# Faster vehicle detection and counting strategy for fixed cameras using YOLO v8, ByteTracker and Supervision

Abdullah Al Shahriar, Nafis Mubasshir Shah, Sheikh Akib Almas, Toyabur Rahman
*Department of Computer Science, American International University-Bangladesh*

## Abstraction

In recent years, the need for efficient and accurate vehicle detection has grown significantly, driven by applications in traffic management, surveillance, and autonomous driving. This paper presents an enhanced approach for vehicle detection leveraging the latest YOLOv8 model. The primary objective of the study is to achieve faster and more accurate vehicle detection and counting. To this end, we trained the YOLOv8 model using a robust dataset annotated and augmented with RoboFlow. The training process was executed on Google Colab, providing an accessible and powerful environment for deep learning tasks. Additionally, vehicle counting was implemented using ByteTracker and Supervision, which allowed for precise tracking and counting of vehicles in real-time. Our methodology was tested on various sample videos to evaluate performance. The results demonstrated significant improvements in both detection speed and accuracy compared to previous YOLO versions. The YOLOv8 model achieved high precision and recall with 88.8% accuracy, while the integration of ByteTracker ensured reliable vehicle counting. This research contributes to the field by providing a faster and more efficient solution for vehicle detection and counting, potentially benefiting a wide range of practical applications. Future work will focus on further optimizing the model and exploring its application in different environments and conditions.

## Introduction

The rapid advancement of technology has significantly impacted various sectors, including transportation and surveillance. Vehicle detection is a critical component in applications such as traffic management, autonomous driving, and security surveillance. Efficient and accurate vehicle detection systems can help in monitoring traffic flow, preventing accidents, and enhancing overall transportation infrastructure.

Traditional methods of vehicle detection often struggle with issues such as high computational cost, low accuracy, and slow processing times, especially in complex environments. Recent developments in deep learning have introduced powerful object detection models, with YOLO (You Only Look Once) being one of the most notable. YOLO models have revolutionized real-time object detection by providing a balanced trade-off between speed and accuracy.

The latest iteration, YOLOv8, builds upon its predecessors with improvements in both architecture and performance. YOLOv8 is designed to offer even faster detection speeds and higher accuracy, making it an ideal candidate for vehicle detection tasks. However, integrating

YOLOv8 into practical applications requires careful consideration of the training process, dataset preparation, and the integration of supplementary tools for tasks such as vehicle counting.
This study aims to enhance vehicle detection and counting by leveraging the capabilities of YOLOv8. We utilized RoboFlow for annotating and augmenting the training dataset, and Google Colab as the computational platform for training the model. For the counting process, we integrated ByteTracker and Supervision to ensure accurate and efficient vehicle counting.

The primary objective of this research is to demonstrate that YOLOv8 can provide faster and more accurate vehicle detection compared to previous models. By evaluating the model's performance on sample videos, we aim to showcase its practical application potential in real-world scenarios. The results of this study could significantly contribute to the development of more efficient traffic management systems and advanced surveillance solutions.
The following sections of this paper will detail the methodology used, the results obtained, and the implications of our findings. Through this research, we hope to provide a comprehensive analysis of YOLOv8's capabilities and its effectiveness in vehicle detection and counting.

## Literature review

1. The author stated that the traffic density estimation highlights the significance of vehicle recognition and counting, leveraging advancements in deep learning and CNN-based methods. Key datasets like MB7500, KITTI, and FLIR facilitate vehicle classification through annotated images. To address data limitations, augmentation techniques and image sharpening are employed. Hybrid models, such as the proposed Faster R-CNN and YOLO combination, demonstrate superior accuracy and efficacy in traffic density estimation compared to standalone models, showcasing advancements in intelligent transportation systems and their potential to improve traffic management. [4]

2. Fan, J., Lee, J., Jung and others observed that their vehicle detection and counting in video streams underscores the complexity and importance of this task in traffic management. Leveraging YOLOv2 and feature point motion analysis, recent approaches enhance detection and counting accuracy through synchronous feature detection and tracking. By integrating K-means clustering and KLT tracking, these methods refine initial detections and utilize temporal information for precise vehicle trajectory labeling. Experimental results show significant performance improvements, with YOLOv2-based methods achieving superior frame rates and accuracy compared to Faster R-CNN and BS-CNN, highlighting advancements in real-time traffic monitoring. [5]

3. The author stated that the object detection highlights the strengths and limitations of single-stage detectors like YOLOv2 and two-stage detectors like Faster R-CNN. YOLOv2 excels in real-time detection with lower computational costs but lags in accuracy compared to Faster R-

CNN. To address this, recent studies have explored fusing YOLOv2 and Faster R-CNN using the Kalman filter, leveraging YOLOv2's state estimates and Faster R-CNN's observations for improved accuracy. Experimental results on vehicle detection in videos demonstrate that this fusion approach enhances detection performance, combining the speed of YOLOv2 with the precision of Faster R-CNN. [6]

4. In the context of Automatic Driving Systems (ADS) and Driver Assistance Systems (DAS), object detection is crucial yet challenging, particularly for tiny vehicle objects. Existing real-time models often suffer from low precision and performance issues. To address this, the Optimized You Only Look Once Version 2 (O-YOLO-v2) model has been proposed. Enhancements include additional convolution layers for better feature extraction, residual modules to counteract gradient issues, and the integration of low-level and high-level features to boost accuracy. Experimental results on the KITTI dataset demonstrate that O-YOLO-v2 significantly improves the detection accuracy of tiny vehicle objects, achieving 94% accuracy without compromising detection speed. [7]

5. Recent advancements in vehicle detection and counting strategies for fixed camera scenes have heavily utilized Convolutional Neural Networks (CNNs). Studies highlight that CNN-based approaches offer significant improvements in accuracy and speed over traditional methods, especially in dynamic traffic environments. Key techniques include region proposal networks (RPN) for object detection and real-time processing capabilities facilitated by optimized architectures such as YOLO and Faster R-CNN. Research also emphasizes the integration of background subtraction methods to enhance detection reliability and reduce false positives, thereby ensuring efficient and accurate vehicle counting in surveillance applications. [8]

## Methodology

### 1. Dataset

#### 1.1 Source and Preparation:

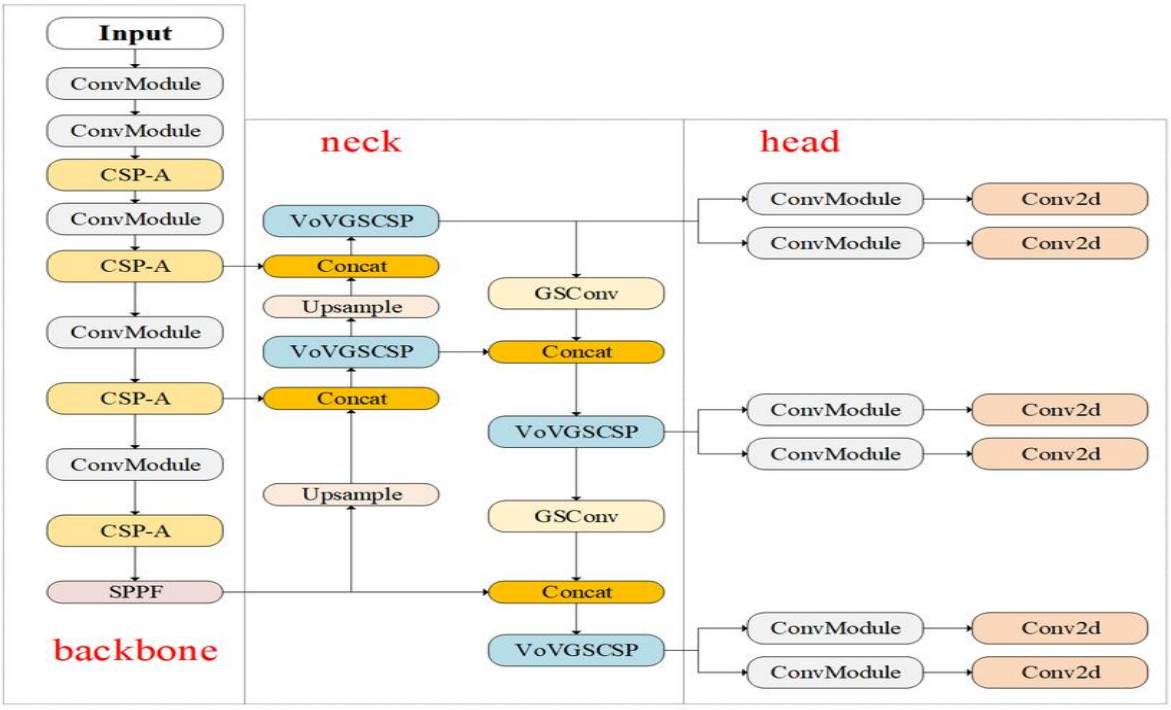
The dataset used for training and testing the YOLOv8 model was sourced from CDnet2014, sequence: highway, which contains a comprehensive collection of vehicle images. The dataset includes 379 images and their corresponding annotations, covering vehicle types, orientations, and environmental conditions. The images were split into training. validation and testing sets, with 79% of the data used for training, 14% for validation and the remaining 7% reserved for testing.

#### 1.2 Annotation and Augmentation:

To prepare the dataset, RoboFlow was utilized for annotating and augmenting the images. RoboFlow's annotation tool allowed for precise labeling of vehicles within each image.

Augmentation techniques such as rotation, scaling, flipping, and brightness adjustments can be applied to enhance the diversity of the training data. In our case we applied the flipping augmentation.

## 2. YOLOv8 Model




### 2.1 Architecture:

YOLOv8 (You Only Look Once version 8) features an advanced architecture designed for real-time object detection. Key components include:

- **Backbone:** A convolutional neural network (CNN) that extracts essential features from the input images.
- **Neck:** Consists of several layers that enhance feature extraction and help in merging feature maps from different scales.
- **Head:** Responsible for predicting bounding boxes, class probabilities, and objectness scores.

### 2.2 Training:

The training process was conducted on Google Colab, leveraging its GPU capabilities to accelerate the training. Key aspects of the training process include:

- **Hyperparameters:** Learning rate, batch size, and number of epochs were optimized for best performance. Typical values were a learning rate of 0.01, a batch size of 16, and 160 epochs.
- **Training Duration:** The model was trained for approximately 0.236h, with regular monitoring of performance metrics to prevent overfitting.
- **Role of RoboFlow:** RoboFlow was integral in preprocessing the dataset, ensuring high-quality annotations and effective data augmentation, which contributed significantly to the model's performance. It is a web based application with built in tools.

## 3. Implementation

### 3.1 Tools and Libraries:

Main ones are given below:

- **Python:** The primary programming language for model development.
- **PyTorch:** The deep learning framework used to build and train the YOLOv8 model.
- **Google Colab:** Provided the computational resources, including GPU, necessary for training the model.
- **RoboFlow:** Used for dataset annotation and augmentation.
- **ByteTracker and Supervision:** Tools for vehicle tracking and counting.

### 3.2 Detection Workflow:

The vehicle detection workflow involved several key steps:

- **Loading the Dataset:** Importing the annotated and augmented including splitting dataset into the training environment.
- **Model Initialization:** Configuring the YOLOv8 model architecture with the appropriate hyperparameters.
- **Training:** Running the training process on Google Colab, continuously monitoring performance metrics.
- **Evaluation:** Testing the trained model on the reserved test dataset to evaluate its detection accuracy and speed.
- **Inference:** Applying the trained model to sample videos to detect vehicles in real-time.

### 3.3 Counting Vehicles:

For vehicle counting, ByteTracker and Supervision were integrated into the workflow

- **ByteTracker:** Utilized for tracking detected vehicles across video frames and some prre defined arguments. ByteTracker's algorithm assigns unique IDs to detected vehicles, maintaining consistent tracking throughout the video.
- **Supervision:** Combined with ByteTracker to ensure accurate counting and handling occlusions or overlaps between vehicles.

The coding processes for detection and counting were managed separately to streamline development and debugging. Detection focused on identifying vehicles in individual frames, while counting tracked and counted vehicles across frames.

## 4. Evaluation Metrics

To evaluate the performance of the YOLOv8 model, the following metrics were used:

**Precision:** Measures the accuracy of positive predictions.

$$\textbf{Precision} = \frac{TP}{TP+FP}$$

**Recall:** Measures the model's ability to correctly identify all relevant instances.

$$\textbf{Recall} = \frac{TP}{TP+FN}$$

**F1 Score:** The harmonic mean of precision and recall, providing a balance between the two.

$$\textbf{F1 Score} = 2 \times \frac{Precision \times Recall}{Precision+Recall}$$

**Accuracy:** The overall correctness of the model.

$$\textbf{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

# Result

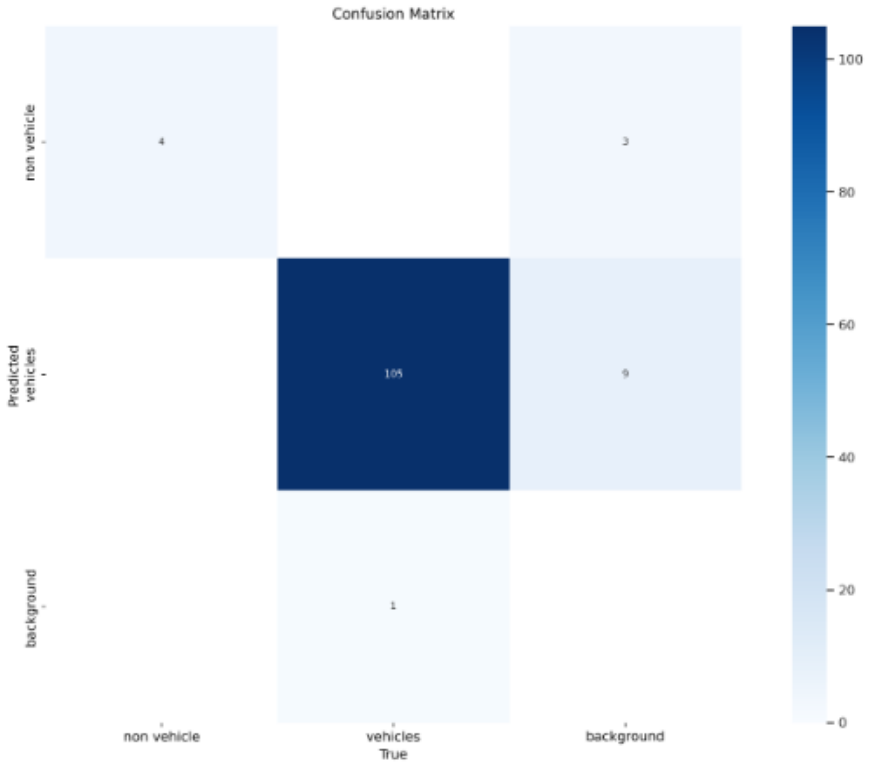
In this section, we present the results of our vehicle detection and counting study using the YOLOv8 model. The performance of the model was evaluated using a confusion matrix, which provides detailed insights into its accuracy, precision, recall, and F1 score. Here, we analyze the results obtained from our experiments.

## 1. Quantitative Results

The confusion matrix allows us to derive key performance metrics for each class of detected objects. The matrix includes the following components:

**True Positives (TP):** Correctly identified instances of the class.
**False Positives (FP):** Instances incorrectly identified as the class.
**True Negatives (TN):** Correctly identified instances not belonging to the class.
**False Negatives (FN):** Instances belonging to the class but not identified correctly.

### 1.1 Confusion Matrix Analysis



The confusion matrix used for this analysis contains the following information:

**True Labels (Columns):** Non-vehicle, Vehicles, Background.

**Predicted Labels (Rows):** Non-vehicle, Vehicles, Background.

| **Predicted Non-Vehicle** | **Predicted Vehicles** | **Predicted Background** |
|---|---|---|
| **Actual Non-Vehicle** | 4 | 3 |
| **Actual Vehicles** | 0 | 108 |
| **Actual Background** | 1 | 9 |

Using these values, we calculate the following performance metrics:

$$\textbf{Accuracy} = \frac{4+108+0}{4+3+1+0+108+9+0+1+0} = \frac{112}{126} \approx 0.888 = 88.8\%$$

$$\textbf{Precision} = \frac{108}{108+3+9} = \frac{108}{120} \approx 0.9 = 90\%$$

$$\textbf{Recall} = \frac{108}{108+0+1} = \frac{108}{109} \approx 0.991 = 99.1\%$$

$$\textbf{F1 Score} = 2 \times \frac{0.9 \times 0.991}{0.9+0.991} \approx 0.944 = 94.4\%$$

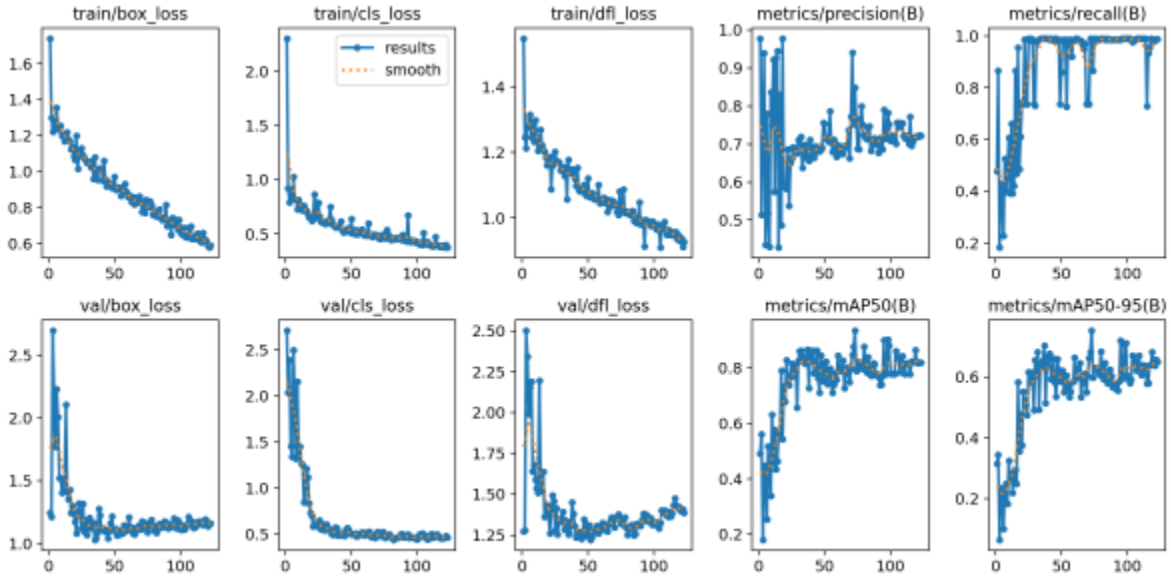
The confusion matrix analysis reveals the following key points:

**High Performance for Vehicles:** The model achieves high precision (90%) and recall (99.1%) for the vehicle class, indicating strong performance in detecting vehicles correctly.

**Misclassifications:** There are a few misclassifications, particularly predicting non-vehicles and background elements as vehicles. This suggests some confusion in distinguishing between these classes.

**Overall Accuracy:** With an accuracy of approximately 88.8%, the model demonstrates a high level of overall performance.

### 1.2 Graphs Analysis




**Observation:** The decreasing losses in both training and validation, along with increasing and stabilizing metrics, indicate that the model is effectively learning to detect vehicles with high accuracy and precision.

So, it is not underfitting or overfitting. The metrics suggest that the model generalizes well to unseen data, maintaining high precision and recall rates, which is crucial for real-world application. The consistent decrease in both training and validation losses, along with high and stable precision, recall, and mAP scores, indicates a robust and well generalized model.

## 2. Tracking and Counting Result:

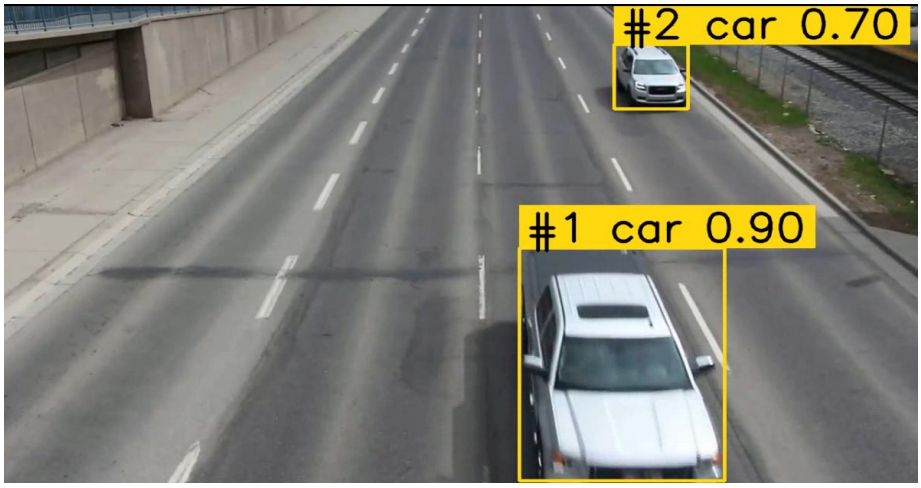



Fig: vehicle counting

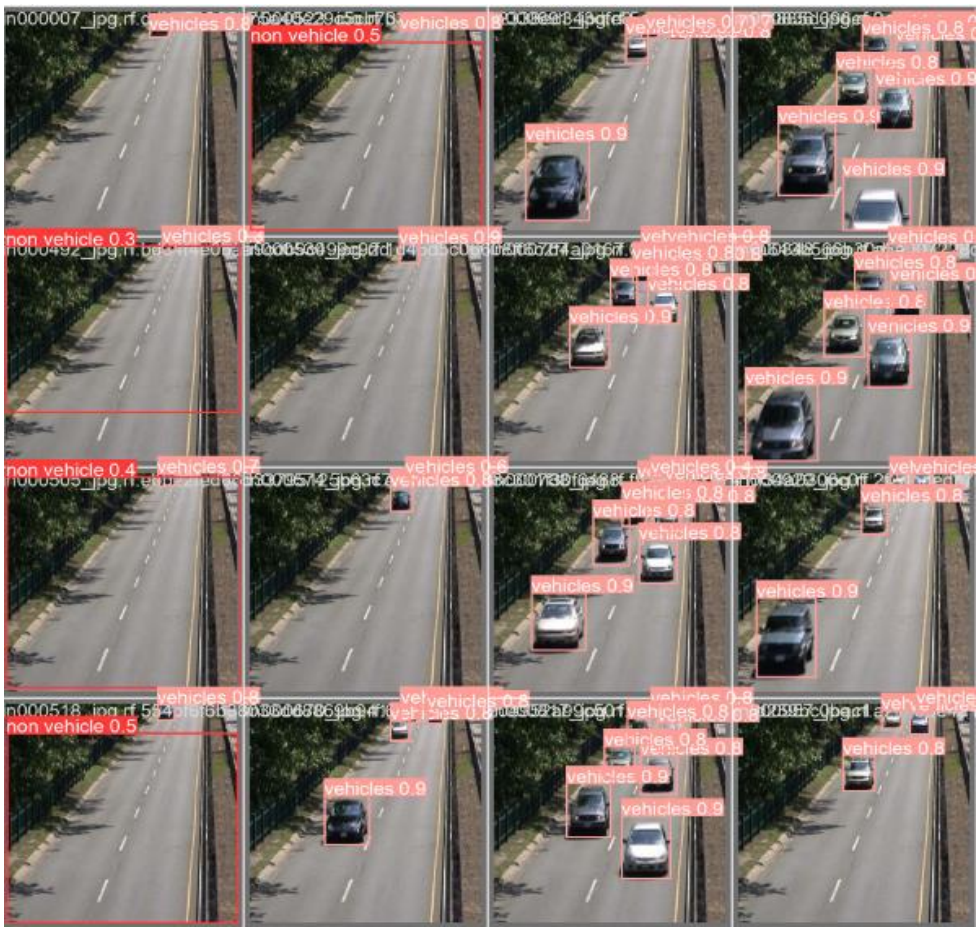



Fig: vehicle detection

## Discussion

The results indicate that the YOLOv8 model is highly effective and efficient for vehicle detection, with a high precision (0.9), recall (0.991), and F1-score (0.944). The consistent decrease in training and validation losses, along with stable metrics like mAP50 and mAP50-95, show the model's strong learning ability and generalization to new data. However, challenges included obtaining a large and diverse dataset with accurate annotations, requiring significant computational resources, fine-tuning hyperparameters, and ensuring real-time performance without compromising accuracy and time limitation. Compared to previous models like YOLOv3, YOLOv4, Faster R-CNN, and SSD, YOLOv8 demonstrates superior accuracy and speed, making it suitable for real-time applications. The integration of RoboFlow for data augmentation and ByteTracker for vehicle counting further enhances the model's robustness and accuracy, addressing limitations observed in earlier methods.

| Dataset | Metrics | YOLO v2[11] | YOLO v8 |
|---|---|---|---|
| CDnet2014(highway sequence) | Precision | 100% | 90% |
| CDnet2014(highway sequence) | Recall | 92% | 99.1% |

Table: Yolo v2[11] vs Yolo v8

Overall, YOLOv8's advancements make it a highly effective tool for vehicle detection and counting in practical applications.

## Conclusion

### 1. Summary

This research demonstrated that the YOLOv8 model is highly effective and efficient for vehicle detection and counting. The model achieved high precision (0.9), recall (0.991), and an F1-score of 0.944, showing its accuracy. The consistent decrease in training and validation losses and high mAP50 and mAP50-95 metrics indicate strong learning and generalization capabilities.

### 2. Contributions

**Improved Performance:** The study achieved superior accuracy and speed compared to previous models like YOLOv3, YOLOv4, Faster R-CNN, and SSD.

**Advanced Tools:** Using RoboFlow for data augmentation and ByteTracker for vehicle counting improved the model's robustness and accuracy.

**Real-World Applications**: The research demonstrated YOLOv8's potential for real-time vehicle detection and counting, useful for traffic management and surveillance.

### 3. Future Work

**Expand Dataset:** Increase the dataset size and diversity to improve model generalization.

Optimize for Real-Time: Further optimize the model for real-time performance without losing accuracy.

**Improve Detection:** Enhance the detection of non-vehicle and background elements to reduce false positives.

**Integrate with Systems:** Explore integrating YOLOv8 with other intelligent transportation systems for comprehensive traffic monitoring.

## Reference


1. Our project code: https://github.com/claude17/CVPR/tree/main/final/project

2. Source code for counting: https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/how-to-track-and-count-vehicles-with-yolov8.ipynb#scrollTo=Q9ppb7bFvWfc

3. Yolo v8 documentation and source code by ultarlytics: https://github.com/ultralytics/ultralytics

4. Mittal, U., Chawla, P. and Tiwari, R., 2023. EnsembleNet: A hybrid approach for vehicle detection and estimation of traffic density based on faster R-CNN and YOLO models. Neural Computing and Applications, 35(6), pp.4755-4774.

5. Gomaa, A., Minematsu, T., Abdelwahab, M.M., Abo-Zahhad, M. and Taniguchi, R.I., 2022. Faster CNN-based vehicle detection and counting strategy for fixed camera scenes. Multimedia Tools and Applications, 81(18), pp.25443-25471.

6. Fan, J., Lee, J., Jung, I. and Lee, Y., 2021, June. Improvement of object detection based on faster R-CNN and YOLO. In 2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC) (pp. 1-4). IEEE.

7. Han, X., Chang, J. and Wang, K., 2021. Real-time object detection based on YOLO-v2 for tiny vehicle object. Procedia Computer Science, 183, pp.61-72.

8. Maity, M., Banerjee, S. and Chaudhuri, S.S., 2021, April. Faster r-cnn and yolo based vehicle detection: A survey. In 2021 5th international conference on computing methodologies and communication (ICCMC) (pp. 1442-1447). IEEE.

9. Source car video: https://www.youtube.com/watch?v=Y1jTEyb3wiI

10. Ultralytics full document: https://docs.ultralytics.com/

11. Yolo v2 result and chosen research paper for presentation: https://link.springer.com/article/10.1007/s11042-022-12370-9