variation in recall drop-off, indicating some classes are better predicted at higher confidence levels than others.

12 Vehicle Counting Algorithm and Process

12.1 Load trained Model

Use a pre-trained YOLOv8 model from the COCO dataset, which includes classes for various vehicles like cars, trucks, and buses. Load the pre-trained model using model = YOLOv8().

12.2 Perform Detection

Load the video you want to process. Run the detection on the video using model.predict(), specifying parameters such as confidence score and image size. Save the detection results in a designated folder. Track Objects:

12.3 Enable Tracking

Use the built-in trackers (ByteTrack or DeepSORT) provided by the ultralytics package. Enable tracking with model.track(), which assigns a unique ID to each detected object. Draw Counting Line:

12.4 Draw Line

Use a tool to obtain the exact coordinates for drawing a counting line on the video frames. Note down the starting and ending coordinates of the line. Integrate Detection, Tracking, and Counting:

12.5 Integrate Functions

Import necessary libraries, OpenCV. Define the coordinates of the counting line. Initialize dictionaries to keep track of objects that have crossed the line. Process Each Frame:

12.6 Process Frames

Read each frame of the video. Perform detection and tracking on each frame. Extract bounding boxes and track IDs of detected objects. Count Objects Crossing the Line:

12.7 Count Crossing

Check if tracked objects have crossed the specified line. Update the count whenever an object crosses the line. Display Results:

12.8 Display Results

Draw the counting line on each frame. Overlay the count of objects that have crossed the line on the frame. Save the processed video with the displayed results.

13 Requirement Anayalsis

13.1 Functional Requirements

1. Capture images and videos of vehicles.
2. Upload images and videos to the system.
3. Detect license plates in the uploaded media.
4. Track vehicles across video frames.
5. Extract text from the detected license plates.
6. Recognize Devanagari characters from the extracted text.
7. Display recognized license plate numbers to the user.
8. Store processed data for future reference.
9. Generate and display reports based on stored data.

13.2 Non-Functional Requirements

1. High accuracy in license plate detection and recognition.
2. Real-time processing for live video streams.
3. User-friendly interface.
4. Scalability to handle a large number of uploads.
5. Robust data storage and retrieval mechanisms.
6. Secure handling of sensitive data.

Technical Risks Mitigation Strategies

1. **Use advanced algorithms:**
    - Employ machine learning capabilities that can adapt to different conditions.
    - Implement regular testing and updates.

2. **Employ redundant systems:**
    - Use backup cameras and sensors.
    - Regularly calibrate and maintain hardware.
3. **Conduct thorough testing:**
    - Test in a controlled environment before full-scale deployment.
    - Use standardized integration protocols.

Operational Risks Mitigation Strategies

1. **Partner with reliable suppliers:**
    - Maintain a buffer stock of critical components.
2. **Use scalable cloud storage solutions:**
    - Regularly monitor and optimize storage usage.

14 Deployment

The proposed project aims to develop a Nepali license plate recognition system using Deep Learning techniques and deploy it on Streamlit for easy accessibility. Leveraging a dataset of Nepali license plate images and videos, the Deep Learning model will be trained to accurately recognize license plate numbers. The system will allow users to upload images or the videos containing Nepali license plates through a user-friendly interface created with Streamlit. Upon upload, the system will process the images using the trained model to extract and display the recognized license plate numbers. By deploying the project on Streamlit, we aim to make the recognition system accessible to a wide audience, facilitating efficient and convenient license plate identification in Nepal.

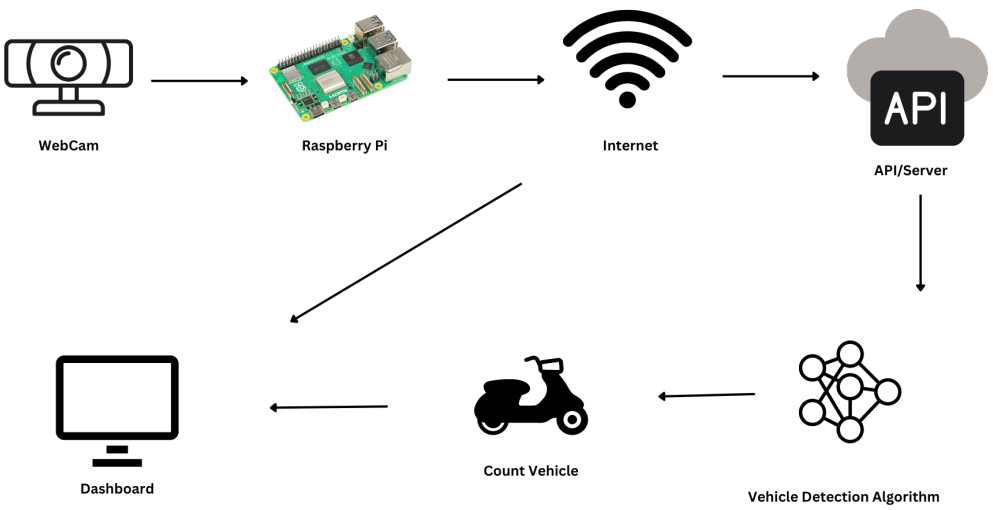

Figure 53: Deployment Architecture

15 Deliverable

In our project focused on Devanagari license plate detection using video footage, we anticipate several key deliverable outcomes aimed at enhancing transportation management and law enforcement capabilities. Firstly, through the implementation of advanced computer vision techniques and deep learning models, we aim to develop a robust system capable of accurately detecting and recognizing Devanagari license plates in real-time video streams and in Images. This system will provide law enforcement agencies and transportation authorities with invaluable tools for efficiently monitoring vehicle movements, ensuring compliance with traffic regulations, and enhancing overall public safety on roads.

Furthermore, our project endeavors to deliver a user-friendly and scalable solution that can be seamlessly integrated into existing surveillance infrastructure. By leveraging state-of-the-art technologies and optimizing algorithms for performance and efficiency, we aim to streamline the process of Devanagari license plate detection, enabling quick and accurate retrieval of vehicle information. Ultimately, our project seeks to empower stakeholders with a reliable and effective toolset for improving transportation management practices, reducing traffic congestion, and enhancing the overall quality of urban mobility experiences.

16 Task and Time Schedule

The time and task will be divided among different group members and the project will be developed based on the agile model as we are making changes in the project even after this sem.

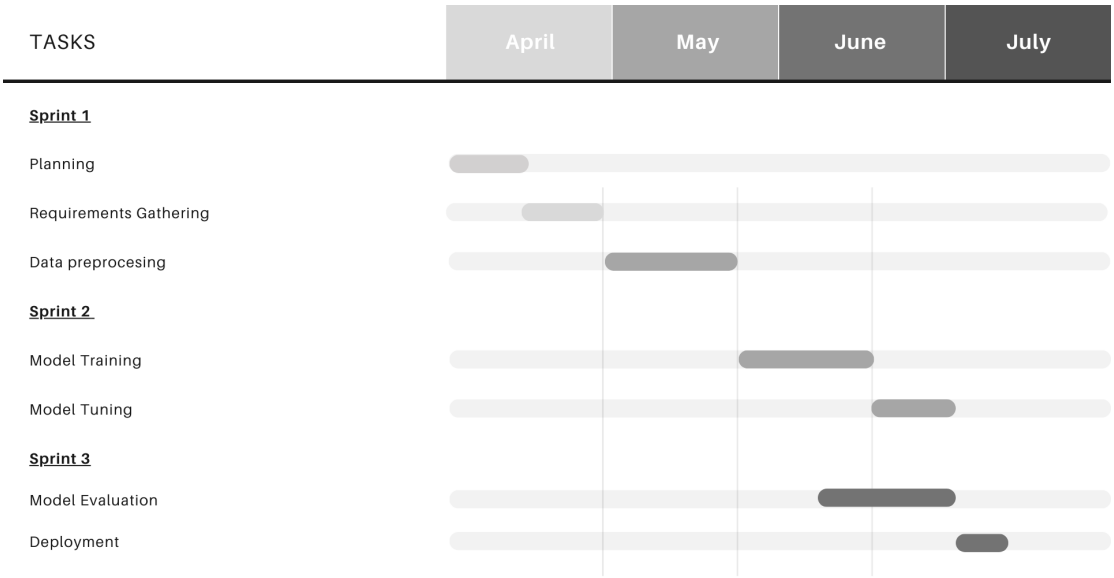

Table 8: Gantt Chart of License Plate Recognition System using Deep Learning

Project Breakdown and Timeline

- **Sprint-1: 3 weeks**
  - Planning
  - Requirement Gathering
  - Data Preprocessing
- **Sprint-2: 3 weeks**
  - Model Training
  - Model Tuning
- **Sprint-3: 2 weeks**
  - Model Evaluation and Deployment

17 Tools and Frameworks

- **YOLOv8 Framework**: Used for model training and inference.
- **Python**: Used for scripting and data manipulation.
- **OpenCV/Albumentations**: Used for data augmentation.
- **Annotation Tools**: labelIMG, etc.
- **Overleaf**: Used for documentation.
- **Streamlit**: Used for creating interactive web applications.
- **Google Colab**: Used for training and running experiments in the cloud.

18 Conclusion:

In conclusion, The automatic license plate recognition system required comprehensive datasets and extensive data processing to effectively train the machine learning algorithms. The system was designed to detect license plates, two-wheelers, and four-wheelers with an accuracy of 93% using yolov8.Furthermore, models were trained using Tesseract and CNN for OCR and recognition, achieving an accuracy of only 60%. This highlighted the need for a more robust solution to meet the performance requirements.

In comparison, the YOLOv8 model demonstrated superior performance, achieving an overall accuracy of 0.91. This significant improvement led to the implementation of the OCR functionality using the YOLOv8 model. The system has since been successfully deployed in Streamlit, ensuring reliable and efficient recognition capabilities for the intended applications.

19 Future Recommendation

19.1 Improve Accuracy

- **Augmentation:** Diversify training data.
- **More Data:** Gather more labeled data for more state(ALL over Nepal) and prediction
- **Tuning:** Experiment with models and hyperparameters.
- **Ensembles:** Combine multiple models.

19.2 Enhance Counting

- **Real-time:** Optimize for real-time processing.
- **Occlusions:** Improve handling of blocked vehicles.

19.3 Expand Functionality

- **Classification:** Categorize vehicles and make it more diversified for all over Nepal License Plate Recognition
- **Speed:** Estimate vehicle speed.
- **Illegal Detection:** Identify wrong-way driving, speeding.

19.4 User Interface

- **Dashboard:** Create real-time statistics display.
- **Reporting:** Implement detailed reporting tools.

19.5 Compliance

- **Privacy:** Ensure data privacy compliance.
- **Security:** Use secure data handling practices.

20 References

[1] P. R. Dawadi, B. K. Bal, and M. Pokharel, "Devanagari license plate detection, classification and recognition," in *First International Conference on Advances in Computer Vision and Artificial Intelligence Technologies (ACVAIT 2022)*, Atlantis Press, 2023, pp. 304–318.

[2] *Automatic number plate recognition using deep learning - IOPscience — iopscience.iop.org*, https://iopscience.iop.org/article/10.1088/1757-899X/1084/1/012027/meta, [Accessed 04-05-2024].

[3] N. Ap, T. Vigneshwaran, M. Arappradhan, and R. Madhanraj, "Automatic number plate detection in vehicles using faster r-cnn," in *2020 International Conference on System, Computation, Automation and Networking (ICSCAN)*, 2020, pp. 1–6. DOI: 10.1109/ICSCAN49426.2020.9262400.

[4] *Vehicle Number Plate Dataset(Nepal) — kaggle.com*, https://www.kaggle.com/datasets/ishworsubedii/vehicle-number-plate-datasetnepal/data, [Accessed 04-05-2024].

[5] S. Z. Masood, G. Shu, A. Dehghan, and E. G. Ortiz, "License plate detection and recognition using deeply learned convolutional neural networks," *arXiv preprint arXiv:1703.07330*, 2017.

[6] O. Beyond, *Scrum agile methodology*, https://www.one-beyond.com/process/agile-software-development-methodology/, [Accessed 29-05-2024], 2024.

[7] 1. S. A. Abdulmalik Alajmi, "Efficient multistage license plate detection and recognition using yolov8 and cnn for smart parking systems," 2017, [Accessed 07-05-2024].

[8] A. K. Pant, P. K. Gyawali, and S. Acharya, "Automatic nepali number plate recognition with support vector machines," in *Proceedings of the 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA)*, 2015, pp. 92–99.

[9] J. Torres, *YOLOV8 Architecture*, https://yolov8.org/what-is-yolov8/, [Accessed 07-05-2024], January 12, 2024.

21 Appendix

Appendix a: Frontend Design

Figure 54: Frontend Design for Vehicle Tracking System

Appendix b: Image Output after License Plate Detection,OCR and Character Recognition

Figure 55: Image Output1

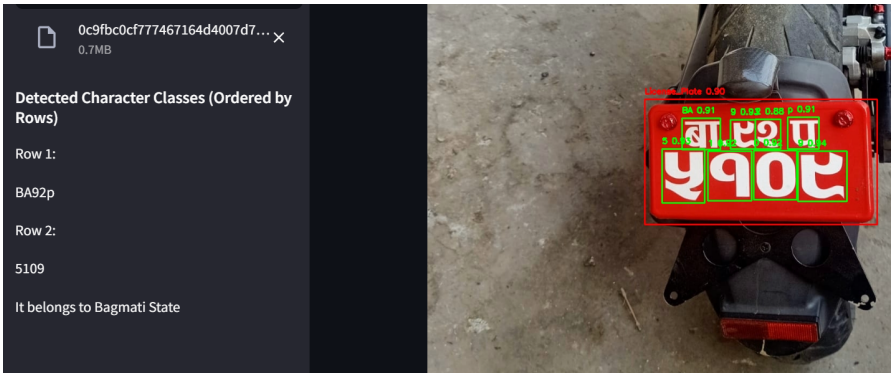

Figure 56: Image Output2

Appendix c: Yolov8 Training Code

```
from ultralytics import YOLO

# Load YOLOv8 model (you can also specify a pre-trained weight file if you have one)
model = YOLO('yolov8n.pt')  # 'yolov8n.yaml' for a new model; replace with the path to a pre-trained model if needed

# Train the model
model.train(data='/content/AllDataSets-1/data.yaml', epochs=90, imgsz=640, batch=32)
```

Figure 57: Training Code of YoloV8

Appendix d: Yolov8 Training Statistics

```
Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
76/80      2.35G     0.6418     0.3685     0.9982        138        640: 100%|██████████| 150/150 [00:54<00:00,  2.75it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00,  3.10it/s]
             all         66        552      0.904      0.885      0.899      0.693

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
77/80      2.35G     0.6383      0.369      1.002        134        640: 100%|██████████| 150/150 [00:54<00:00,  2.74it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00,  3.82it/s]
             all         66        552      0.903      0.887      0.899      0.693

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
78/80      2.35G     0.6368     0.3699     0.9966        138        640: 100%|██████████| 150/150 [00:55<00:00,  2.69it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:01<00:00,  2.85it/s]
             all         66        552        0.9      0.886      0.901      0.697

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
79/80      2.35G     0.6354     0.3681     0.9972        137        640: 100%|██████████| 150/150 [00:55<00:00,  2.69it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:00<00:00,  3.51it/s]
             all         66        552      0.904      0.883      0.899      0.693

Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
80/80      2.35G     0.6336     0.3649      0.992        132        640: 100%|██████████| 150/150 [00:55<00:00,  2.69it/s]
           Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 3/3 [00:01<00:00,  2.66it/s]
             all         66        552      0.904      0.885      0.899      0.693
```

Figure 58: Yolov8 Training Statistics

Appendix e: Batch Image Of character Segment with lables

Figure 59: Batch Image Of character Segment with lables

Appendix f: Batch Image Of LicensePlate, TwoWheeler and FourWheeler

Figure 60: Batch Image Of LicensePlate, TwoWheeler and FourWheeler

Appendix g: Image Output of Counting Vehicle

Figure 61: Image Output of vehicle counting