Automatic License Plate Detection and Recognition

Bipika Sapkota and Vyshali Poola

Oakland University

MIS-5600

Deep Learning and Text Analytics

Dec 11th, 2025

Contents

1. Introduction

Automatic License Plate Recognition (ALPR) has become an essential component of modern intelligent transportation systems, supporting applications such as traffic law enforcement, automated tolling, smart parking systems, border security, and autonomous vehicle perception. An effective ALPR system must perform two major tasks, accurately localize the license plate within an image and reliably extract the alphanumeric characters present on the plate. However, this process is challenging due to variations in lighting, viewing angles, motion blur, occlusions, plate designs, and image quality across real world environments.

In this project, we worked on two state of the art deep learning approaches for the license plate detection component of the ALPR pipeline:

1. **Faster R-CNN**, a two-stage region proposal-based detector, known for high detection accuracy.
2. **YOLOv8**, a one stage detector designed for fast, real-time inference while maintaining strong accuracy.

Faster R-CNN is a two-step detection model that is known for being very accurate, especially when plates are small, unclear, or partly hidden. We trained and customized this model so it could reliably locate license plates in challenging images. YOLOv8, on the other hand, is a faster, one step detection model designed for real time use. We trained YOLOv8 on the same dataset to compare its speed and accuracy with Faster R-CNN and to see how well it performs in practical scenarios.

2. Problem Statement

ALPR is widely used across transportation, law enforcement, parking automation, and toll systems. However, the accuracy of these systems heavily depends on two critical components:

1. **License plate detection** - identifying the exact plate region within an image.
2. **License plate text recognition** - extracting the alphanumeric characters accurately

To address these challenges, this project evaluates two modern deep learning-based detection models.

2.1 Questions to be answered

1. Which detection model YOLOv8 or Faster R-CNN provides higher accuracy and robustness for license plate localization?
2. Which OCR method, EasyOCR or PaddleOCR produces better text extraction accuracy from detected plates?
3. How do preprocessing techniques impact OCR performance in end-to-end ALPR pipelines?
4. How do the models compare in terms of accuracy (mAP, precision, recall, F1 score) and processing speed?

2.2 Business Case/need

Modern transportation systems as transportation systems become more automated, reliable ALPR technology plays a key role in improving safety, efficiency, and everyday operations. It helps agencies and businesses monitor vehicles automatically, reduce manual work, and prevent errors in tasks like parking management, traffic enforcement, and access control. As a result, companies must consider the following essential requirements when selecting or deploying an ALPR system.

1. **Automating License Plate Reading**
   Companies and government agencies need automatic systems to detect and read license plates without human intervention to reduce manual effort and errors.

2. **Improving Accuracy in Real World Conditions**
   Vehicles appear at different angles, speeds, lighting, and resolutions. Businesses need a model that can reliably detect plates even under difficult conditions.

3. **Supporting Public Safety & Law Enforcement**
   Accurate plate detection helps police track stolen vehicles, monitor traffic violations, and enhance road safety.

4. **Speeding Up Transportation Operations**
   ALPR systems are used in toll booths, parking lots, and traffic monitoring. Faster and more accurate models (like YOLO or Faster RCNN) improve efficiency and reduce delays.

5. **Choosing the Best Model for Deployment**
   Organizations must decide whether a fast model (YOLO) or a more detailed model (Faster RCNN) is better for their system. This project helps compare both to guide real world adoption.

3. Dataset

The project uses a vehicle license plate dataset obtained from **Roboflow.com**, containing images of cars.

- **Total images:** 8,152
- **Training set:** 7,337 images
- **Validation set:** 815 images
- **Split ratio: 80:20** (train : validation)

The dataset includes various lighting conditions, angles, backgrounds, and plate styles, making it suitable for evaluating object detection and OCR performance.

3.1 Identifying and acquiring relevant data

It reflects real world vehicle images, supporting model testing under diverse conditions.

3.2 Data Cleaning and Transformation

1. **Standardizing image size** - All images were automatically resized to 640×640 during training.

2. **Format conversion** - Roboflow exported the data in YOLO and COCO Json format, with separate image and label folders.

3. **Normalization & augmentation** - Both models performed internal transformations such as scaling and normalization.

4. Exploratory Data Analysis

EDA helped us examine the dataset by reviewing sample images, verifying annotation accuracy, and understanding patterns such as plate size, position, and image clarity. This step ensured the dataset was clean and suitable for training reliable deep learning models.

4.1 Ensuring data quality and Visualization

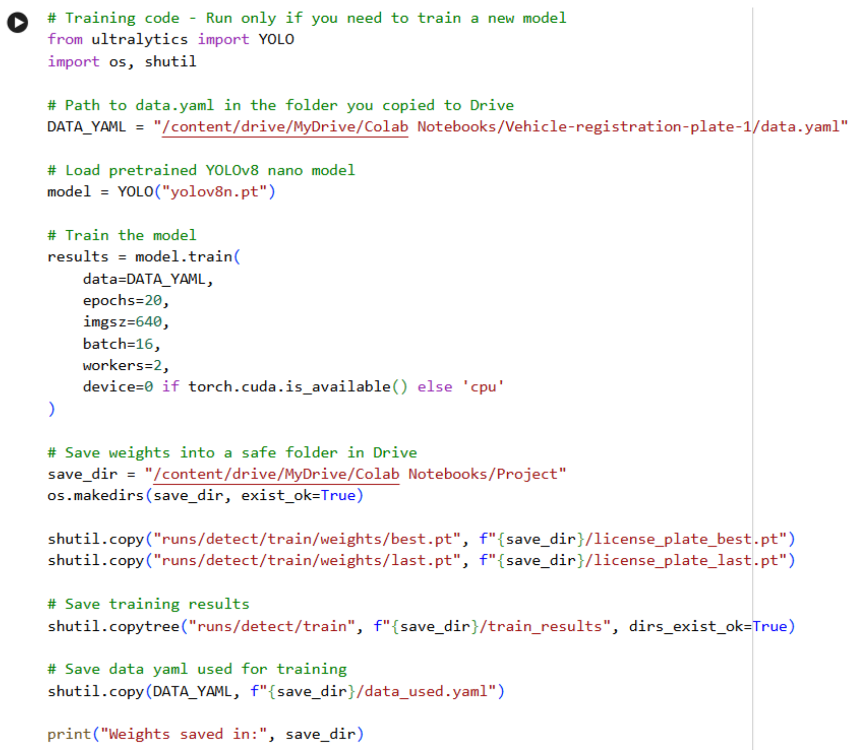
As part of the EDA process, these sample images highlight the diversity within the dataset. Some images contain vehicles and license plates clearly visible in normal traffic scenes, while others include challenging conditions such as long-distance shots, unusual angles, or cluttered backgrounds. This variability helps ensure that the detection models learn to generalize across different environments. The examples also help verify that the dataset includes both easy and difficult cases for plate detection. By visually inspecting such samples, we confirm that the data quality and variety are suitable for training robust detection and OCR models. This step is essential because understanding these variations early helps guide preprocessing decisions and ensures that the detection models are trained on realistic scenarios.

Low visibility cliff scene
Challenging lighting conditions

Low contrast, Poor visibility
Low resolution, small targets

5. Model Building

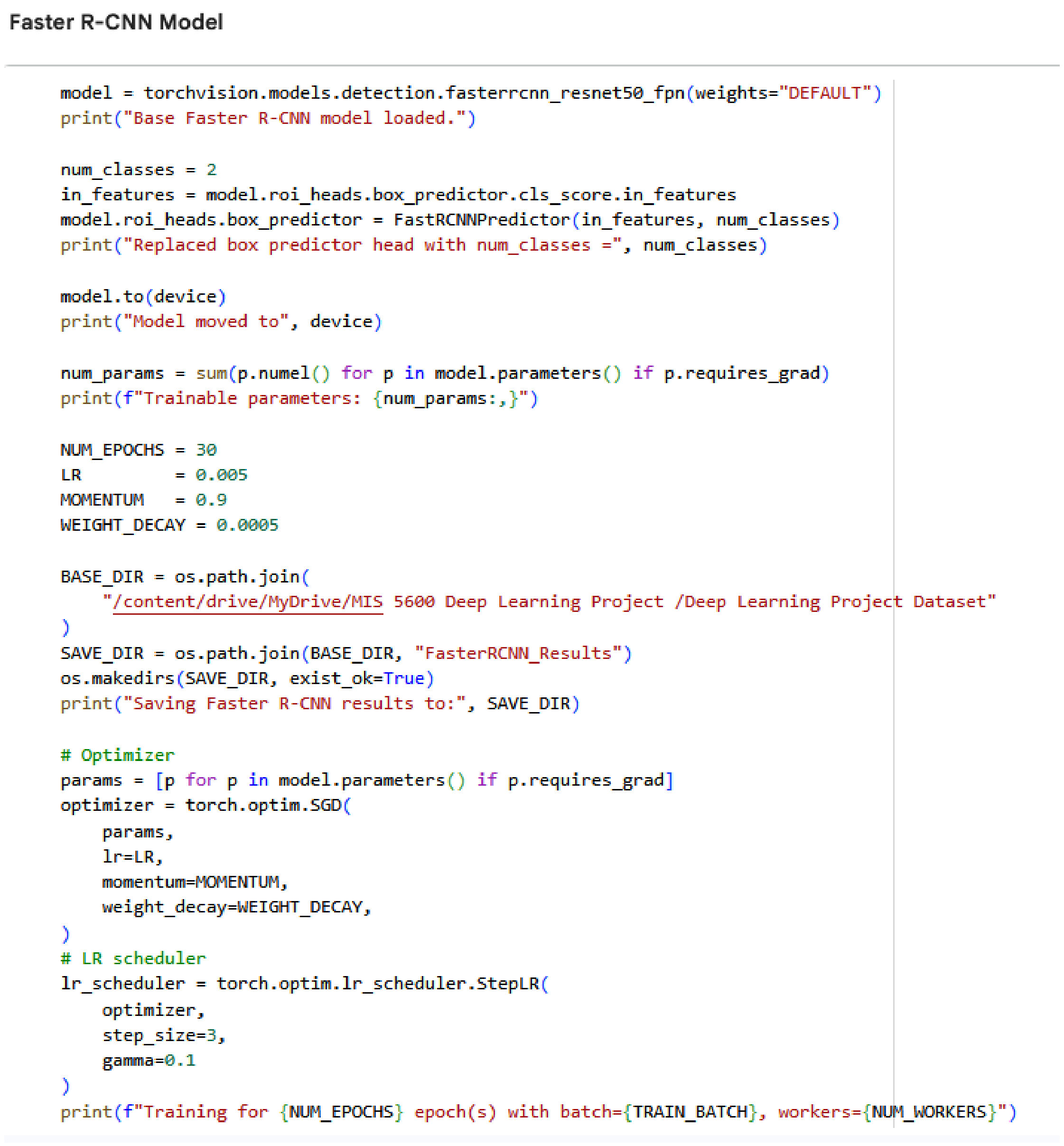
5.1 YOLO Model



```python
# Training code - Run only if you need to train a new model
from ultralytics import YOLO
import os, shutil

# Path to data.yaml in the folder you copied to Drive
DATA_YAML = "/content/drive/MyDrive/Colab Notebooks/Vehicle-registration-plate-1/data.yaml"

# Load pretrained YOLOv8 nano model
model = YOLO("yolov8n.pt")

# Train the model
results = model.train(
    data=DATA_YAML,
    epochs=20,
    imgsz=640,
    batch=16,
    workers=2,
    device=0 if torch.cuda.is_available() else 'cpu'
)

# Save weights into a safe folder in Drive
save_dir = "/content/drive/MyDrive/Colab Notebooks/Project"
os.makedirs(save_dir, exist_ok=True)

shutil.copy("runs/detect/train/weights/best.pt", f"{save_dir}/license_plate_best.pt")
shutil.copy("runs/detect/train/weights/last.pt", f"{save_dir}/license_plate_last.pt")

# Save training results
shutil.copytree("runs/detect/train", f"{save_dir}/train_results", dirs_exist_ok=True)

# Save data yaml used for training
shutil.copy(DATA_YAML, f"{save_dir}/data_used.yaml")

print("Weights saved in:", save_dir)
```

The YOLOv8 model was used as the base architecture for license plate detection. The model was trained on the Roboflow dataset using a 20 epoch training process with 640×640 image resolution and a batch size of 16. During training, YOLO automatically optimized bounding box predictions, classification confidence, and localization accuracy. After training completed, the best performing weights and training logs were saved to Google Drive for further evaluation and comparison with Faster R-CNN. This trained YOLO model serves as the core detector that later feeds cropped plate regions into the OCR pipeline for text recognition.

5.2 Faster R-CNN Model



Faster R-CNN Model

```
model = torchvision.models.detection.fasterrcnn_resnet50_fpn(weights="DEFAULT")
print("Base Faster R-CNN model loaded.")

num_classes = 2
in_features = model.roi_heads.box_predictor.cls_score.in_features
model.roi_heads.box_predictor = FastRCNNPredictor(in_features, num_classes)
print("Replaced box predictor head with num_classes =", num_classes)

model.to(device)
print("Model moved to", device)

num_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print(f"Trainable parameters: {num_params:,}")

NUM_EPOCHS = 30
LR          = 0.005
MOMENTUM    = 0.9
WEIGHT_DECAY = 0.0005

BASE_DIR = os.path.join(
    "/content/drive/MyDrive/MIS 5600 Deep Learning Project /Deep Learning Project Dataset"
)
SAVE_DIR = os.path.join(BASE_DIR, "FasterRCNN_Results")
os.makedirs(SAVE_DIR, exist_ok=True)
print("Saving Faster R-CNN results to:", SAVE_DIR)

# Optimizer
params = [p for p in model.parameters() if p.requires_grad]
optimizer = torch.optim.SGD(
    params,
    lr=LR,
    momentum=MOMENTUM,
    weight_decay=WEIGHT_DECAY,
)
# LR scheduler
lr_scheduler = torch.optim.lr_scheduler.StepLR(
    optimizer,
    step_size=3,
    gamma=0.1
)
print(f"Training for {NUM_EPOCHS} epoch(s) with batch={TRAIN_BATCH}, workers={NUM_WORKERS}")
```

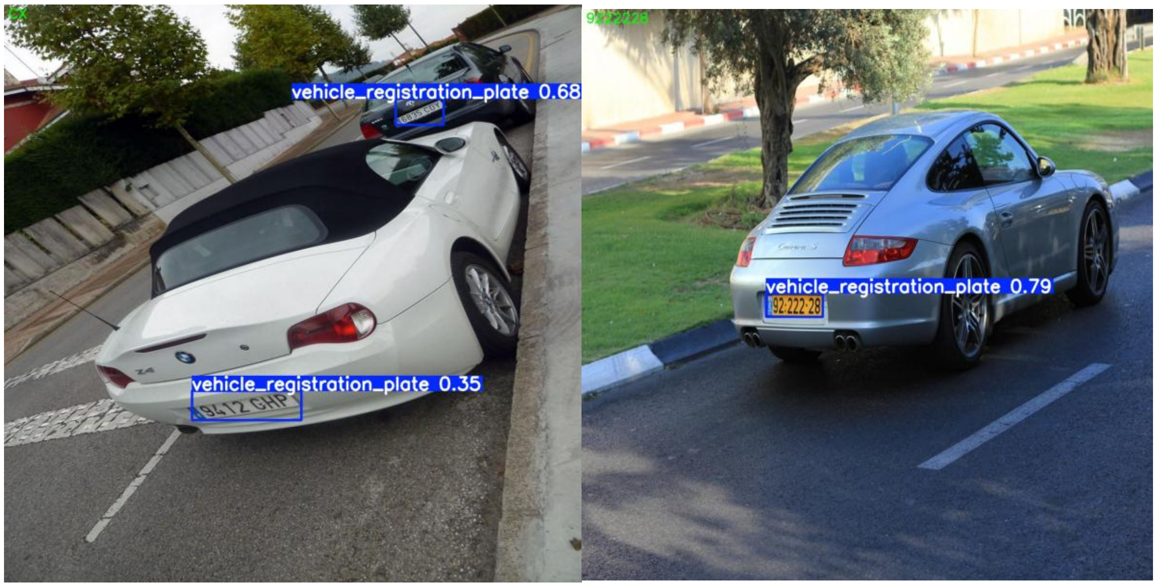
To build the Faster R-CNN detection model, we first loaded a pretrained COCO model and customized it by replacing its default classification head with a new predictor configured for two classes: background and license plate. The modified model was then moved to the GPU, and we calculated the number of trainable parameters to ensure that only the necessary layers were fine-tuned. For training, we set the hyperparameters to 30 epochs, a learning rate of 0.005, momentum of 0.9, and weight decay of 0.0005, and used an SGD optimizer to update the model weights. A StepLR scheduler was also added to reduce the learning rate every three epochs and stabilize training. Finally, we created a directory to save all model outputs, logs, and trained weights for later evaluation and comparison with the YOLOv8 model**.**

6. Model Application

After training both detection models, the next step was to apply them to new, unseen images and evaluate how well they performed in a full ALPR pipeline. Each model was used to automatically detect the location of the license plate within an input image, producing bounding boxes and confidence scores. The detected plate regions were then cropped and passed to the OCR component EasyOCR or PaddleOCR to extract the alphanumeric characters. This end-to-end application allowed us to test how accurately the models performed on real images, compare their speed and reliability, and observe how detection quality affects the success of OCR. By running the complete pipeline, we were able to determine which model combination provides the most effective and robust ALPR solution under realistic conditions.

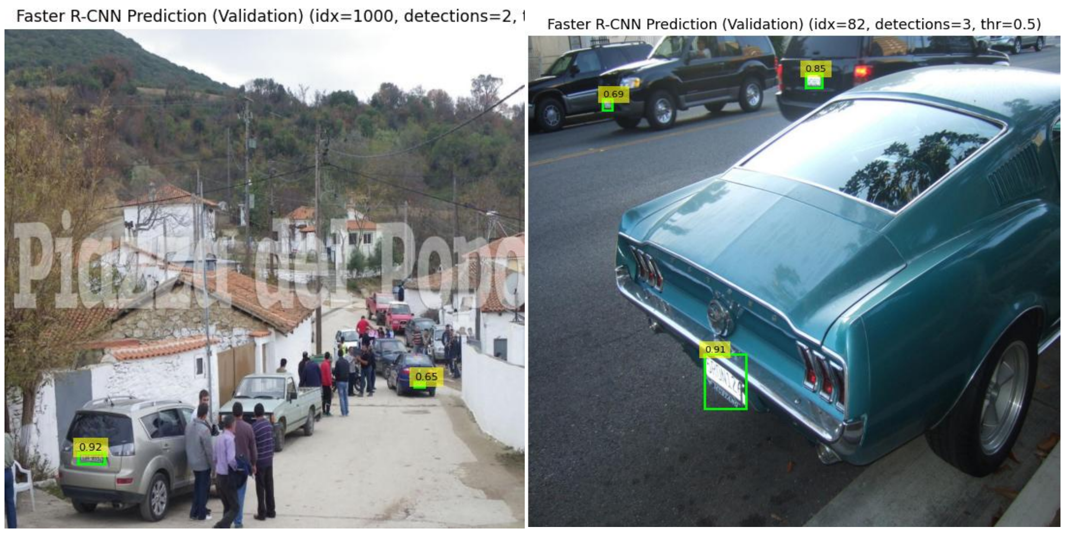
6.1 Generating insights

The YOLOv8 model successfully detected the vehicle registration plates in various road scenarios, as shown in the examples below. Each detected plate is highlighted with a bounding box and a corresponding confidence score, which represents how certain the model is that the identified region truly contains a number plate.


vehicle_registration_plate 0.68
vehicle_registration_plate 0.35
9412 GHP
vehicle_registration_plate 0.79
92-222-28

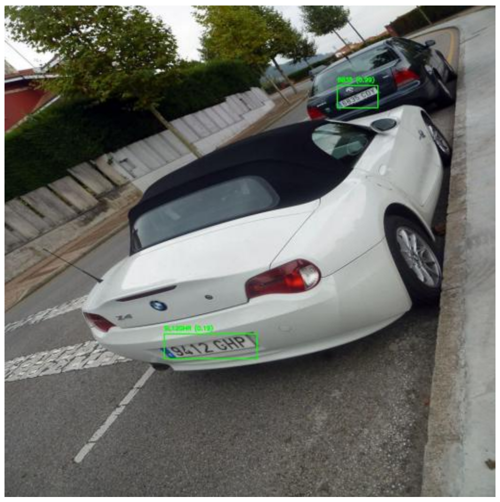
For instance, in the first image, the model detected the plate with a confidence of **0.68**, while in the second example, the detection confidence increased to **0.79**, indicating a stronger level of certainty. These confidence scores provide immediate insight into model reliability across different lighting conditions, camera angles, and vehicle types.

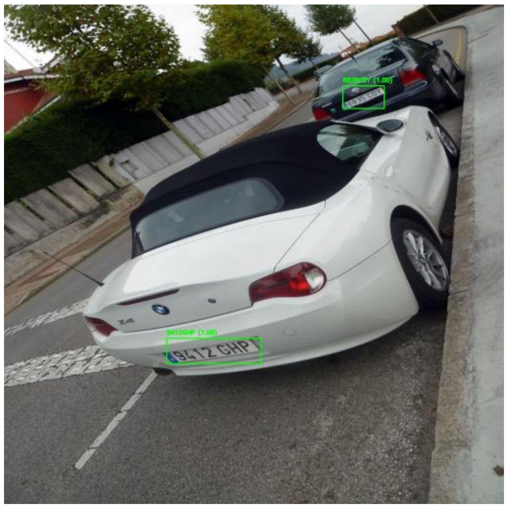
Likewise, the Faster R-CNN model was able to detect vehicle registration plates across a wide range of scenarios, including crowded street scenes and angled close-up views. The model successfully produced accurate bounding boxes with confidence scores, even when plates were small or partially obscured. These results highlight Faster R-CNN's strength in handling complex environments and its ability to maintain reliable detection performance under challenging visual conditions.


Faster R-CNN Prediction (Validation) (idx=1000, detections=2, t
0.65
0.92
Faster R-CNN Prediction (Validation) (idx=82, detections=3, thr=0.5)
0.85
0.69
0.91

6.2 Applying the model to solve the problem

Once the YOLO model successfully identified and localized the license plate, the next stage involved extracting the text using Optical Character Recognition (OCR). In this project, we evaluated two OCR engines, EasyOCR and PaddleOCR to understand their performance differences on the same detected plate region. The left figure shows with EasyOCR and the right figure shows with PaddleOCR.





EasyOCR Results

EasyOCR provides quick and lightweight text extraction. However, in several cases including the example shown it correctly recognized only part of the plate. For instance, EasyOCR extracted:

- **EasyOCR output:** 6835
- **Issue:** It captured only the first four characters instead of the full plate "6835 CDY".

This happens because EasyOCR is sensitive to tilted or skewed plates, low contrast between characters and plate, partial visibility and lack of strong preprocessing.

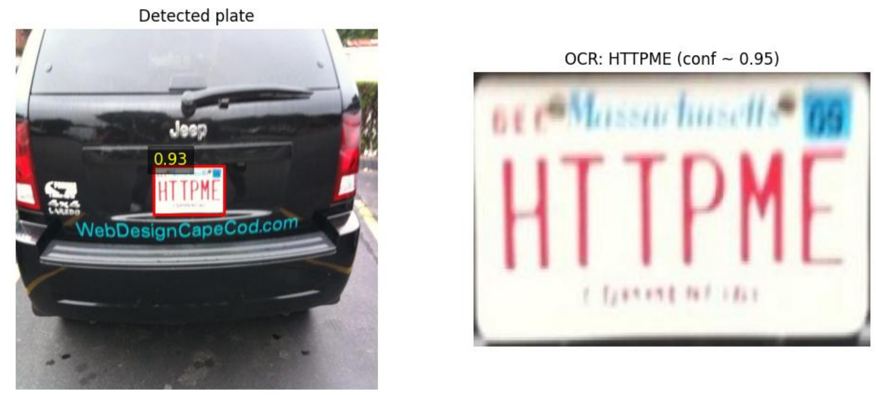
PaddleOCR Results

PaddleOCR produced significantly better results for the same input. After applying enhanced preprocessing (deskewing, grayscale conversion, upscaling), PaddleOCR recognized the full plate correctly:

- **PaddleOCR output:** 6835 CDY
- **Observation:** PaddleOCR is more robust for multi character lines and supports stronger text line detection.

PaddleOCR uses a more advanced detection network for identifying complete text lines by better handling of rotation, blur, and low resolution characters and improved character segmentation.

Another example of a Faster RCNN EasyOCR Detection is shown below.



```
Predicted plate text: 'HTTPME' (det score=0.930, OCR conf≈0.948)
```

The detected plate is accurately cropped and passed to the OCR model. EasyOCR successfully reads the characters "HTTPME" with a high confidence score of 0.95, demonstrating strong performance even with slight blur and varying color patterns on the plate. This example highlights how reliable OCR extraction becomes when the detection model provides a clear, well-framed plate region.

6.3 Decision making and value proposition

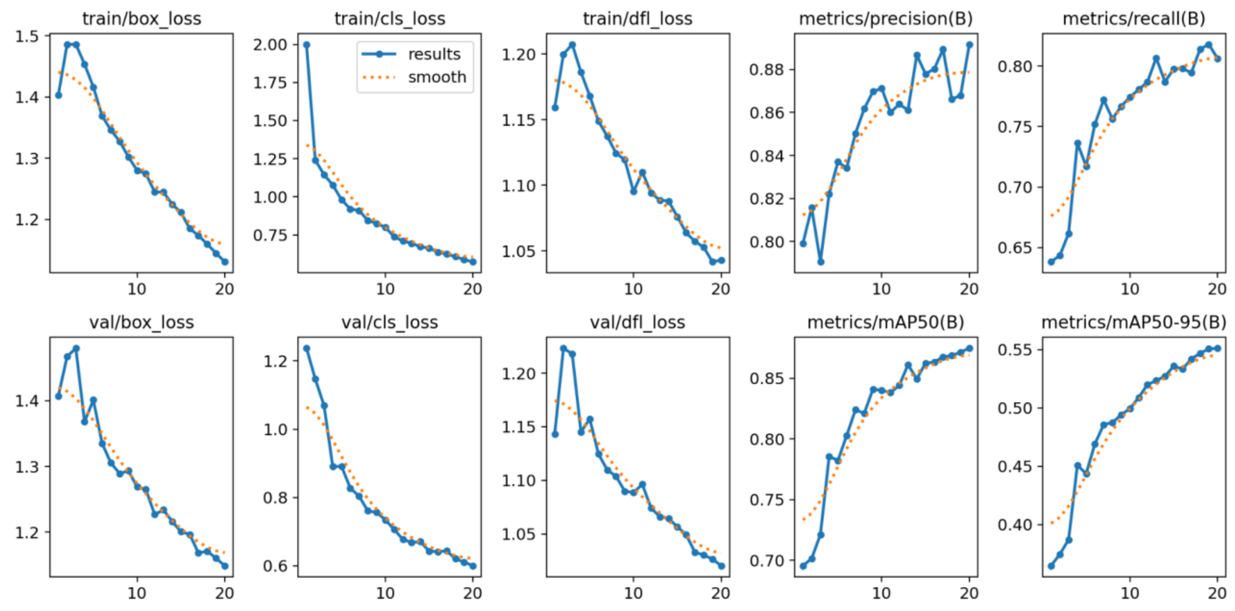
This system provides meaningful decision support by automatically converting vehicle images into structured, actionable data. Organizations no longer need to rely on manual plate transcription, which is slow, costly, and error prone. YOLO enables high confidence plate detection, while PaddleOCR ensures accurate text extraction even in challenging visual conditions. Together, these capabilities support faster processing in parking enforcement, automated gate entry, and traffic analytics. The value proposition lies in improved operational efficiency, reduced human workload, and the ability to scale license plate recognition reliably across thousands of images.

7. Model Dissemination

The final ALPR models and results were organized, documented, and shared in a way that supports reuse, transparency, and further development. All trained weights, evaluation metrics, sample predictions, and code files were stored in Google Drive for easy access. Visual examples of detection and OCR outputs are also included to help stakeholders understand system performance. This dissemination process allows easy access and clear understanding of the model outputs.

7.1 Monitoring model performance

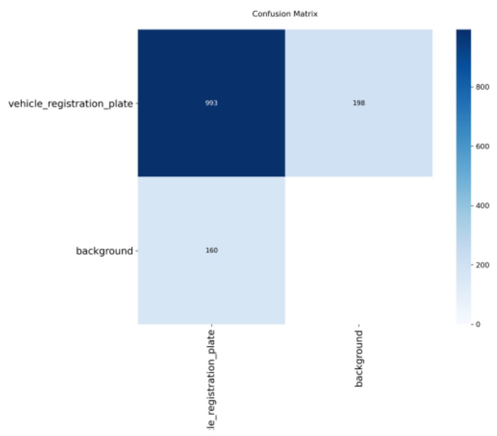
YOLO Model

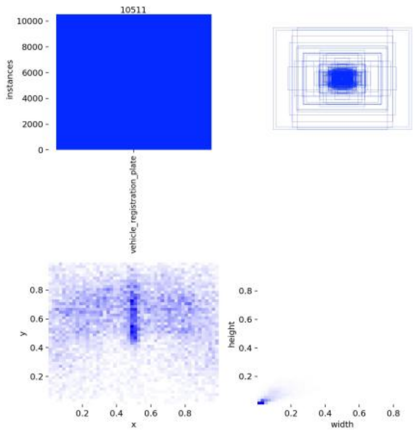


YOLO model shows excellent learning behavior across the 20 training epochs:

- All training and validation losses decrease smoothly → stable and healthy learning.
- Precision (0.89) and recall (0.82) indicate strong detection reliability.

- mAP@50 of 0.87 shows the model accurately detects license plates in most images.
- mAP@50-95 of 0.55 demonstrates good localization performance even under strict IoU criteria.



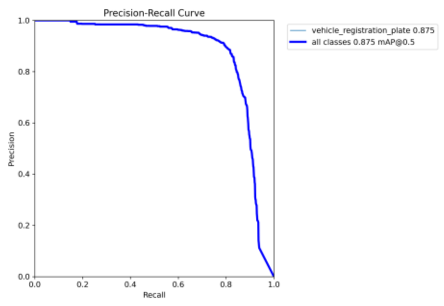


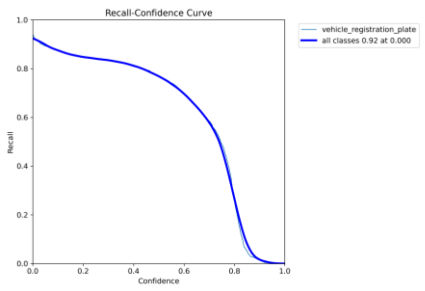
Confusion matrix

The confusion matrix shows that the YOLO model correctly detects most license plates: **993** predictions are true positives.There are **198** false positives where the model predicted a plate on background regions, and **160** false negatives where a real plate was missed and treated as background. Overall, this pattern is consistent with the precision/recall values from training the detector is generally reliable but still occasionally fires on background clutter and can miss plates in harder conditions.

Label distribution plot.

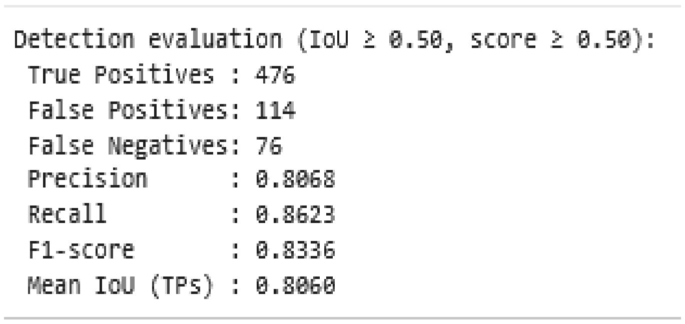
The label heatmaps confirm that the dataset is highly focused on a single class with around 10,500 annotated plates. Most plates are relatively small in the image and tend to appear in the lower middle region of the frame, which makes sense for rear view traffic photos where plates sit near the bottom of the vehicle. This concentrated spatial and size distribution explains why YOLO learns to detect plates well in typical driving scenes, but may struggle more with very close up or unusual camera angles that are under-represented in the training data.





- The PR curve stays high across most recall values, indicating that the model consistently makes correct detections with very few false positives.
- Recall gradually decreases as confidence threshold increases, which is expected higher confidence thresholds filter out more predictions, including some true positives.
- The model achieves a maximum recall of around 0.92 at low confidence, showing that the model can capture nearly all true plates when using a permissive confidence threshold.

Faster RCNN Model



```
Detection evaluation (IoU ≥ 0.50, score ≥ 0.50):
 True Positives : 476
 False Positives: 114
 False Negatives: 76
 Precision      : 0.8068
 Recall         : 0.8623
 F1-score       : 0.8336
 Mean IoU (TPs) : 0.8060
```

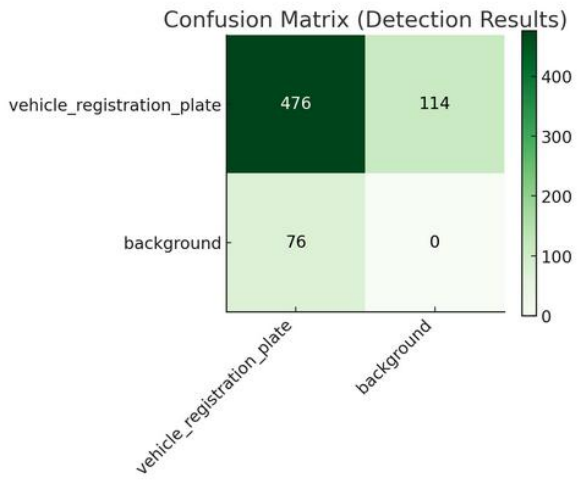
The model shows strong overall detection performance, achieving a precision of 0.8068, meaning most of the plates it predicted as positive were correct.

With a recall of 0.8623, the model successfully detected the majority of actual license plates, missing relatively few.

The F1-score of 0.8336 indicates a good balance between precision and recall, showing that the model performs reliably across different detection scenarios.

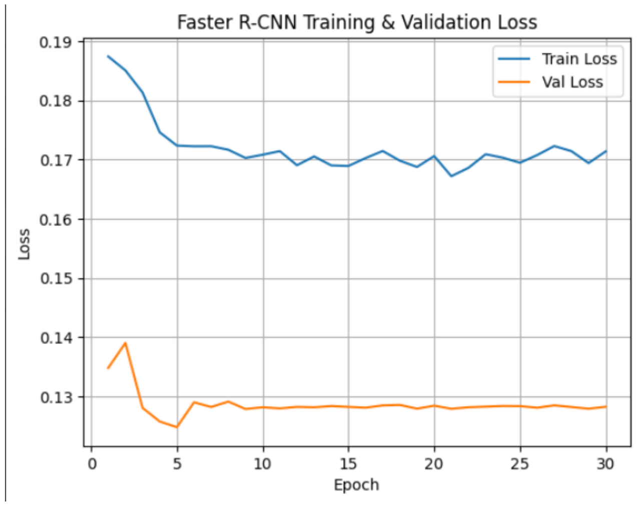
The mean IoU of 0.8060 suggests that the predicted bounding boxes closely match the ground-truth locations, demonstrating accurate localization.

Confusion Matrix



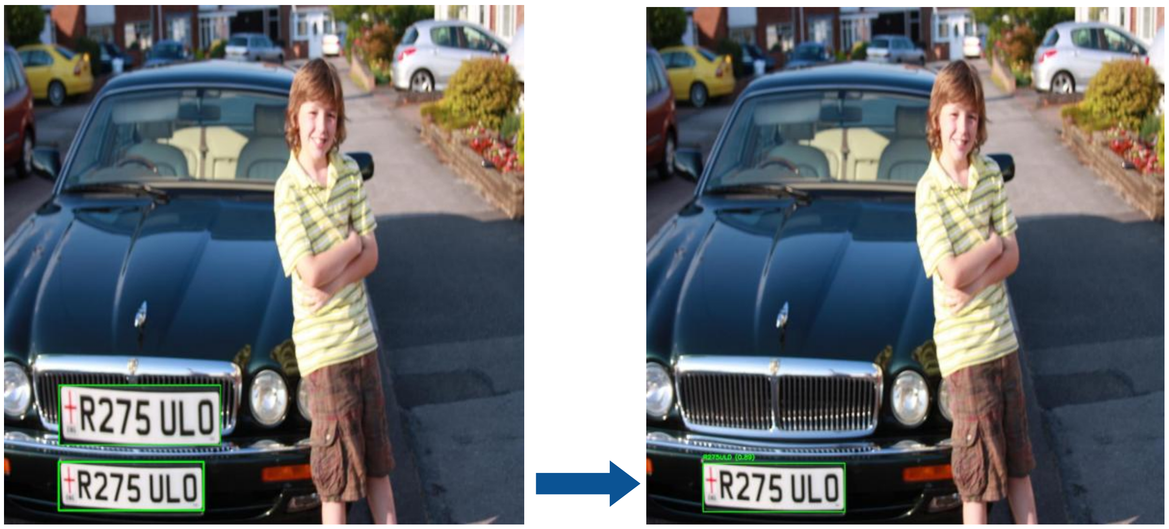
The confusion matrix for the Faster R-CNN model shows that it correctly detected 476 license plates, demonstrating strong localization performance. However, the model also produced 114 false positives, where background regions were mistakenly identified as plates, and missed 76 actual plates (false negatives). While the model achieves high true detections, these errors highlight opportunities to further refine thresholding or training parameters to improve overall precision and recall.

Faster R-CNN training & validation loss graph



Faster R-CNN training and validation loss curves show a steady decrease during the early epochs, indicating that the model was learning effectively. The training loss continues to drop and then levels off around epoch 10, while the validation loss remains lower and stable after an initial decline. This pattern suggests that the model is not overfitting, as both losses move smoothly without separating from each other. Since the model was trained for 30 epochs, these trends demonstrate consistent learning and stable performance throughout the entire training process.

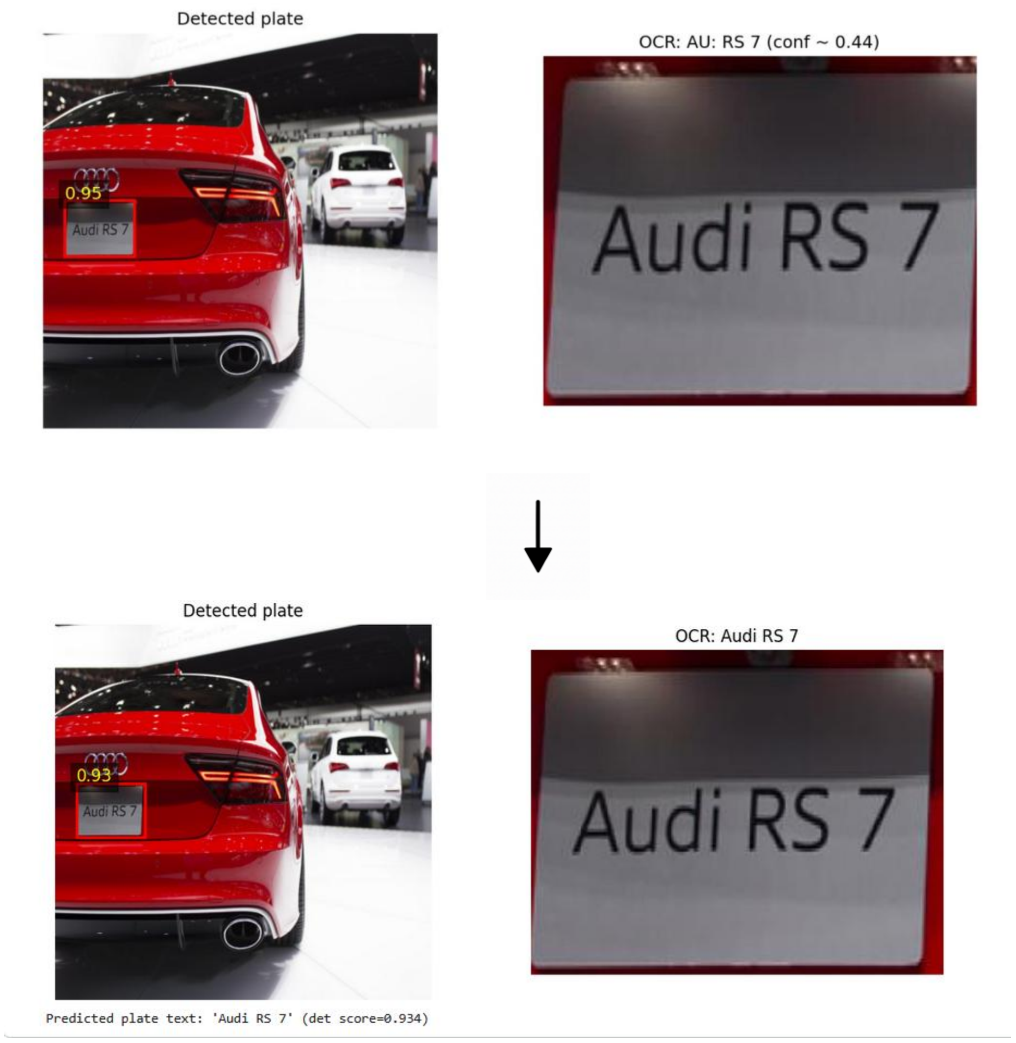
7.2 Model improvement



During testing, a few images showed incorrect or low confidence plate recognition, mainly due to blurs, lighting variations, and tilted plates. To address these challenges, we added several helper functions based preprocessing steps that significantly improved OCR accuracy.

The images above illustrate this improvement. In the first image, the model detects the plate but the OCR prediction is inaccurate because the text is slightly blurred and slanted. After applying the preprocessing pipeline grayscale conversion, deskewing, contrast enhancement, and super-resolution the OCR model receives a much cleaner plate region. As a result, the second image shows a correct and high confidence OCR prediction of the same license plate.

These preprocessing enhancements help the model to reduce noise and blur before OCR, Straighten tilted plates, improve character clarity with upscaling and strengthen edges and contrast for better text recognition. Overall, these improvements increase the reliability of the recognition pipeline and allow the system to handle real world images more effectively.



```
Predicted plate text: 'Audi RS 7' (det score=0.934)
```

8. Conclusion and Challenges

Conclusion

In this project, we successfully developed and evaluated an ALPR system using two modern deep learning detectors YOLOv8 and Faster R-CNN along with OCR methods for text extraction.

Based on the Model results, YOLOv8 demonstrated stronger overall detection performance compared to Faster R-CNN in this ALPR system. YOLO achieved higher precision (0.892), recall (0.804), and F1 score (0.846), along with a superior mAP@50 of 0.872, indicating more accurate and consistent plate localization across the dataset. In contrast, Faster R-CNN achieved competitive but lower precision (0.8068) and F1 score (0.8336), with a higher recall of 0.8623 but more false positives and false negatives. These results show that YOLOv8 provides a better balance of accuracy, robustness, and efficiency for real-time ALPR applications. Overall, the findings highlight YOLOv8 as the more effective model for license plate detection in this project, while Faster R-CNN remains a strong alternative for scenarios where recall is prioritized.

Overall, both models demonstrated strong performance in detecting license plates across diverse real-world conditions, and the OCR pipeline showed clear improvements after applying preprocessing techniques such as grayscale conversion, deskewing, and super-resolution. These results highlight the effectiveness of deep learning-based approaches for building reliable ALPR systems.

Challenges

During the development of the ALPR system, the primary challenge was the long training time required for the model, as each epoch consumed a lot of computing power. Additionally, variations in lighting, blur, and plate orientation sometimes affected OCR accuracy, requiring extra preprocessing steps. Despite these challenges, the project provided valuable hands-on experience in model training, evaluation, and real-world computer vision problem solving. Even with this constraint, the project demonstrated that deep learning models can reliably detect license plates and contribute effectively to automated recognition tasks.

9. Contribution of each member and lessons learnt

Vyshali Poola
I worked on the YOLO model for license plate detection and performed the OCR pipeline, including preprocessing steps and accuracy improvements. I also validated the model, analyzed performance metrics, and prepared the results for the report.

Bipika Sapkota
I worked on the Faster R-CNN model, including training, tuning, and evaluating its performance by doing EASY OCR pipeline. I analyzed performance metrics and prepared the results for the report.

We compared both the models and contributed the findings and visual outputs to the report.

10. Acknowledgement

We would like to express my sincere gratitude to Professor Dr. Vijayan Sugumaran for the guidance and support provided throughout this project. Your introduction to advanced deep learning models along with your insights on applying these techniques to real world scenarios, has been invaluable. Thank you for encouraging us to explore, experiment, and refine our understanding of modern computer vision methods.

11. References

Ultralytics YOLOv8 Documentation. Available at: https://docs.ultralytics.com/

Roboflow License Plate Dataset. Dataset source page: https://roboflow.com/

README

- Vehicle-registration-plate-1/ → Dataset (+ data.yaml)
- train/, valid/, test/ → Image splits
- train_results/ → YOLO training outputs
    - weights/best.pt
    - results.png (training curves)
    - confusion_matrix.png, labels.jpg, PR curves, etc.
- Final_project_YOLO_model.ipynb → Main Notebook
- Deep_Learning_Project_CNN → Main Notebook
- Faster_RCNN_Confusion_Matrix → Image
- Faster_RCNN_loss_curves → Image
- Faster_RCNN_training_log → csv
- FasterRCNN_ValDetections → csv

After the opening. ipynb file just mount the drive and copy the dataset path of Vehicle-registration-plate-1 and copy it in the model path and run it.
I have already run the model and trained it and saved it on a drive path called train_results.