LICENSE PLATE RECOGNITION USING YOLO AND CNN

TEAM MEMBERS

CH.VAISHNAVI KRISHNA-CB.SC.U4AIE24108

NETHRA.R-CB.SC.U4AIE24147

SAHAANA SHRI.SK-CB.SC.U4AIE24149

MOUNIKA.CH-CB.SC.U4AIE24169

INTRODUCTION:

➜ License Plate Recognition (LPR) uses YOLO to quickly detect plates and CNN to accurately recognize characters.

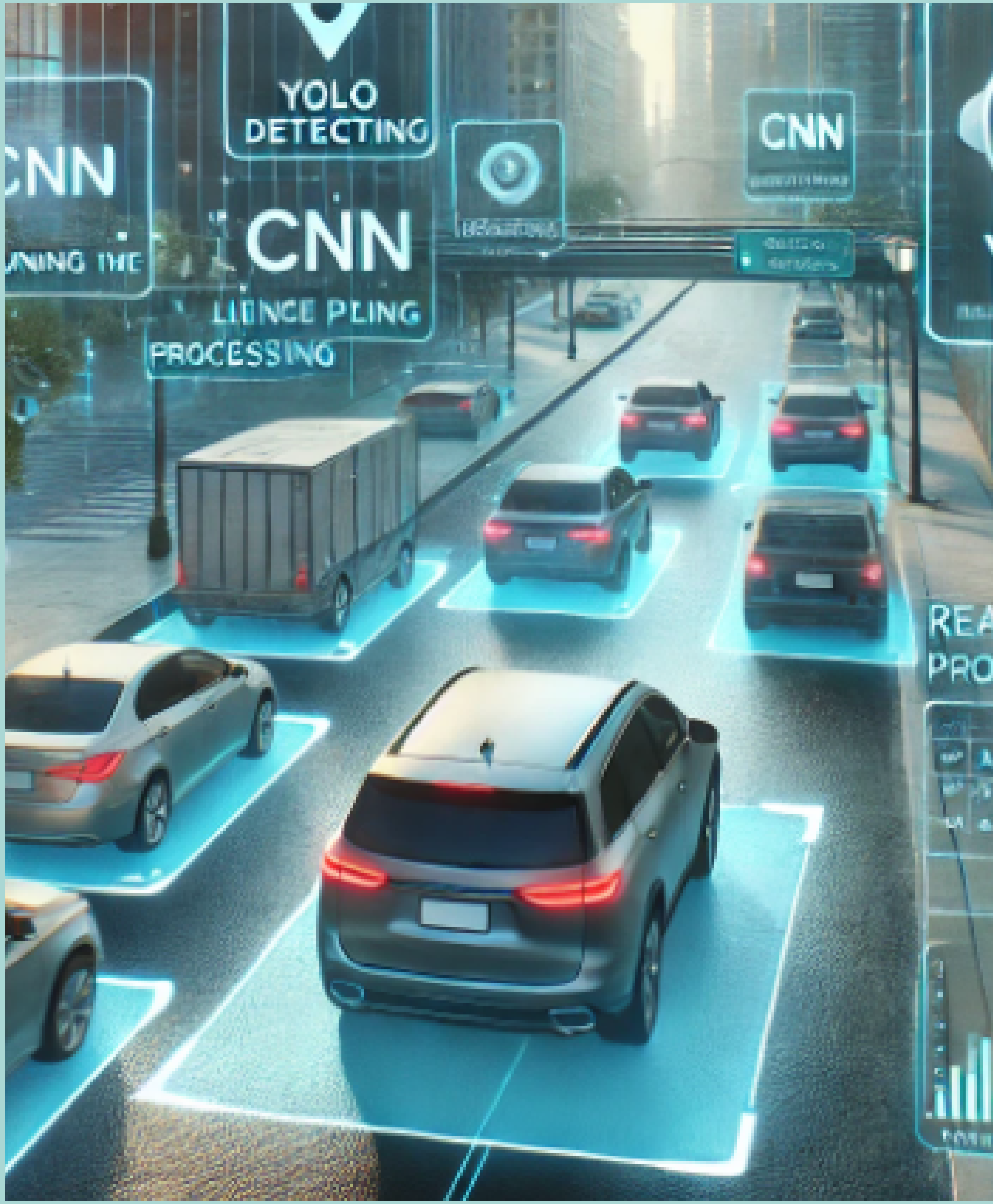
➜ This combination ensures fast and reliable vehicle identification for traffic monitoring and security.

➜ It is widely used in toll systems, law enforcement, and
smart city applications.

OBJECTIVES:

➜ Accurately detect license plates using YOLO in real-time.

➜ Recognize characters on the plate using a CNN model.



METHODOLOGY

DETECTING BOUNDING BOXES USING YOLO :

1.Image Division: The image is divided into a grid (e.g., 13x13). Each cell in this grid is responsible for detecting objects if the object's center is within the cell.

2. Anchor Boxes: These predefined boxes are applied to each grid cell to help predict the shape and size of objects.

3.Bounding Box Prediction to yolo format:

Each grid cell predicts multiple bounding boxes with:

- Center (x, y): Where the box is on the image.
- Width and Height: How big the box is.

4.Class Prediction: The model predicts the type of object in each grid cell. Each grid cell predicts the probability of objects (like license plates).

5.Confidence Score: How sure the model is that an object is present in the box.

Purpose: Provides the basic information needed to draw a box around an object.

6.Non-Max Suppression (NMS):

- YOLOv8 uses NMS to filter out redundant boxes.
- Keeps the box with the highest confidence score and removes boxes with high IoU (overlapping) to avoid multiple detections of the same object.

To measure how well the predicted box overlaps the actual object:

IoU=Area of Union/Area of Overlap

7.Single Forward Pass:

The entire process of detecting objects and drawing bounding boxes happens in one quick step through the model.

8.Final Output:

The model gives the final bounding boxes with class labels and confidence scores.

PREPROCESSING:

Why is Preprocessing Important?

- When we capture images for License Plate Recognition (LPR), they often contain **noise, distortions, or unwanted details like background clutter, shadows, or reflections.**
- These raw images can interfere with detection and recognition, leading to misclassification or **poor accuracy in our model**.
- Preprocessing **cleans and standardizes** the images, ensuring they are in the same format, size, and quality, which improves model performance.
- It **enhances feature extraction**, making important details like edges and characters more visible for better recognition.

PREPROCESSING TECHNIQUES:

1.License Plate Detection & Cropping

- Function Used: YOLO label files with bounding box conversion

Importance: Extracts the license plate region from full car images for further processing.

2. Resizing & Normalization

- Function Used: cv2.resize(cropped_plate, (128, 64))

Importance: Standardizes image dimensions to ensure consistent input for deep learning models.

- Function Used: resized_plate.astype(np.float32) / 255.0

Importance: Scales pixel values to [0,1], improving model stability and convergence.

3. Grayscale Conversion:

- Function Used: cv2.cvtColor(resized_plate, cv2.COLOR_RGB2GRAY)

Importance: Converts the RGB image to grayscale, reducing computational complexity and improving character recognition by enhancing contrast

.

4.Noise Reduction:

- Function Used:cv2.fastNlMeansDenoising(gray_plate, None, 30, 7, 21)

Importance: Removes noise while preserving important edge details, making it easier for character segmentation and recognition.

5. Binary Conversion (Thresholding):

- Function Used: cv2.threshold(denoised_plate, 127, 255, cv2.THRESH_BINARY)

Importance: Converts the image into a black-and-white representation, making character regions more distinguishable from the background, which is essential for OCR-based recognition.

5. Segmentation:

- Function Used: cv2.findContours(binary_img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

Importance: Extracts individual character regions by detecting contours and bounding boxes.

How Preprocessing Improves Accuracy

- Removes unwanted noise and background – Helps the model focus on relevant features.
- Ensures uniform input – Prevents variations in size and quality from affecting predictions.
- Enhances edge and character clarity – Leads to better segmentation and feature extraction.
- Improves model generalization – Clean input allows CNN and YOLO to learn effectively, reducing misclassification.
- Boosts final recognition accuracy – Well-preprocessed images result in more reliable license plate detection and character recognition.

CNN

INPUT LAYERS:

➤ Data Handling: Processes raw images(224*224*3)

➤ Normalization: Scales pixel values (0-1) for better model performance.

CONVOLUTIONAL LAYER:

➤ Extracts features (edges, textures) using filters(kernels)

➤ Multiplies filter with input, sums values, and adds bias b.

ACTIVATION LAYER (RELU):

➤ReLU makes negatives 0 and keeps positives, helping learn complex patterns by introducing non-linearity

$$f(x) = max\ (0, x)$$

Pooling Layer:

- reduce the spatial dimensions of feature maps while preserving the most important features.
- Selects the maximum value from each region of the feature map.

Output layer:

- Flattening: Converts the 2D feature maps into a 1D vector.
- Fully Connected (Dense) Layer: Performs final decision-making.
- Softmax → Produces probabilities for each class.

In our code we are using the following functions:

```
Conv2D(32, (3,3), activation='relu', input_shape=(224, 224, 3)),
MaxPooling2D(pool_size=(2,2)),
Flatten(),
Dropout(0.5),
Dense(36, activation='softmax')
```

We are using EasyOCR for character recognition which is an inbuilt library for Optical Character Recognition

PROPOSED TIME LINE:

Week 1-2: *Research & Planning*
Study existing LPR techniques
Collect datasets and labe l images.

Week 3-4: License Plate Detection (YOLO)
Train YOLO on labeled license plate data.

Week 5-6: Character Recognition (CNN)
Train CNN for character classification.
Preprocess images

Week 7-8: Integration & OCR
Combine YOLO and CNN into a single pipeline.
Implement OCR for final text extraction.

Week 9: Testing & Optimization
Improve accuracy (handle poor lighting, blurred images).
Optimize for real-time video processing.

Week 10:
Final Review & Report

RESULTS:

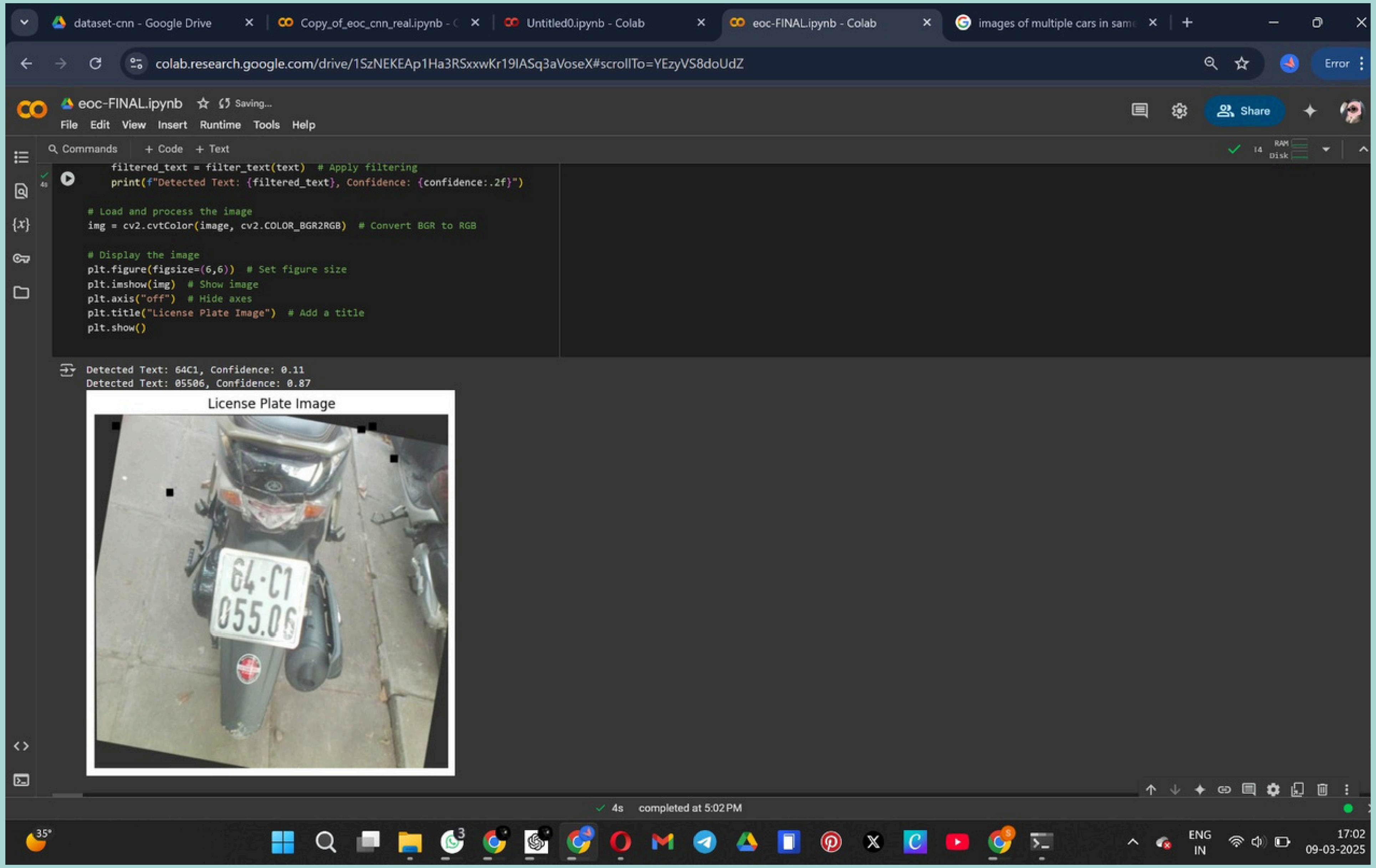
➤*YOLO OUTPUT:*

```python
from google.colab import drive
drive.mount('/content/drive')
import cv2
import matplotlib.pyplot as plt

image_path = "/content/drive/MyDrive/license_plate_recognition/images/train/Cars429.png"  # Change to an actual image
label_path = "/content/drive/MyDrive/license_plate_recognition/labels/train/Cars429.txt"

img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
h, w, _ = img.shape

with open(label_path, "r") as f:
    data = f.readline().split()
    _, x_center, y_center, width, height = map(float, data)

x_min = int((x_center - width / 2) * w)
y_min = int((y_center - height / 2) * h)
x_max = int((x_center + width / 2) * w)
y_max = int((y_center + height / 2) * h)

plt.imshow(img)
plt.gca().add_patch(plt.Rectangle((x_min, y_min), x_max - x_min, y_max - y_min, edgecolor='red', linewidth=2, fill=False))
plt.show()
```

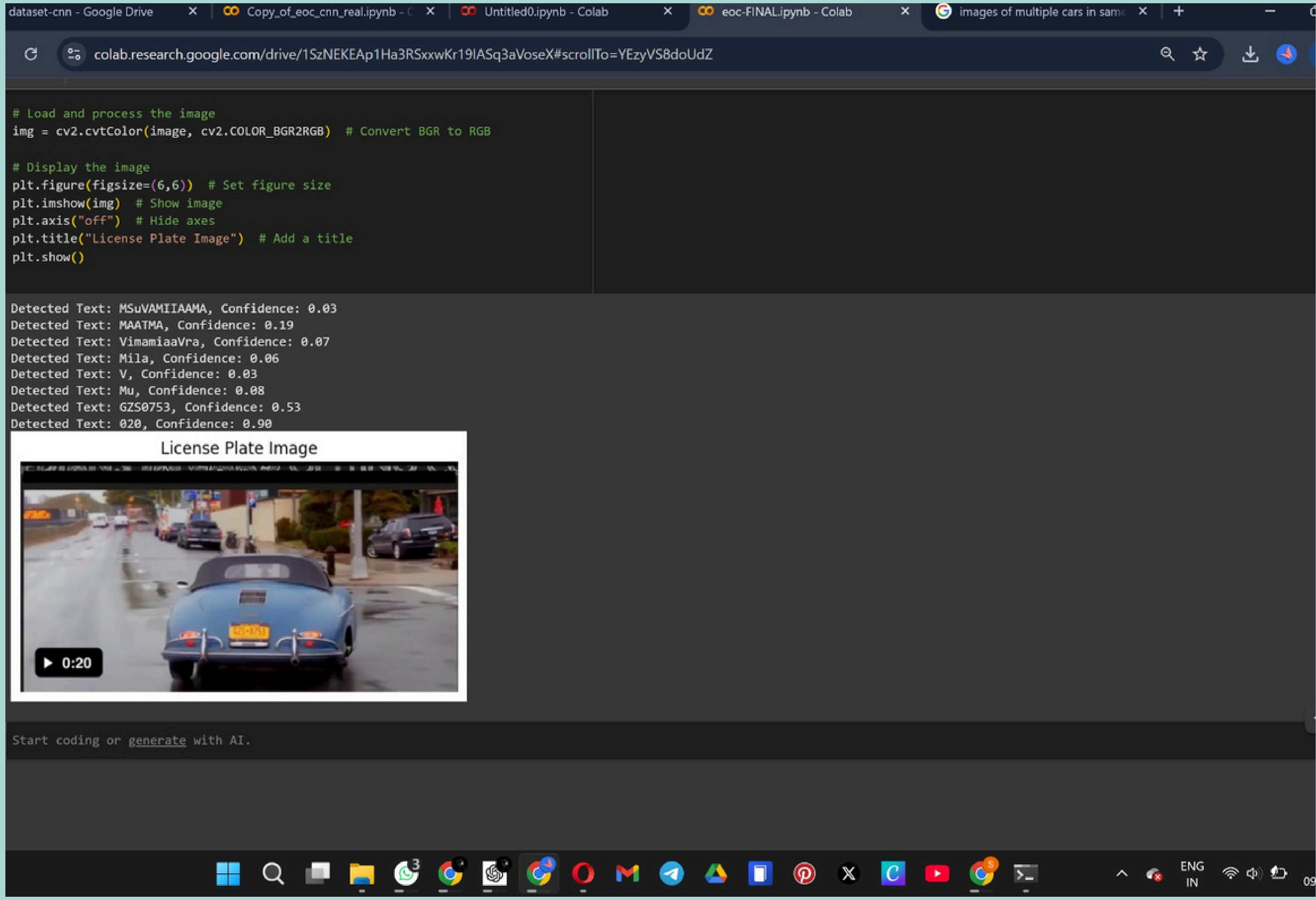
Mounted at /content/drive

PREPROCESSING OUTPUT:

CNN OUTPUT:



eoc-FINAL.ipynb
File Edit View Insert Runtime Tools Help
filtered_text = filter_text(text) # Apply filtering
print(f"Detected Text: {filtered_text}, Confidence: {confidence:.2f}")
# Load and process the image
img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Convert BGR to RGB
# Display the image
plt.figure(figsize=(6,6)) # Set figure size
plt.imshow(img) # Show image
plt.axis("off") # Hide axes
plt.title("License Plate Image") # Add a title
plt.show()
Detected Text: 64C1, Confidence: 0.11
Detected Text: 05506, Confidence: 0.87
License Plate Image
64-C1
055.06
completed at 5:02 PM
17:02
09-03-2025



colab.research.google.com/drive/1SzNEKEAp1Ha3RSxxwKr19IASq3aVoseX#scrollTo=YEzyVS8doUdZ
# Load and process the image
img = cv2.cvtColor(image, cv2.COLOR_BGR2RGB) # Convert BGR to RGB
# Display the image
plt.figure(figsize=(6,6)) # Set figure size
plt.imshow(img) # Show image
plt.axis("off") # Hide axes
plt.title("License Plate Image") # Add a title
plt.show()
Detected Text: MSuVAMIIAAMA, Confidence: 0.03
Detected Text: MAATMA, Confidence: 0.19
Detected Text: VimamiaaVra, Confidence: 0.07
Detected Text: Mila, Confidence: 0.06
Detected Text: V, Confidence: 0.03
Detected Text: Mu, Confidence: 0.08
Detected Text: GZS0753, Confidence: 0.53
Detected Text: 020, Confidence: 0.90
License Plate Image
0:20
Start coding or generate with AI.

THANK YOU