Licence Plate Detection

YOLO v8, OpenCV, Roboflow, EasyOCR

Index

1. Data Collection

i) Collected real-world vehicle images from **OLX** (multiple Indian states)

ii) Included **white & yellow board vehicles**, focused on **cars, tempos, and lorries**

2. Roboflow Dataset Preparation (Workflow Steps)

Created a Roboflow project & workflow - Pipeline

i) Step 1: Created Object Detection project (YOLOv8-based)

ii) Step 2: Annotated manually – drew bounding boxes on number plates

iii) Step 3: Labeled each box as `license_plate` or custom label

iv) Step 4: Used dynamic crop to isolate regions of interest

v) Step 5: Uploaded manually cropped number plates into a second detection project

vi) Step 6: Used OpenAI / OCR step to extract characters

vii) Step 7: Re-trained on your own YOLOv8 model and used EasyOCR for character recognition

3. Train-Test-Validation Split (70:20:10)

Train = teaching the model
Test = small internal exam
Validation = final exam after training

4. Label Format for YOLOv8

i) Label files (.txt) with normalized bounding boxes (YOLO format)

ii) Structured as: `class x_center y_center width height` — all in relative pixel values

5. YOLOv8 Training

i) Trained with `yolov8n.pt` (nano version)

ii) Used `data.yaml` with train/val/test paths

iii) Tested at **30 epochs**, and learned that **50–100 epochs give better accuracy**

6. Deployment & Prediction Methods

Method 1: Roboflow Inference SDK (cloud)

Method 2: Local YOLOv8 prediction (download weights & run inference)

Integrated with **Streamlit**, **EasyOCR**, and **OpenCV**

This gives both online + offline deployment power.

7. Final Streamlit Apps

i) One App Using -> YOLOv8 License Plate Detection + OCR (Enhanced & Debuggable)

ii) One App Using -> License Plate Detection + OCR (Roboflow + OpenCV)

Classification Report:

Plot Name	What It Shows	Why It Matters
Confusion Matrix	Correct vs incorrect classifications	Detect misclassification
F1 Curve	Balance between precision & recall	Select best threshold
P Curve	Precision vs threshold	Know how "strict" model is
R Curve	Recall vs threshold	Know how many you’re missing
PR Curve	Precision vs Recall	Trade-off view of model accuracy

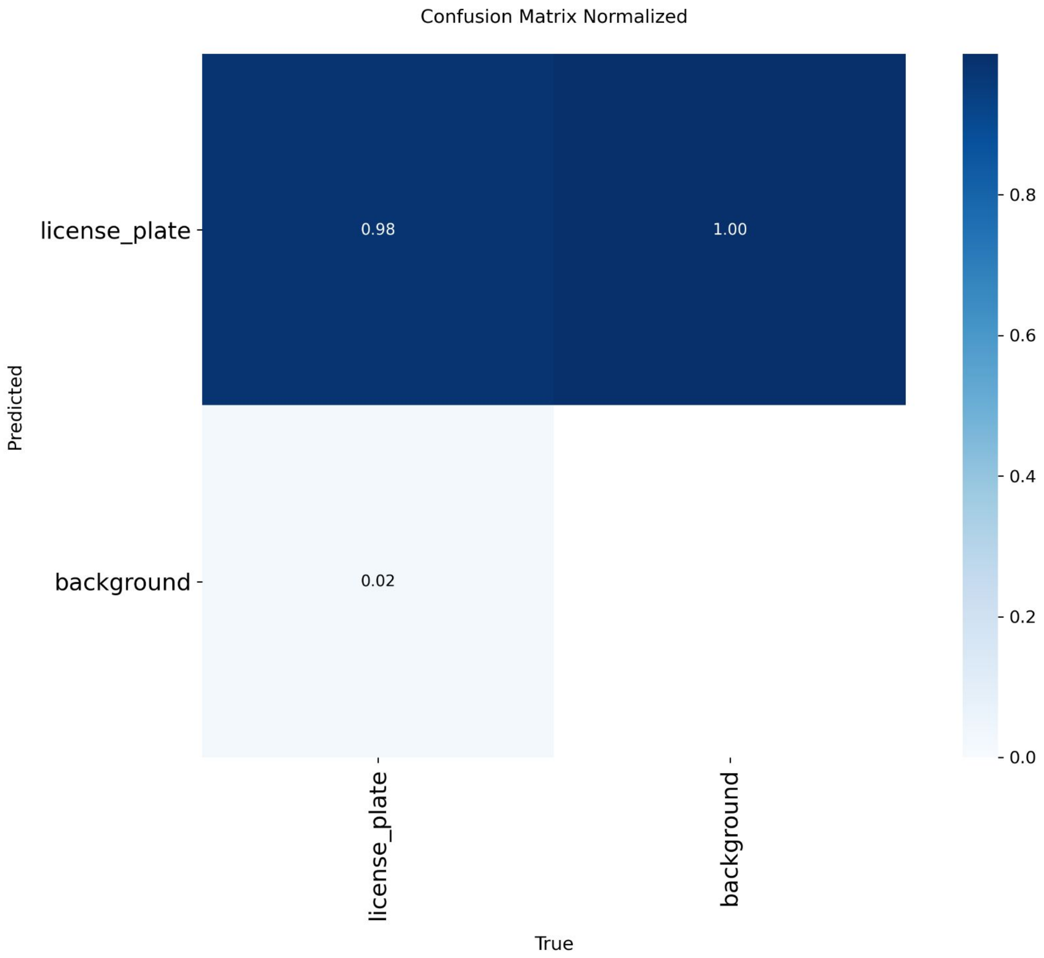
✅ 1. Confusion Matrix (Normalized & Raw)

- **Purpose:** Shows how well the model is classifying each object (in your case, usually just one class like "license_plate").
- 📊 **Confusion Matrix (Raw):**
  - Rows: actual class
  - Columns: predicted class
  - Example:

	Pred: Plate	Pred: Background
True: Plate	✅ TP	❌ FN
True: Background	❌ FP	✅ TN

- 📊 **Normalized Version:** Same, but values are percentages, easier to interpret.

> 📌 **Use case:** Check if your model is confusing license plates with background or missing them (False Negatives).

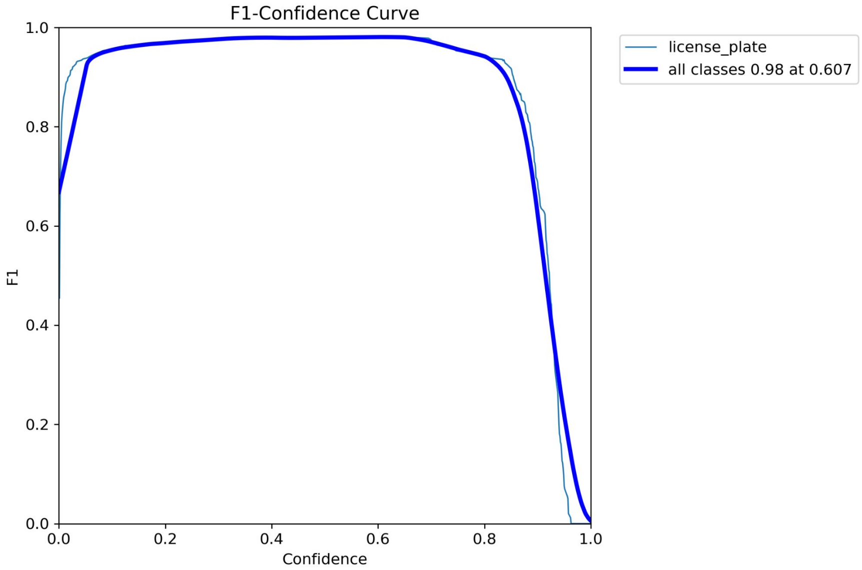


My Machine is little bit confusing because my model have [False Negative] licence plate has background

✅ 2. F1 Curve

- **Purpose:** Shows the **F1-score** (harmonic mean of precision and recall) across different confidence thresholds.
- **Why useful?** It helps you pick a good threshold for deciding whether a detection is "good enough."

> 📌 **Peak point of the curve = optimal balance** between false positives and false negatives.

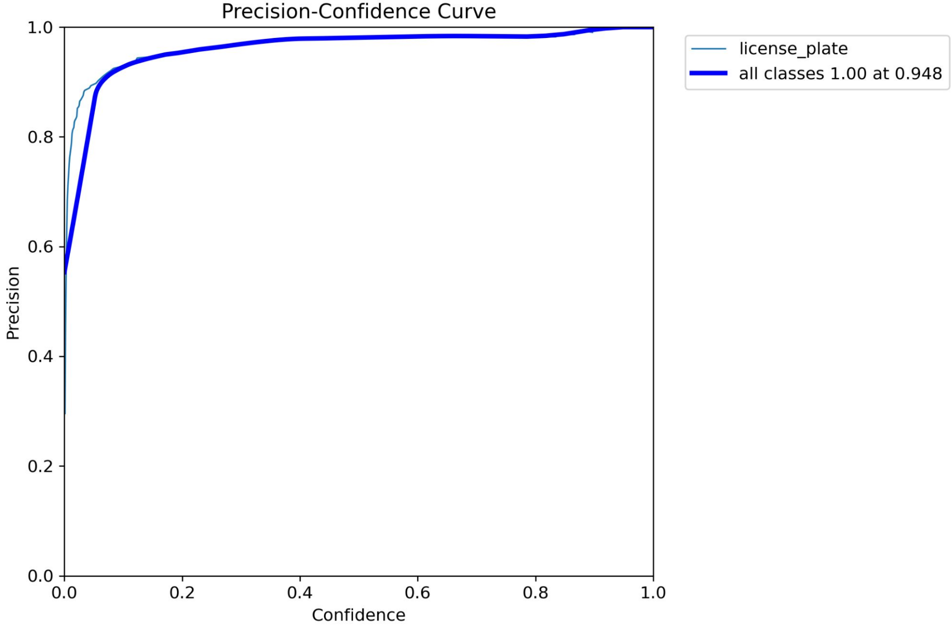


- **F1 Score** combines precision and recall, so 0.98 means your model is both highly accurate (few false positives) and highly complete (few false negatives).
- **Threshold 0.67** means predictions with confidence above 0.67 are considered positive, and at this threshold, your model is performing extremely well.

✅ 3. Precision (P) Curve

- **Purpose:** Shows how **precision** (correct detections out of all predicted) varies with confidence threshold.

> 📌 High precision = fewer false positives.



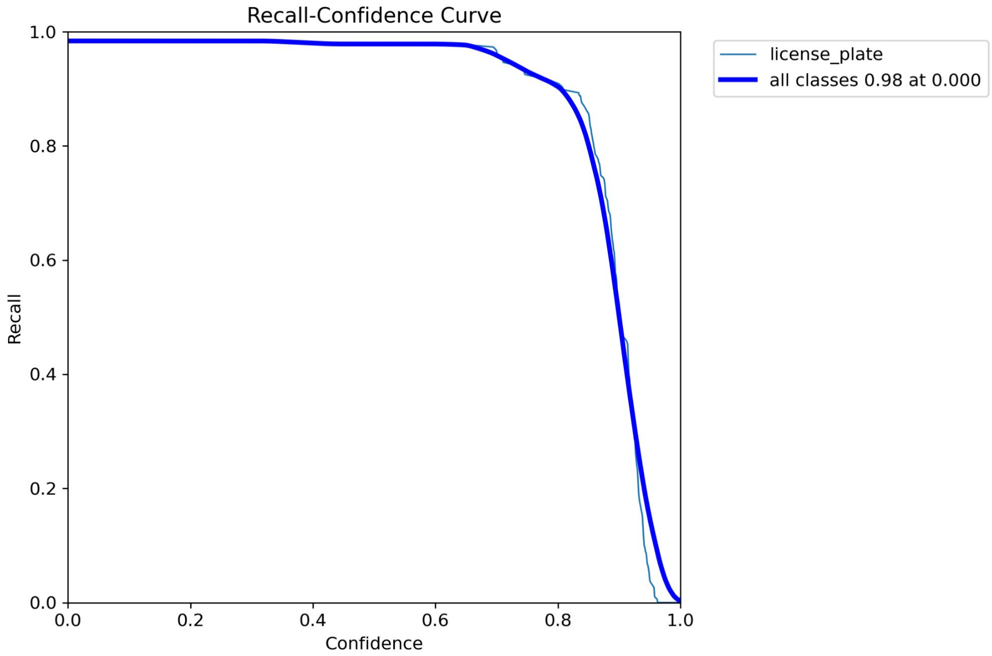
Confidence Curve shows that at a confidence threshold of about 0.95, your model achieves a precision of 1.00 (100%) for all classes, including "license_plate".

- At high confidence, your model makes almost no false positive predictions.
- The curve is very close to the top left, which is ideal.
- This is a strong indicator that your model is highly reliable for detecting license plates, especially when you use a confidence threshold around 0.95.

✅ 4. Recall (R) Curve

- **Purpose:** Shows how **recall** (correct detections out of all actual) varies with threshold.

> 📌 High recall = fewer false negatives (your model doesn't miss many plates).



- **Y-axis (Recall):** The proportion of actual license plates that your model correctly detects (true positives / all actual positives).
- **X-axis (Confidence):** The confidence threshold for your model's predictions (from 0 to 1).

- At low confidence thresholds (left side), recall is very high (close to 1.0), meaning your model detects almost all license plates, even if some predictions are less certain.
- As you increase the confidence threshold (move right), recall stays high until about 0.8–0.9, then drops sharply. This means that at high confidence, the model only predicts the most certain plates, missing some less obvious ones.
- The legend says: "all classes 0.98 at 0.000", meaning at a confidence threshold of 0, recall is 0.98 (98%). This is excellent.

- Your model detects nearly all license plates at low confidence thresholds.
- If you want to maximize recall (find every plate), use a lower threshold.
- If you want to avoid false positives, use a higher threshold, but recall will decrease.

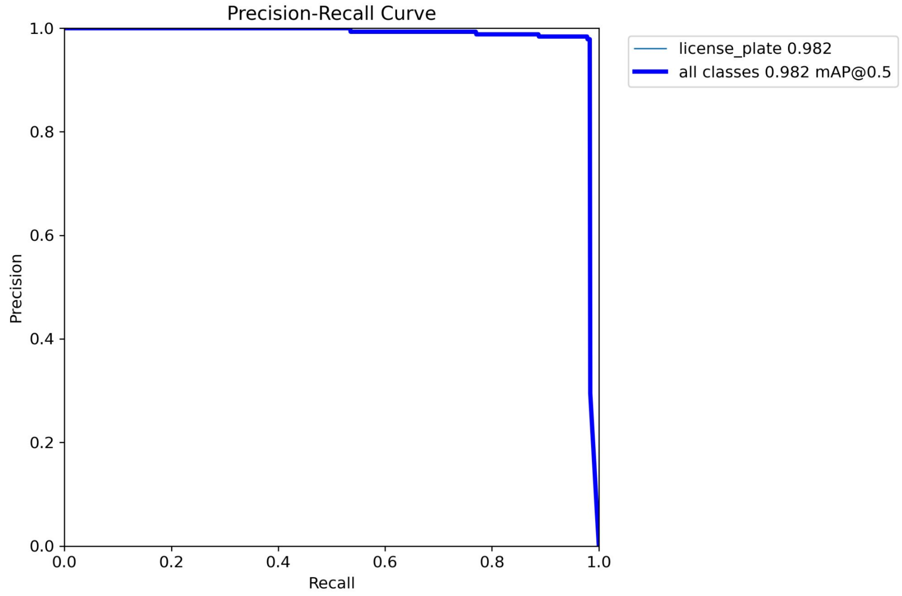
Overall:
This is a very good recall curve, showing your model is highly effective at detecting license plates, especially at reasonable confidence thresholds.

✅ 5. Precision-Recall (PR) Curve

- Combines precision and recall in one plot.
- Good models have curves **closer to the top-right corner.**

> 📌 **Area under this curve** is a good indicator of model performance.

- **X-axis (Recall):** The proportion of actual license plates your model correctly detects (true positives / all actual positives).
- **Y-axis (Precision):** The proportion of detected license plates that are actually correct (true positives / all predicted positives).



- This model achieves very high precision and recall simultaneously.
- An mAP@0.5 of 0.982 means my model is extremely accurate at detecting license plates, with very few false positives or false negatives.

Fine-Tuning Model

One App Using -> **License Plate Detection + OCR (Roboflow + OpenCV)**

Uploaded Image

🔍 Running license plate detection...

0 : {
"output" : "The license plate reads: TN 87 C 5106."
"classes" : NULL

Uploaded Image

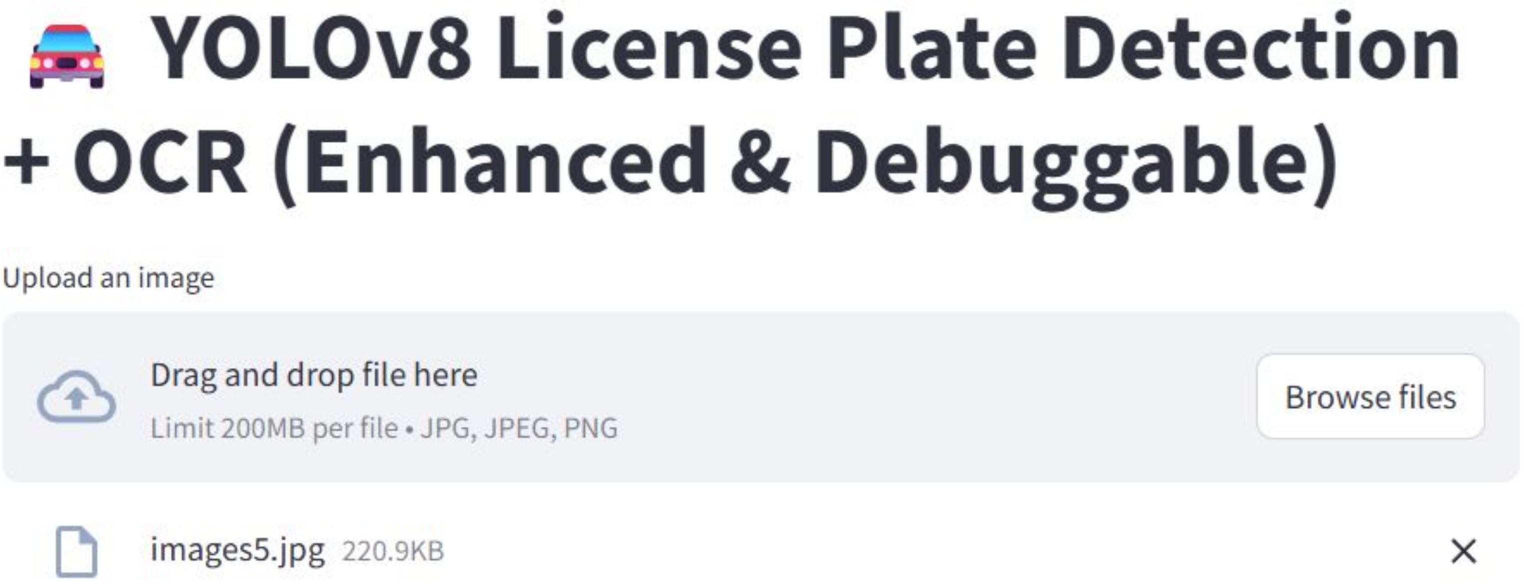
🔍 Running license plate detection...

✅ Detection completed!

0 : {
"output" : "The characters on the license plate are: MH20DY2366"

},
"confidence" : 0.516134858131408
"class_id" : 2
"class" : "car"
"detection_id" : "8dd0bb7a-88c2-453a-b689-a0413d559feb"
"parent_id" : "image"

🚘 YOLOv8 License Plate Detection + OCR (Enhanced & Debuggable) - Fine-tuning




license_plate 0.91
MH 20 EE 7602
Team-BHP.com

Plate 1

Cropped License Plate

Preprocessed (Thresh + CLAHE)

Detected Text: `MH20EE7602`

Download License Plate Text (.txt)

Detected License Plate(s)

🔍 Extracted License Plate Texts:

Plate 1

📷 Cropped License Plate

🧪 Preprocessed (Thresh + CLAHE)

Detected Text: `MP33C3370`

📥 Download License Plate Text (.txt)

Feature	App 1: YOLOv8 + OCR + CLAHE	App 2: Roboflow + OCR
Detection engine	Your own YOLOv8 model	Roboflow cloud model
Requires GPU	Optional (for local speed)	❌ No GPU needed
OCR method	EasyOCR	EasyOCR
Image preprocessing	CLAHE + Thresholding	Basic crop
Output quality (OCR accuracy)	High with good preprocessing	Depends on Roboflow box
Offline compatibility	✅	❌ Needs Internet
Training flexibility	Full control	Based on Roboflow

✅ The method involving fine-tuning is:

> ✔️ App 1: YOLOv8 Custom Model + OCR + CLAHE Preprocessing

- **Collected real-world images** from OLX (cars, tempos, lorries, white/yellow boards).
- **Manually annotated** the license plates (bounding boxes).
- **Trained a YOLOv8 model** (e.g., on 1375 images) using your dataset.
- This training involved **fine-tuning YOLOv8 weights** (like `yolov8n.pt`) on **your custom data**.
- You adjusted **epochs**, experimented with **training/validation/test split**, and saved the final weights (`best.pt`).

We fine-tuned a YOLOv8 object detection model on a custom OLX vehicle dataset with annotated license plates, achieving accurate localization and text recognition using EasyOCR. For comparison, we also implemented an inference-only version using Roboflow's hosted detection pipeline

App	Fine-Tuning?	Why
App 1: YOLOv8 + OCR	✅ Yes	You trained (fine-tuned) YOLOv8 on your annotated OLX dataset
App 2: Roboflow + OCR	❌ No	Only using Roboflow's inference API, no training involved

Thank You