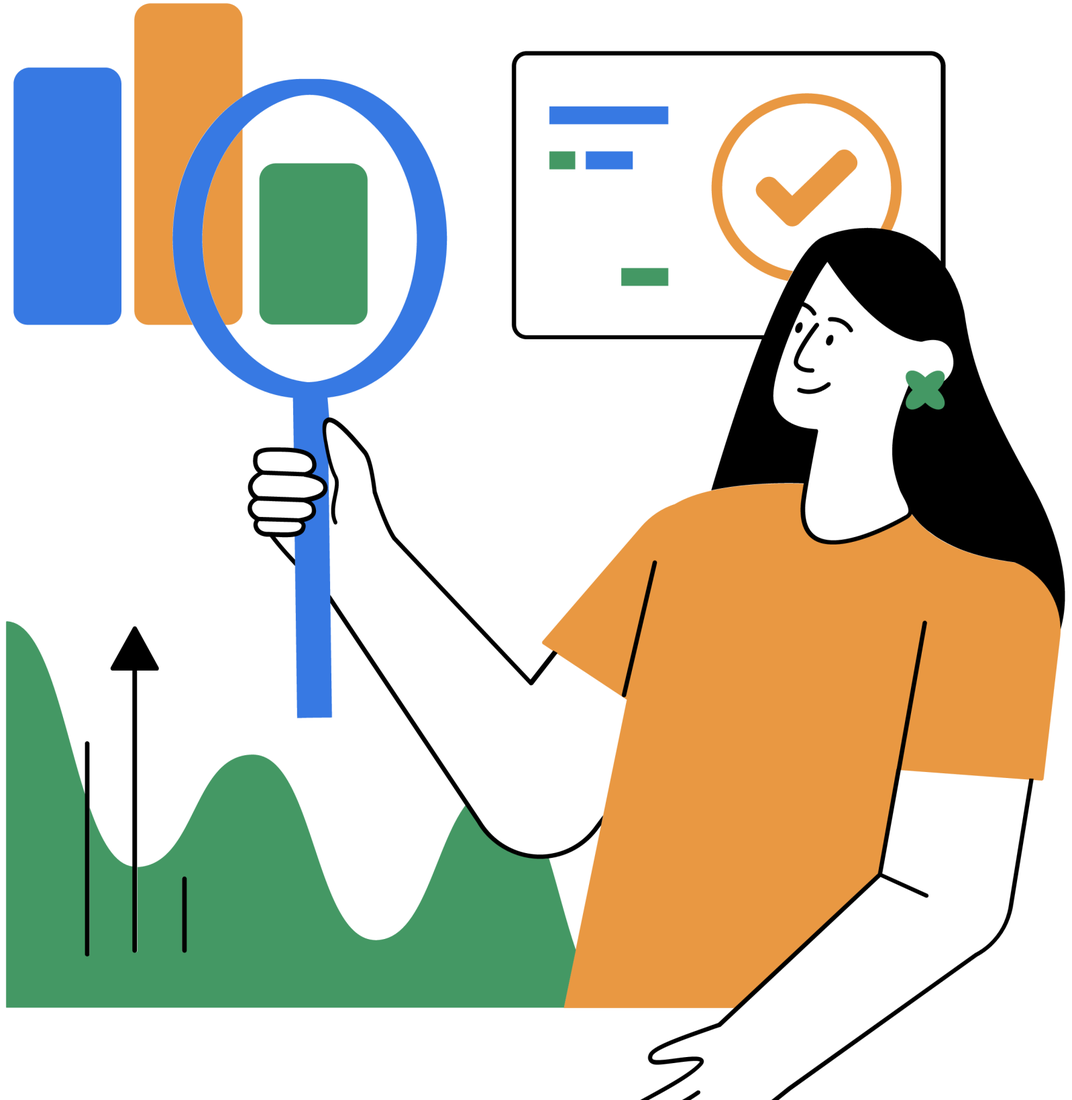
PROGRAMMING FOR DATA SCIENCE

Topic: Car License Plate Detection

Lecturer: Mr. KEAN VESAL

Presented by Group 1:

PHYRUN Pichchhorda	e20220895
NOV PANHAVATH	e20221643
PRIM Veasna	e20221402
NOB Sreynich	e20220741
LUN Rathana	e20220368



Content

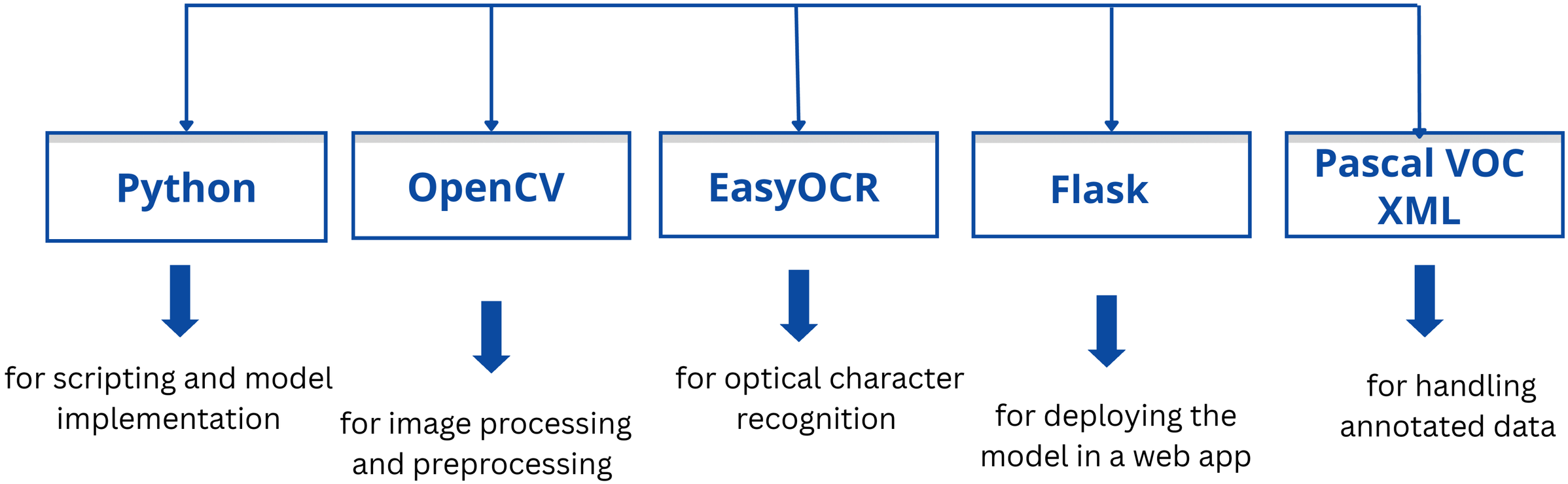
1. Introduction

- Car plate detection and recognition are key for smart transport, security, and vehicle tracking.
- The project aims to develop a system for detecting vehicle license plates in images and extracting their characters using Optical Character Recognition (OCR)
- The main challenge is accurately detecting plates despite lighting, angles, and image quality, then correctly extracting characters for uses like database logging and monitoring.
- This project combines image processing, object detection, and OCR to detect plates and extract characters from the cropped regions.

Objective

- To develop a simple and efficient car plate detection system
- To apply image preprocessing for better OCR accuracy
- To extract readable text from plates using EasyOCR
- To optionally deploy the system using Flask for interactive testing

Tools and Technologies



2. Dataset Overview

Dataset Structure

- dataset/
  - train/
    - images/
    - annotations/
  - val/
    - images/
    - annotations
  - test/
    - images/
    - annotations/

Image of car:

Annotation Format

Each .xml file contains:

- Image filename and size
- Object class (e.g., "plate")
- Bounding box (xmin, ymin, xmax, ymax)

This structure allows the model to learn where the plates are located and apply detection algorithms accordingly.

Challenges in the Dataset

- Variations in lighting and shadows
- Low-resolution or blurred plates
- Plates at different angles or partially occluded
- Multiple vehicles in a single image (some noisy samples)

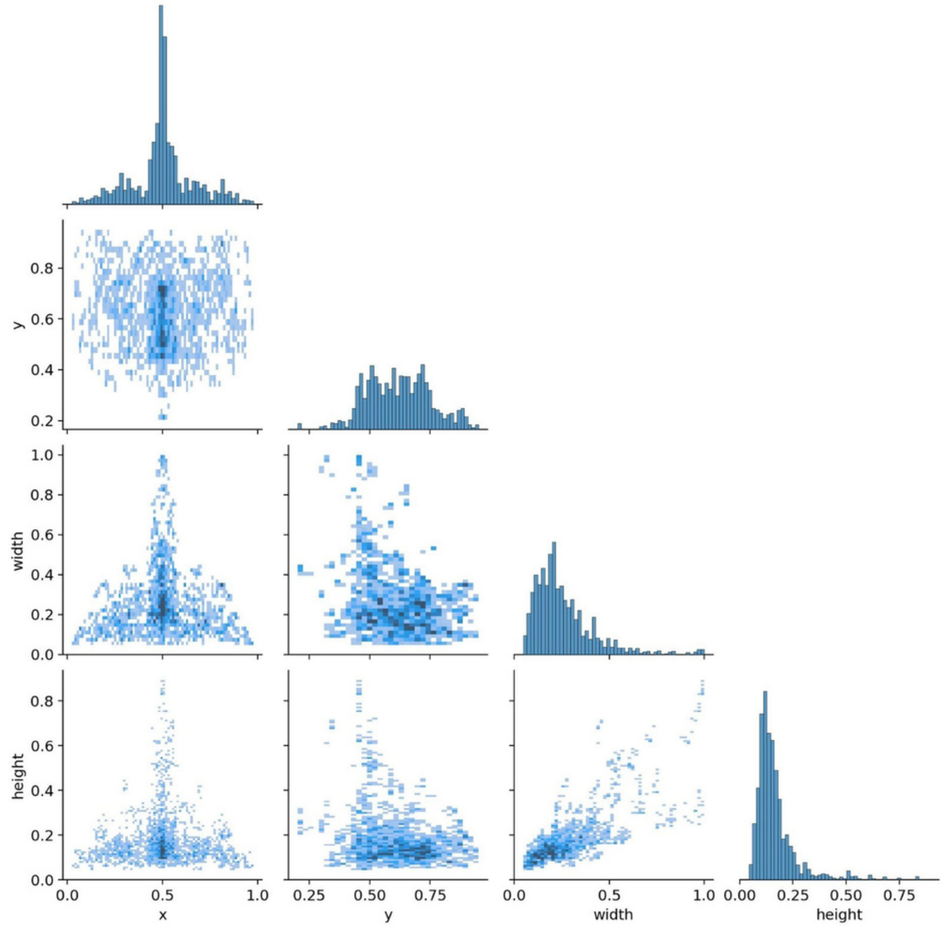
To handle these challenges, preprocessing and filtering steps were applied to improve the model’s performance and OCR accuracy.

3. Methodology

Preprocessing

- **Cropping:** Cut out the plate area
- **Resizing:** Make it a fixed size standard size (e.g., 300×100)
- **Grayscale Conversion**: Convert to grayscale
- **Bilateral Filtering:** Applied to preserve edges while smoothing noise
- **Contrast Enhancement (CLAHE):** Improve contrast in dark areas

	filename	width	height	class	xmin	ymin	xmax	ymax
0	Cars0.png	500	268	licence	221	120	424	178
1	Cars1.png	400	248	licence	129	123	267	165
2	Cars10.png	400	225	licence	135	0	308	153
3	Cars100.png	400	267	licence	170	109	219	136
4	Cars101.png	400	300	licence	162	197	245	225



3. Methodology

Plate Detection

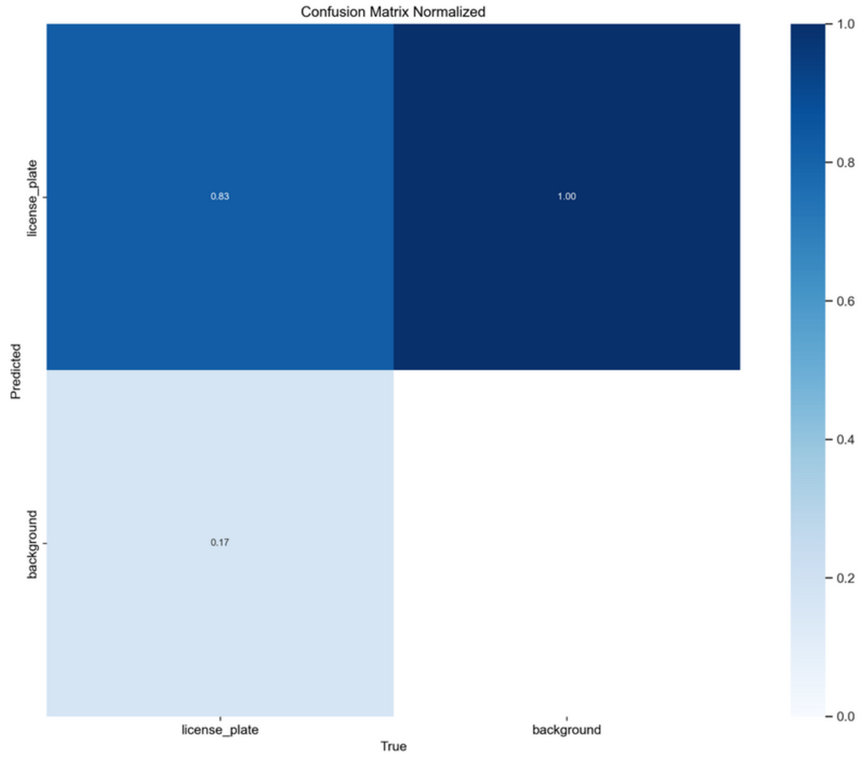
- Annotation Usage: Used XML files for true bounding boxes
- Bounding Box Extraction: Extracted coordinates to crop plates
- YOLO or Manual Cropping (Depending on project): Used YOLO for detection or manual cropping

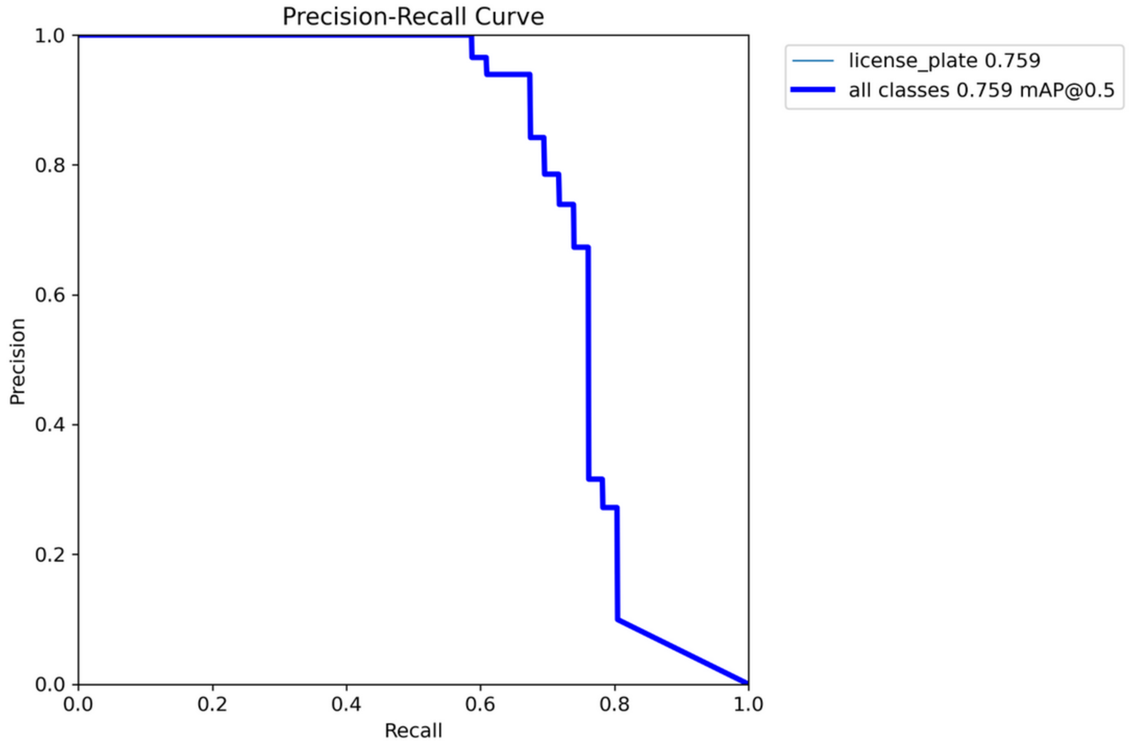
OCR and Recognition

- Tool Used: EasyOCR
- Character Filtering: Only uppercase alphabets and numbers were allowed using allowlist="ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789".
- Output: The OCR result returns the plate number as text.

Model Evaluation

Data training





4. Results

Detection and OCR Output

Original Image:

Cropped Plate Region:

Detected Plate Text: MH12NE8922

4. Results

Sample Results

Image 1

(Plate Crop 1)

Detected Text: DZI7YXR

Sample Results

Image 2

(Plate Crop 2)

Detected Text: DL7CN5617

Image 3

(Plate Crop 3)

Detected Text: 9214

Error Cases

In some cases, the OCR results were incorrect due to:

- Low contrast or dark images
- Blurry or tilted plates
- Complex backgrounds or small plate size

Expected: P018299
OCR Output: 4012927

5. Deployment

Tools and Technologies

Flask – Used to create the web server and routing logic

OpenCV – Handles image preprocessing

EasyOCR – Extracts text from detected plates

HTML/CSS – Builds the front-end user interface

Werkzeug/File Handling – Manages file uploads

Application Workflow

1. The user opens the web app in a browser

2. The user uploads an image of a vehicle

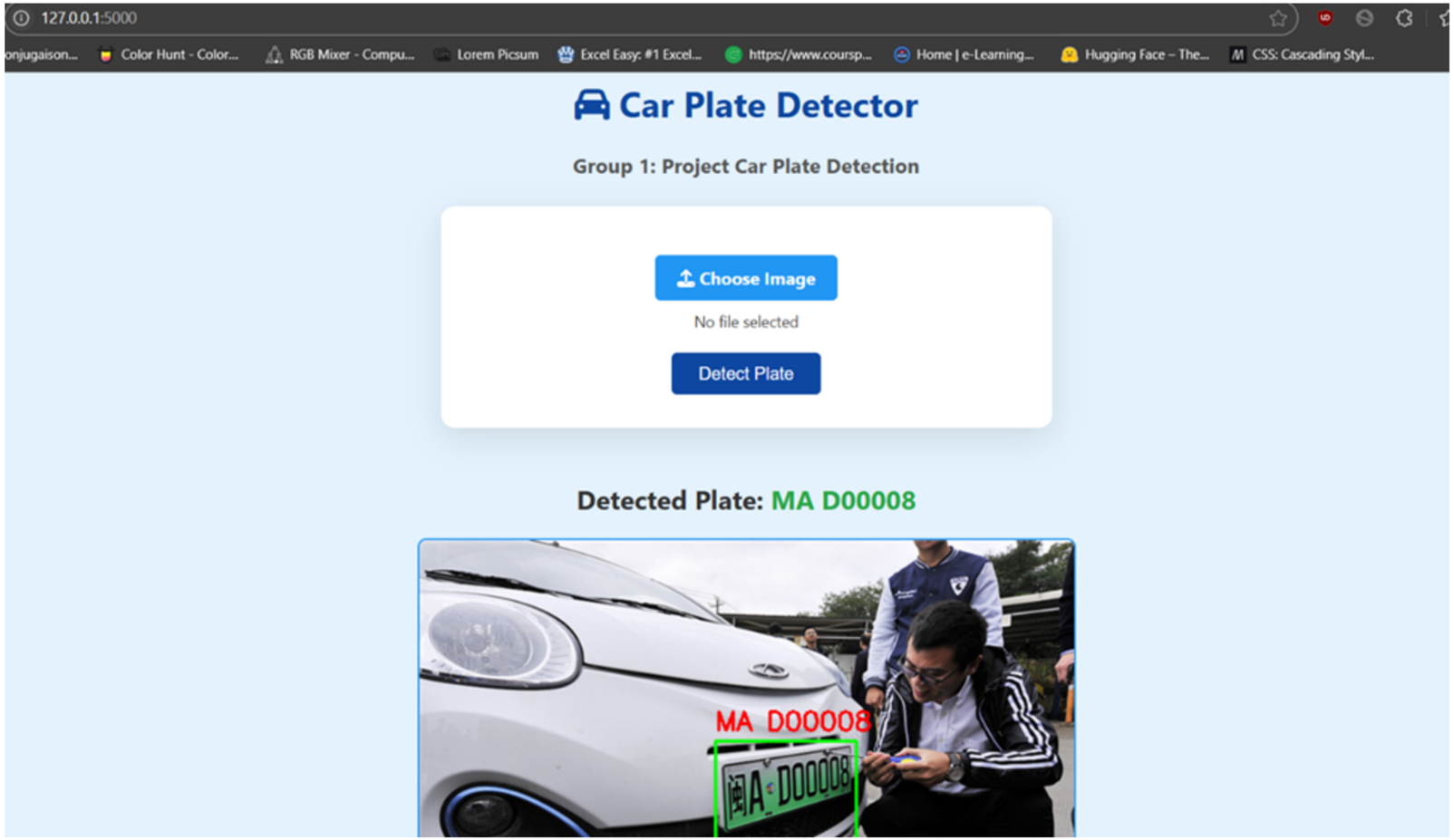
3. The image is processed to detect the car plate using a trained model

4. The detected plate region is cropped and passed to EasyOCR

5. The recognized plate number is displayed on the web page along with the detection box

USER INTERFACE

- The web app includes:
  - An image upload section
  - A result display area showing the detection box and OCR result.



6. Discussion

What Worked Well

- YOLOv8 detection model
- Image preprocessing
- EasyOCR
- Flask web app

Limitations

- OCR Accuracy Drops
- No Multiple Plate Detection
- Model Sensitivity

Opportunities for Improvement

- Data Augmentation
- Post-processing OCR
- Multiplate Support
- Deployment Upgrade

7. Conclusion

The project built a car plate detection and recognition system using Python, YOLOv8, OpenCV, and EasyOCR.
It detects plates and reads text accurately. The process includes

- preprocess images
- detect plates with YOLO
- read text with EasyOCR
- optionally use Flask for interaction.

The system worked well on clear images but struggled with poor-quality scenes. With improvements like error correction and tuning, it could be used in traffic, parking, and security systems.

Thank You