# `YOLOv8-Assisted Vehicle Detection

NIKITHA R L , POORNAMIRTHA S,VIGNESHWARAN S, Ms DHANYA K R

Student ,Assistant Professor ,Department of Computer Science with Artificial Intelligence & Data Science, KPR College of Arts, Science and Research, Coimbatore, India.

**E-mail:**nikithaa0777@gmail.com

**Abstract**

Vehicle detection is an important application of computer vision that can be used to monitor and understand traffic conditions. With the increasing number of vehicles on roads, manually monitoring traffic has become difficult and time-consuming. This project focuses on developing a simple Vehicle Detection System that can automatically identify vehicles from images or video using computer vision and image-processing techniques.The proposed system takes a video or image as input and processes it to detect vehicles present on the road. A machine learning or object detection model is used to identify different types of vehicles such as cars, buses, trucks, and motorcycles. Once a vehicle is detected, the system displays a bounding box around it and can also count the number of vehicles present in the given scene. This allows traffic information to be obtained quickly without requiring continuous manual observation.

The main objective of this project is to understand the basic concepts of Artificial Intelligence, Computer Vision, Image Processing, and Object Detection and apply them to a real-world problem. The system can be implemented using tools such as Python, OpenCV, and a pre-trained object detection model. The project is designed to be simple, efficient, and easy to understand, making it suitable for basic traffic-monitoring applications. In the future, the system can be improved by adding features such as vehicle speed detection, number plate recognition, traffic-density analysis, accident detection, and traffic-rule violation detection. Such improvements could make the system more useful for smart transportation and traffic management. Overall, this project demonstrates how computer vision and AI can be used to automatically detect and count vehicles, providing a basic foundation for intelligent traffic monitoring systems.

**Keywords**

VehicleDetection, YOLOv8, Object Detection, Computer Vision, Deep Learning ,Intelligent Transportation, Traffic Monitoring.

## 1. Introduction

Artificial Intelligence (AI) has transformed computer vision by enabling machines to analyse and interpret visual information from images and video. Computer vision systems are now used in transportation, surveillance, healthcare, retail, manufacturing and intelligent automation. Within transportation, automatic vehicle detection is a fundamental capability because it provides information about the presence, position and category of road users without requiring continuous manual observation.

Traffic scenes are visually complex. A single image may contain cars, buses, trucks, motorbikes and other vehicle types at different scalesandpositions.Vehiclesmayalsobepartiallyoccluded,affectedby lighting conditions, or surrounded by visually similar objects. A practical detection system therefore needs to perform both localization and classification. Object detection methods address this requirement by predicting a bounding box, a class label and a confidence score for each detected object.

YOLO, meaning You Only Look Once ,is a one-stage object-detection family designed to predict object locations and classes efficiently. YOLOv8 is a later Ultralytics implementation that provides a lightweight YOLOv8n model as well as larger model variants. The YOLOv8 architecture uses a backbone for feature extraction, a neck for multi-scale feature fusion and a detection head for predicting object classes and bounding boxes. This architecture provides a practical balance between computational cost and detection performance.

The present project applies YOLOv8n to a six-class vehicle dataset. The implementation uses Google Colab for model training and Google Drive for dataset and model storage. The project work flow consists of dataset preparation, YAML configuration, preprocessing and augmentation during training, transfer learning from pretrained YOLOv8n weights, validation using standard object-detection metrics and prediction on a new road image.

The main objective is to develop and evaluate a model capable of detecting six vehicle categories: Car, Threewheel, Bus, Truck, Motorbike and Van. The final system is intended as a basic vehicle-detection component that can be extended toward traffic monitoring, vehicle counting, road analytics and intelligent transportation applications.

## 2. Related Works

Object detection has progressed from region-based convolutional neural networks to efficient one-stage detectors such as the YOLO family. Earlier traffic-detection studies demonstrated that YOLO-based methods can provide practical real-time detection, although crowded scenes and small objects remain challenging. More recent studies have investigated lightweight YOLO variants and improved YOLOv8 architectures for vehicle detection. Recent work on improved YOLOv8 vehicle detection has explored backbone replacement, attention mechanisms and localization-loss improvements to increase accuracy and efficiency.

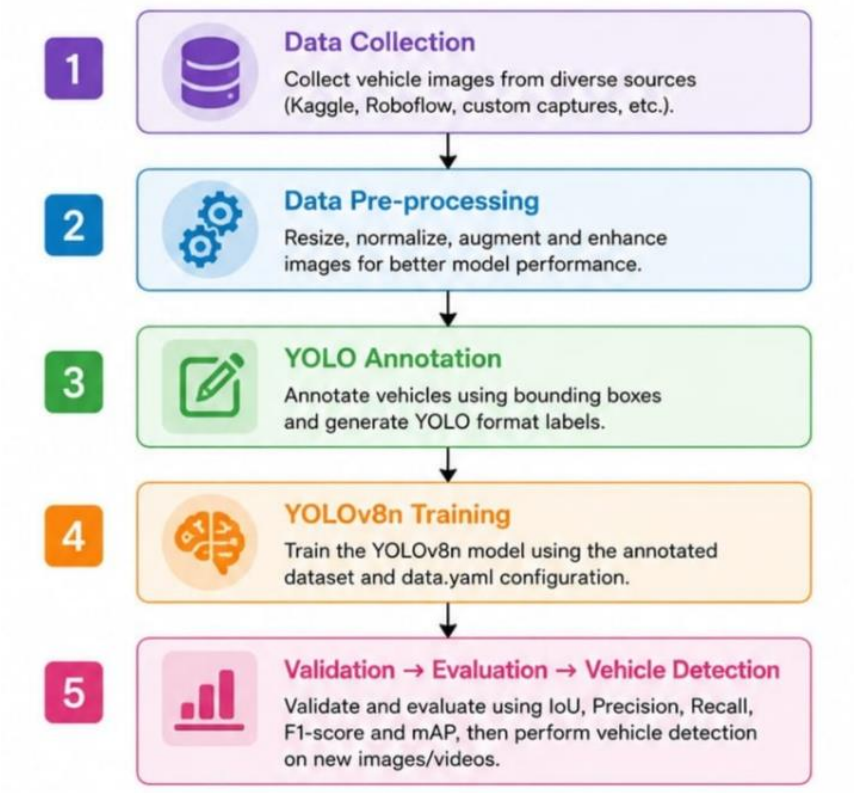
Table 1. Comparative summary of existing approaches for vehicle detection

| Author / Study | Method | Reported result | Limitation / observation |
|---|---|---|---|
| Rahmanetal.(2021) | YOLOv5+ensemble for dense traffic | mAP@0.5=0.458ona dense-traffic dataset | Crowded scenes and small objects remain difficult. |
| Geetha(2024) | Comparison of YOLOv5variants | Evaluated precision,recall, F1 and mAP across model variants | Performance varies with model size and vehicle class. |
| Guoetal.(2024) | ImprovedYOLOv8 vehicle detector | Reported strong class-wise vehicle recognition after architecture improvements | Higher complexity is introduced through model modifications. |
| UltralyticsYOLOv8 | YOLOv8family/ YOLOv8n | Lightweight model with efficient object detection | Performance still depends on dataset quality and scene conditions. |
| Present project | YOLOv8ntransfer learning | mAP@0.5=98.0%; mAP@0.5:0.95=90.6% | Motorbike has comparatively lower strict-IoU performance. |

The reviewed work indicates that YOLO-based detectors are suitable for traffic applications because they perform localization and classification in a single detection pipeline. However, real-world vehicle detection remains sensitive to small objects, occlusion, crowded scenes, illumination and dataset diversity. These observations motivate the use of a pretrained YOLOv8n model with augmentation and multi-class evaluation in the present project.

## 3. Proposed Work – Methodology

The proposed vehicle-detection methodology follows the same structured sequence used in the reference project: data collection, preprocessing, annotation and class configuration, model training, validation and prediction. The dataset is organized in YOLO format and described through a data.yaml configuration file. The trained model is then evaluated using IoU, Precision, Recall, F1-score and mean Average Precision measures.




The final pipeline begins with the vehicle dataset and its YOLO-format labels. The data.yaml file identifies the training and validation image directories and maps class identifiers to vehicle names. The YOLOv8n pretrained model is then fine-tuned for 120 epochs. The resulting best weights are validated and used for inference on a new image.

### 3.1 Data Collection

The project uses a vehicle dataset organized for YOLO object detection. The notebook confirms the presence of train, valid and classes.txt components, and the final data.yaml points to the vehicle dataset directory. Six vehicle categories are defined: Car, Threewheel, Bus, Truck, Motorbike and Van. During training, the dataset scanner reported 2,123 training images, including 2,100 images with vehicle annotations and 23 background images. The validation scan reported 902 images, including 899 images with annotations and 3 background images.

Table 2. Dataset distribution used in the project

| Dataset split | Images | Annotated images | Background images |
|---|---|---|---|
| Training | 2,123 | 2,100 | 23 |
| Validation | 902 | 899 | 3 |
| Totalscanned | 3,025 | 2,999 | 26 |

The final validation run used 902 images and 1,149 annotated vehicle instances. The object instances are distributed across the six classes as shown below. These values represent annotated objects in the validation set rather than the number of images belonging exclusively to each class.

| Class | Validation instances |
|---|---|
| Car | 200 |
| Threewheel | 227 |
| Bus | 185 |
| Truck | 151 |
| Motorbike | 216 |
| Van | 170 |
| **Total** | **1149** |

### 3.2 Data Pre-processing

Before model training, the dataset was organized into YOLO-compatible training and validation directories. The final YAML configuration specifies the dataset path, training images and validation images, followed by the six class names. The training configuration uses a 640 × 640 image size. YOLOv8 applies its standard image preprocessing and augmentation pipeline during training, including mosaic augmentation, horizontal flipping, scaling, HSV colour augmentation and Rand Augment-related transformations recorded in the training configuration.

The preprocessing stage is important because object-detection models require a consistent representation of both images and bounding-box annotations. The annotation files use YOLO format, in which each object is represented by a class identifier followed by normalized bounding-box coordinates.

#### 3.2.1 Image Resizing

The training configuration specifies an image size of 640 × 640 pixels. Resizing or letterbox processing allows the model to operate on a consistent target resolution while retaining the spatial information needed for bounding-box prediction.

$I_{resized} = \text{Resize}(I, 640, 640)$

#### 3.2.2 Pixel Value Scaling

Digital images contain pixel intensities represented on a finite numerical range. Neural network pipelines commonly transform these values into a normalized numerical representation during preprocessing. This provides a consistent input range for convolutional computation.

$I_{norm}(x,y) = I_{resized}(x,y) / 255$

#### 3.2.3 Bounding Box Normalization

YOLO annotation format represents each object using five values: class, center-x, center-y, width and height. The coordinates and dimensions are normalized with respect to image width and height. For an image with width W and height H:

(class, x, y, w, h)

| Parameter | Formula | Meaning |
|---|---|---|
| x | x / W | Normalized center X-coordinate |
| y | y / H | Normalized center Y-coordinate |
| w | w / W | Normalized bounding-box width |
| h | h / H | Normalized bounding-box height |

**Where:**

- **x, y** = bounding-box center coordinates in pixels
- **w, h** = bounding-box width and height in pixels
- **W, H** = image width and height in pixels
- The resulting values are normally between **0 and 1**

This representation makes bounding-box coordinates independent of the original image dimensions and is suitable for the YOLO training pipeline.

### 3.2.4 Bounding Box Representation

If a bounding box is initially expressed using the left, top, right, and bottom coordinates ($x_{min}$, $y_{min}$, $x_{max}$, $y_{max}$), its center coordinates and dimensions can be calculated as follows:

$x_c = (x_{min} + x_{max}) / 2$

$y_c = (y_{min} + y_{max}) / 2$

$w = x_{max} - x_{min}$

$h = y_{max} - y_{min}$

The resulting center coordinates and dimensions are then normalized using the image width (W) and height (H) before being written to the YOLO label file.

$x_c' = x_c / W$

$y_c' = y_c / H$

$w' = w / W$

$h' = h / H$

The final YOLO annotation format is: $(Class, x_c', y_c', w', h')$

### 3.2.5 Data Augmentation

Data augmentation increases the diversity of training samples without requiring new manually labelled images. The training configuration records mosaic augmentation, horizontal flipping, scaling, translation and HSV-based colour augmentation. These transformations help the model encounter changes in vehicle position, scale and appearance during training.

For horizontal flipping, a normalized horizontal coordinate can be expressed as $x' = 1 - x$, while the vertical coordinate remains unchanged. Scaling and translation modify object size and position,allowing the detector to learn vehicle appearance under different spatial conditions.

**Horizontal Flipping**

For an image with normalized horizontal coordinate *x*, horizontal flipping can be represented as:

$x' = 1 - x$

The vertical coordinate remains unchanged:

$y' = y$

This allows the model to learn from vehicles appearing in different horizontal orientations and positions within a road scene.

Scaling

Scaling changes the size of the image or detected vehicle by a scale factor *s*:

$x' = sx$

$y' = sy$

$w' = sw$

$h' = sh$

Scaling helps the model detect vehicles appearing at different sizes and distances from the camera.

**Translation**

Image translation shifts a vehicle horizontally and vertically:

$x' = x + t_x$

$y' = y + t_y$

where $t_x$ and $t_y$ represent horizontal and vertical translation values. Translation helps the model learn to detect vehicles at different positions within an image.

**HSV Colour Augmentation**

Colour variations can be introduced by modifying the Hue (H), Saturation (S), and Value (V) components of an image. In general, the transformed image can be represented as:

$I'HSV = Augment(IHSV, \Delta H, \Delta S, \Delta V)$

where $\Delta H$, $\Delta S$, and $\Delta V$ represent changes applied to the hue, saturation, and brightness components. This helps the model handle differences in illumination and colour conditions that may occur in different traffic scenes.

### 3.2.6 Mosaic Augmentation

Mosaic augmentation combines multiple training images into a single composite training sample. Conceptually, four images can be combined as:

$I_{mosaic} = \text{Combine}(I_1, I_2, I_3, I_4)$

The corresponding bounding-box annotations are transformed to match their new positions in the composite image. Mosaic augmentation exposes the detector to different object scales, positions and backgrounds within a single training sample and is especially useful for traffic scenes containing multiple vehicles.

### 3.2.7 Class Encoding

The project defines six detection classes. The class identifier is stored as the first value in each YOLO annotation. The mapping used in the final data.yaml configuration is:

| Class ID | Vehicle class |
|---|---|
| 0 | Car |
| 1 | Threewheel |
| 2 | Bus |
| 3 | Truck |
| 4 | Motorbike |
| 5 | Van |

### 3.2.8 Final Pre-processed Dataset

After organization and preprocessing, the images and corresponding labels were supplied to the YOLOv8 training pipeline. The overall preprocessing sequence can be summarized as:

Original Image→Resize→Pixel Scaling →Bounding-Box Normalization→Data Augmentation→ YOLOv8 Training

The final data.yaml configuration specifies the dataset root, training image directory, validation image directory and the six class names. This configuration provides the information required by Ultralytics during training and validation.

### 3.3 Model Selection and Training

YOLOv8n was selected as the detection model because the nano variant provides a lightweight implementation suitable for student-level experimentation and fast inference. The project loads pretrained YOLOv8n weights and fine-tunes them on the vehicle dataset. The notebook records a model summary of approximately 3.0 million parameters and 8.1 GFLOPs for the fused model used during validation.

Figure 1. Simplified YOLOv8n architecture used for vehicle detection

The YOLOv8 architecture can be described through three major stages. The Backbone extracts visual features from the input image. The Neck combines feature information at different scales so that both relatively large and small vehicles can be represented. The Detection Head produces bounding-box predictions, class scores and confidence values. The present model is configured for six classes.

Table 3. Model training configuration

| Parameter | Project setting |
|---|---|
| Model | YOLOv8n pretrained |
| Epochs | 120 |
| Image size | 640×640 |
| Batch size | 16 |
| GPU | NVIDIATeslaT4 |
| Optimizer | AdamW(auto-selected) |
| Best weights | best.pt |

Training was completed for 120 epochs in approximately 2.037 hours. The notebook confirms that both last.pt and best.pt were saved, with best.pt subsequently used for validation and prediction.

### 3.4 Model Evaluation

After training, the YOLOv8n model was evaluated on the validation dataset. The evaluation measures how accurately the model localizes and classifies vehicle instances. The reference project uses IoU, Precision, Recall, F1-score, mAP@0.5 and mAP@0.5:0.95, and the same evaluation framework is followed here.

#### 3.4.1 Intersection over Union (IoU)

Intersection over Union measures the overlap between a predicted bounding box and the ground-truth bounding box. It is defined as:

IoU = Area of Intersection / Area of Union

A higher IoU indicates that the predicted bounding box is more closely aligned with the actual vehicle region.

#### 3.4.2 Precision

Precision measures how many of the detections predicted by the model are correct. Precision = TP / (TP + FP)

Here, TP denotes true positives and FP denotes false positives. Higher precision indicates fewer incorrect positive detections.

#### 3.4.3 Recall

Recall measures how many of the actual vehicle instances are successfully detected. Recall = TP / (TP + FN)

Here, FN denotes false negatives. Higher recall indicates that the detector misses fewer actual vehicles.

#### 3.4.4 F1-Score

F1-score provides a balanced measure of Precision and Recall and is expressed as: F1 = 2 × Precision × Recall / (Precision + Recall)

A higher F1-score indicates a stronger balance between correct detections and the ability to find the actual objects. The notebook also records an F1-confidence curve among the available YOLO evaluation curves.

### 3.4.5 Mean Average Precision at IoU 0.5 (mAP@0.5)

mAP@0.5 measures the average detection performance across the six vehicle classes when a predicted bounding box is considered correct at an IoU threshold of 0.5.

$mAP@0.5 = (1/N) \Sigma AP_i$

where $AP_i$ is the Average Precision of class i and N is the number of classes. The project has N = 6.

The stricter mAP@0.5 : 0.95 metric averages detection quality over multiple IoU thresholds from 0.50 to 0.95. It therefore provides a more demanding measure of bounding-box localization quality.

## 4 RESULTS AND DISCUSSION

### 4.1 System Configuration

The proposed vehicle detection system was developed using Google Colab with Python 3.12.13, PyTorch 2.11.0 and the Ultralytics YOLO framework. Training was performed with CUDA GPU acceleration on an NVIDIA Tesla T4 with approximately 14,913 MiB of GPU memory. The fused YOLOv8n model used for final validation contains 73 layers, approximately 3,006,818 parameters and approximately 8.1 GFLOPs.

The data set contains six vehicle classes. The validation evaluation used 902 images and 1,149 annotated instances in the principal final validation output. Google Drive was used to store the dataset and trained weights, while Google Colab provided the computational environment for training, validation and testing.

Table 4. System configuration

| System component | Configuration |
|---|---|
| Development environment | Google Colab |
| Python | 3.12.13 |
| Deeplearning framework | PyTorch2.11.0 |
| Ultralytics | 8.4.x recorded in notebook |
| GPU | NVIDIA TeslaT4,14,913MiB |
| Inputsize | 640×640 |
| Training duration | 120epochs; approximately 2.037hours |

## Performance Analysis

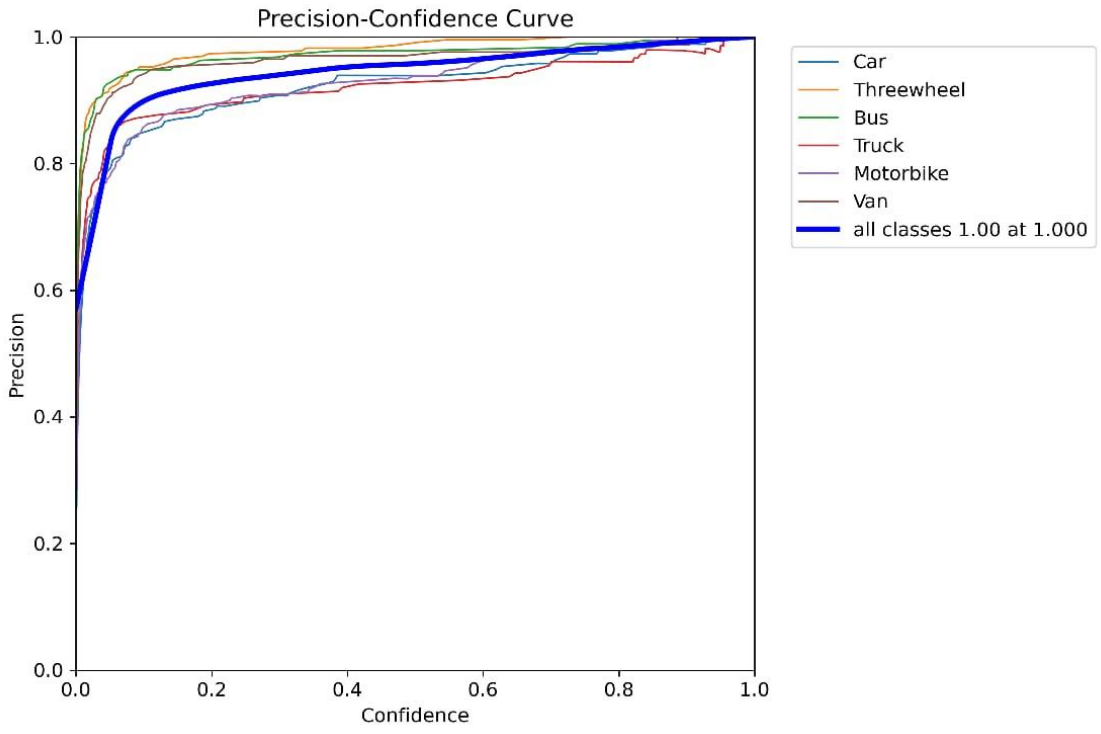
**Class-wise performance metrics**

| Class | Images | Instances | Precision | Recall | mAP@50 | mAP@50–95 |
|---|---|---|---|---|---|---|
| Car | 182 | 201 | 0.969 | 0.910 | 0.966 | 0.931 |
| Three wheel | 167 | 227 | 0.999 | 0.960 | 0.989 | 0.916 |
| Bus | 162 | 185 | 0.984 | 0.973 | 0.994 | 0.954 |
| Truck | 123 | 151 | 0.961 | 0.979 | 0.974 | 0.906 |
| Motorbike | 165 | 216 | 0.975 | 0.887 | 0.969 | 0.771 |
| Van | 157 | 170 | 0.979 | 0.947 | 0.987 | 0.957 |
| **Overall** | **903** | **1150** | **0.978** | **0.943** | **0.980** | **0.906** |

The performance of the proposed YOLOv8-based vehicle detection system was evaluated using Precision, Recall, mAP@0.5 and mAP@0.5:0.95. The final validation output reports approximately 97.8% Precision, 94.3% Recall, 98.0% mAP@0.5 and 90.6% mAP@0.5:0.95.

The high mAP@0.5 value indicates strong detection performance when a 0.50 IoU threshold is used. The lower mAP@0.5:0.95 value reflects the stricter requirement for accurate localization across multiple IoU thresholds. The difference between these metrics is expected in object detection because tighter localization requirements are more difficult to satisfy

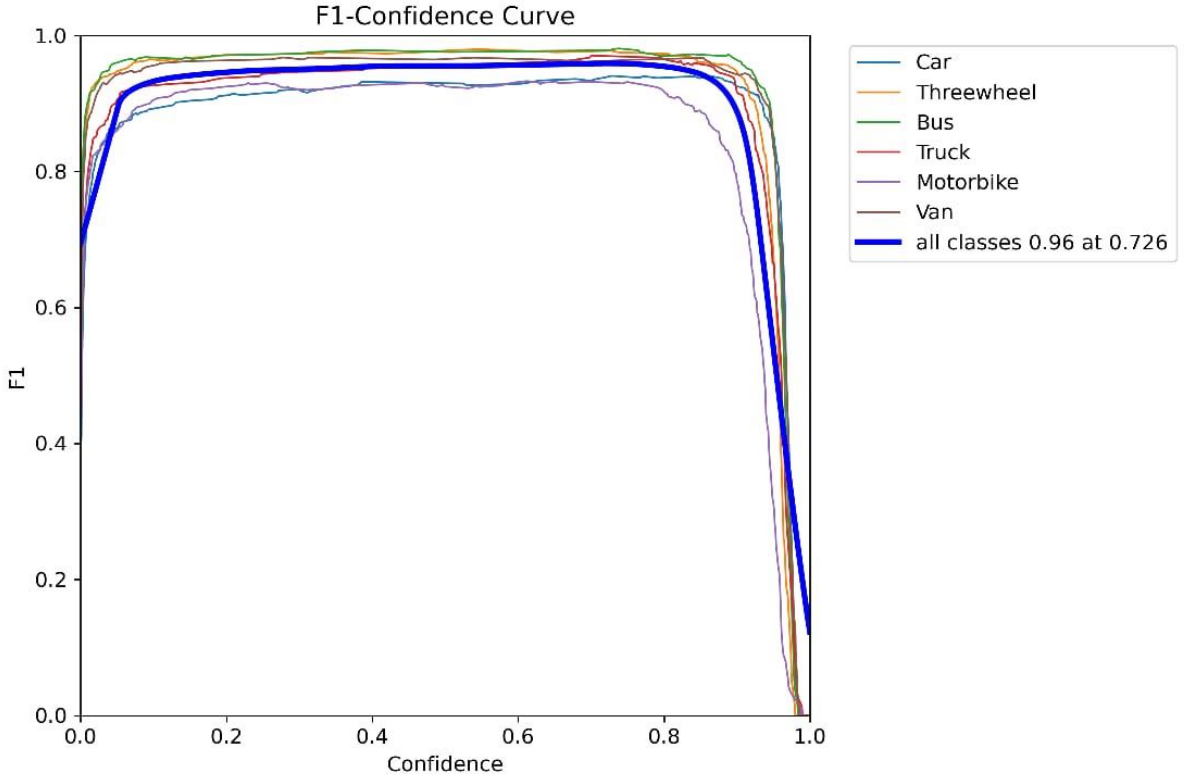
**Precision–Confidence Curve**



The Precision–Confidence Curve illustrates the relationship between the confidence threshold of the YOLOv8 vehicle detection model and its precision for each vehicle class. The graph shows separate curves for Car, Threewheel, Bus, Truck, Motorbike, and Van, along with an overall curve representing all classes. Precision generally increases as the confidence threshold becomes higher, indicating that predictions retained at higher confidence levels are more reliable. The Bus and Threewheel classes maintain particularly high precision across a wide range of confidence values, while Car and Truck show slightly more variation. At a confidence level close to 1.0, the overall precision reaches approximately 1.00. This indicates that the trained model produces highly reliable predictions when only high-confidence detections are considered.

**F1–Confidence Curve**

The F1–Confidence Curve represents the variation of the F1-score with respect to the confidence threshold used by the YOLOv8 model. The F1-score combines precision and recall into a single performance measure, making it useful for selecting an appropriate detection confidence threshold. From the graph, the overall F1-score reaches approximately 0.96 at a confidence threshold of 0.726, as indicated by the thick blue curve. The individual vehicle classes also maintain high F1-scores over a broad range of confidence values. Most classes remain close to the upper region of the graph before gradually decreasing at very high confidence thresholds. This decrease occurs because an excessively high threshold can reject valid detections. Therefore, a confidence value around 0.726 provides a good balance between precision and recall for the vehicle detection model.



**Normalized Confusion Matrix**

The normalized confusion matrix presents the classification performance of the YOLOv8 model across the six vehicle categories: Car, Threewheel, Bus, Truck, Motorbike, and Van, along with the background category. The diagonal values represent correctly classified instances, with values of 0.84 for Car, 0.78 for Threewheel, 0.83 for Bus, 0.86 for Truck, 0.93 for Motorbike, and 0.76 for Van. Among the vehicle classes, Motorbike achieves the highest diagonal value of 0.93, indicating strong classification performance, while Van has a comparatively lower value of 0.76. Some off-diagonal values indicate confusion between visually similar vehicle categories, such as Car and Bus or Car and Van. Overall, the matrix demonstrates that the model successfully distinguishes most vehicle classes while showing some classification confusion between particular categories.

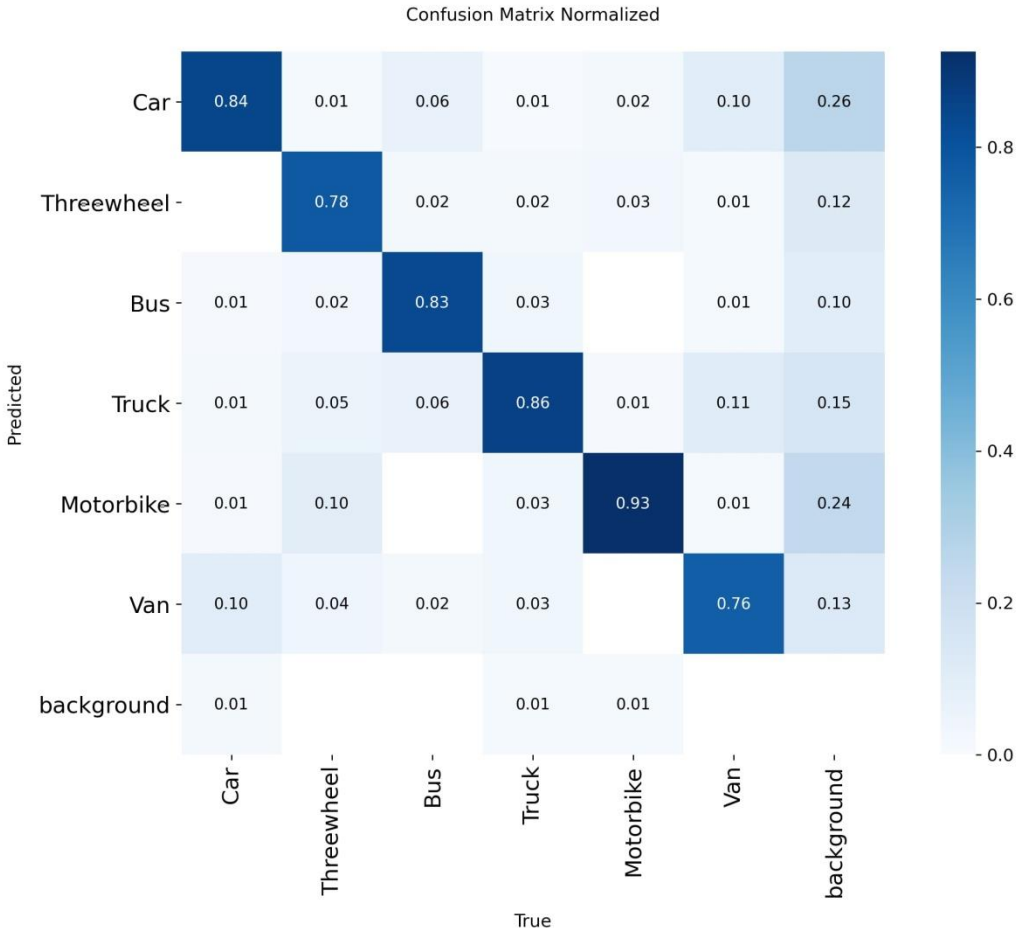


Figure 2. Sample prediction image

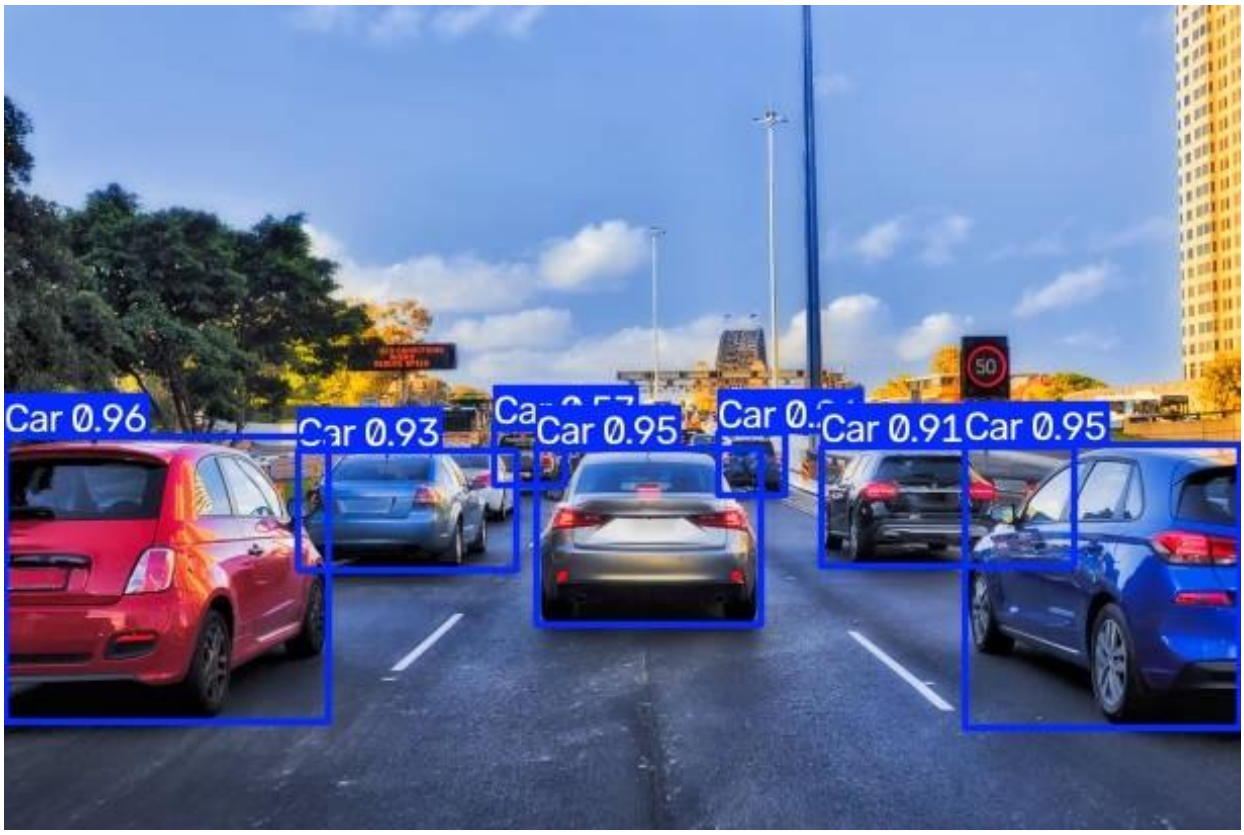



Figure6. YOLOv8n prediction on the project test image.

The note book output reports even detected cars.

The trained best.pt weights were loaded and applied to a new road image using a confidence threshold of 0.25. The notebook output reports that the image contained seven detected cars. The displayed prediction confirms the intended object-detection behaviour: each detected vehicle is surrounded by a bounding box and assigned a class label with a confidence score.

This prediction demonstrates the transition from model evaluation to practical inference. Unlike a classification system that would return only a single image-level label, the YOLOv8 detector identifies multiple objects and localizes them individually within the same traffic scene.

## 4.2 Discussion of Results

The results demonstrate that the proposed YOLOv8n model performs strongly on the validation dataset. The overall Precision of 97.8% indicates that the large majority of reported detections are correct, while the Recall of 94.3% indicates that most annotated vehicle instances are successfully detected. The mAP@0.5 of 98.0% further confirms strong class-wise detection at the standard 0.50 IoU threshold.

The stricter mAP@0.5:0.95 score of 90.6% is lower, as expected, because the metric requires better localization over a range of IoU thresholds. The class-wise results show that Bus has the highest mAP@0.5, while Motorbike is the most challenging category under strict localization. Differences between classes may be associated with object size, viewpoint, appearance, occlusion, class distribution and the visual complexity of road scenes.

The model was trained with augmentation settings including mosaic augmentation and horizontal flipping. Such transformations help expose the detector to different spatial arrangements and visual conditions. Never the less, strong validation performance should not be interpreted as a guarantee of identical performance in every real-world traffic environment. A larger and more diverse evaluation set containing different cameras, weather conditions, night scenes and heavily crowded roads would provide a stronger test of generalization.

The demonstrated prediction on a new image provides qualitative evidence that the trained model can localize multiple vehicles simultaneously. Future development can extend the detector to video streams, vehicle counting, multi-object tracking, traffic-density estimation and real-time dashboard applications.

## Conclusion

The proposed vehicle detection system demonstrates the practical application of YOLOv8n for multi-class vehicle detection. The project follows a systematic workflow consisting of dataset collection and organization, YOLO-format annotation handling, preprocessing and augmentation, transfer learning, model training, validation and image-based prediction.

The YOLOv8n model was trained for 120 epochs using a 640 × 640 input size and a batch size of 16 on an NVIDIA Tesla T4 GPU. The final validation output reports an overall Precision of 97.8%, Recall of 94.3%, mAP@0.5 of 98.0% and mAP@0.5:0.95 of 90.6%. The results show strong detection performance across Car, Threewheel, Bus, Truck, Motorbike and Van classes. The Motorbike class requires comparatively more improvement under strict IoU evaluation.

The prediction stage further demonstrates that the trained best.pt weights can identify multiple vehicles in a previously unseen road image. The system therefore provides a useful foundation for intelligent transportation and traffic-monitoring applications. Future work can focus on larger and more diverse datasets, improved class balance, challenging night and weather scenes, video-based detection, vehicle tracking and automated vehicle counting.

Project outcome: A trained YOLOv8n multi-class vehicle detector with six vehicle categories and strong validation performance, together with a demonstrated inference result on a new road image.

## References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016.

[2] R. Rahman, Z. Bin Azad, and M. B. Hasan, “Densely-Populated Traffic Detection using YOLOv5 and Non-Maximum Suppression Ensembling,” arXiv:2108.12118, 2021.

[3] A. S. Geetha, “Comparing YOLOv5 Variants for Vehicle Detection: A Performance Analysis,” arXiv:2408.12550, 2024.

[4] H. Guo, Y. Zhang, L. Chen, and A. A. Khan, “Research on vehicle detection based on improved YOLOv8 network,” arXiv:2501.00300, 2024.

[5] G. Jocher, A. Chaurasia, and J. Qiu, “Ultralytics YOLOv8,” Ultralytics software release, 2023.

[6] Ultralytics, “YOLO Architecture Explained,” Ultralytics documentation, accessed 2026.

[7] Ultralytics, “YOLOv8 Model Documentation,” Ultralytics documentation, accessed 2026.

[8] Projectnotebook,“SMART_CODERS,”GoogleColabimplementationandvalidationoutputforthe vehicle detection system, 2026.

[9] Vehicle dataset used in the project, organized in YOLO format with six vehicle categories: Car, Threewheel, Bus, Truck, Motorbike and Van.