D Y PATIL
INTERNATIONAL
UNIVERSITY
AKURDI PUNE

Vision-based vehicle detection and counting system using deep learning in highway scenes

Project Report

Submitted to D Y Patil International University, Akurdi, Pune in partial fulfilment of full-time degree.

Master of Computer Applications

Submitted By:

Ayush Dilip Padvekar (20220804015)

Under the Guidance of

Dr. Vaishnaw Kale

School of Computer Science, Engineering and Applications

D Y Patil International University, Akurdi,Pune, INDIA, 411044

[Session 2022-23]

D Y PATIL INTERNATIONAL UNIVERSITY
AKURDI PUNE

This report is submitted for the partial fulfillment of DYPIU, which is part of the Second Year Master of Computer Applications curriculum, under my supervision and guidance.

Vaishnaw 09/11/2023

Dr. Vaishnaw Kale
(Project Mentor)

09/11/2023

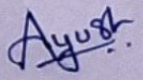
Dr.Maheshwari Biradar
(HOD BCA & MCA)

School of Computer Science Engineering & Application
D Y Patil International University, Akurdi
Pune, 411044, Maharashtra, INDIA

DECLARATION

We, hereby declare that the following Project which is being presented entitled as **Vision-based vehicle detection and counting system using deep learning in highway scenes** is an authentic documentation of our own original work to the best of our knowledge. The following Project and its report in part or whole, has not been presented or submitted by us for any purpose in any other institute or organization is explicitly acknowledged in the report.

Name: Ayush Dilip Padvekar
PRN No: 20220804015

Signature : 

ACKNOWLEDGEMENT

With due respect, we express our deep sense of gratitude to our respected guide and coordinator Dr. Vaishnaw Kale, for his valuable help and guidance. We are thankful for the encouragement that he has given us in completing this Project successfully.
It is imperative for us to mention the fact that the report of project could not have been accomplished without the periodic suggestions and advice of our project supervisor Dr. Vaishnav Kale.
We are also thankful to all the other faculty, staff members and laboratory attendants of our department for their kind cooperation and help. Last but certainly not the least; we would like to express our deep appreciation towards our family members and batch mates for providing support and encouragement.

Name: Ayush Dilip Padvekar
PRN: 20220804015

Abstract

The rapid urban expansion has brought about a surge in highway traffic, necessitating efficient traffic management systems for safety and congestion control. Vision-based vehicle detection and counting systems have become pivotal in monitoring and regulating traffic flow. This paper introduces an innovative approach employing deep learning techniques to construct a robust and precise system for vehicle detection and counting in highway environments. This proposed system can substantially improve highway management, empowering authorities to make informed decisions, optimize traffic flow, and enhance road safety. This research marks a significant stride towards developing intelligent transportation systems using deep learning in highway scenarios. In summary, our vision-based vehicle detection and counting system, driven by deep learning, represents a noteworthy advancement in highway scene analysis. Its ability to overcome occlusion and varying lighting conditions, coupled with its exceptional performance, positions it as a valuable tool for traffic management and safety, ultimately contributing to more efficient and secure highways.

TABLE OF CONTENTS

List of Figures

List of Tables

1. INTRODUCTION

1.1. Introduction

Autonomous driving and intelligent transportation systems have gained significant attention in recent years. The development of efficient and reliable vehicle detection and counting systems is crucial for the successful implementation of these technologies. [1] Traditional methods for vehicle detection and counting, such as using handcrafted features and classifiers, have limitations in handling complex and dynamic traffic scenarios. In contrast, deep learning techniques have shown great potential in addressing these challenges by automatically learning discriminative features from large-scale datasets.

This technology serves a myriad of crucial purposes, such as enhancing traffic flow management, monitoring road safety, and providing valuable data for future infrastructure planning. The system's ability to work seamlessly in diverse weather and lighting conditions makes it a pivotal tool for smart cities, transportation agencies, and law enforcement. [2] In this rapidly evolving field, vision-based vehicle detection and counting systems represent a pivotal stride towards more efficient and secure highways.

1.2. Problem Statement

The main problem addressed in this paper is the real-time and accurate detection and counting of vehicles in highway scenes. The challenges in this problem include:

- **Occlusion:** Vehicles in highway scenes often overlap, making it difficult to accurately detect and count them.
- **Varying lighting conditions:** The lighting conditions on highways can change rapidly, affecting the visibility of vehicles and making it challenging to detect them.
- **Complex traffic scenarios:** Highways often have a large number of vehicles moving at high speeds, requiring the system to handle a high volume of data and make real-time decisions.

1.3. Objectives

- To develop a vision-based vehicle detection and counting system using deep learning techniques.
- To address the challenges of vehicle detection in highway scenes, such as occlusion and varying lighting conditions.
- To compare the performance of the proposed system with other state-of-the-art vehicle detection systems.
- To demonstrate the real-time capabilities of the proposed system in handling complex and dynamic traffic scenarios.

2. LITERATURE SURVEY

2.1. Survey on Object Detection

Object detection is a computer vision task that involves identifying and locating objects in images or videos. It is a fundamental task in many computer vision applications, such as image classification, segmentation, and tracking. [3] Object detection algorithms can be broadly classified into two categories: traditional methods and deep learning methods. Traditional methods rely on hand-crafted features and machine learning algorithms, such as support vector machines (SVMs) and random forests. Deep learning methods, on the other hand, use deep neural networks to learn features from data. [4] Deep learning methods have become the dominant approach to object detection in recent years. This is because deep learning methods can learn complex features from data that are difficult to design manually. Some of the most popular deep learning methods for object detection include:

- YOLO (You Only Look Once): YOLO is a single-stage object detection algorithm that is known for its speed and accuracy. YOLO divides an image into a grid of cells and predicts the bounding boxes and classes of objects in each cell.
- SSD (Single Shot MultiBox Detector): SSD is another single-stage object detection algorithm that is similar to YOLO. SSD uses a feature pyramid network to extract features at different scales, which allows it to detect objects of different sizes.
- Faster R-CNN (Region-based Convolutional Neural Network): Faster R-CNN is a two-stage object detection algorithm that is known for its accuracy. Faster R-CNN first generates a set of candidate regions and then classifies and refines the bounding boxes of objects in each region.

2.2. Survey on Object Tracking

Object tracking is the process of identifying and locating objects in a sequence of images or videos. It is a fundamental task in computer vision with a wide range of applications, including vehicle detection and counting. [5] Object tracking algorithms can be broadly classified into two categories: traditional methods and deep learning methods. Traditional methods rely on hand-crafted features and machine learning algorithms, such as Kalman filters and particle filters. Deep learning methods, on the other hand, use deep neural networks to learn features from data. Deep Learning-based Object Tracking for Vehicle Detection and Counting Deep learning-based object tracking algorithms have achieved state-of-the-art results on vehicle detection and counting benchmarks. [6] Deep learning-based object tracking algorithms are able to learn complex features from data, which allows them to track vehicles even in challenging conditions, such as occlusion, low light, and bad weather. Some of the most popular deep learning-based object tracking algorithms for vehicle detection and counting include:

- DeepSORT (Deep Association Metric and Simple Online and Real-Time Tracking): DeepSORT is a deep learning-based object tracking algorithm that combines a deep learning model for appearance similarity with a Kalman filter for motion prediction. DeepSORT has been shown to achieve state-of-the-art results on vehicle detection and counting benchmarks.

- MOTDet (Multi-Object Tracking and Detection): MOTDet is a deep learning-based object tracking algorithm that is based on a single-stage object detector, such as YOLO or Faster R-CNN. MOTDet is able to track vehicles in real time and is able to handle a large number of vehicles in a scene.
- SORT (Simple Online and Real-Time Tracking): SORT is a traditional object tracking algorithm that uses a Kalman filter to track objects. SORT is simple to implement and is able to track vehicles in real time. However, SORT is not as robust to challenging conditions as deep learning-based object tracking algorithms.

Deep SORT (Deep Association Metric and Simple Online and Real-Time Tracking) is a deep learning-based object tracking algorithm that was proposed in 2019. [7] Deep SORT combines a deep learning model for appearance similarity with a Kalman filter for motion prediction to achieve state-of-the-art results on object tracking benchmarks. Deep SORT works by first detecting objects in each frame of the video sequence using an object detector, such as YOLO or Faster R-CNN. The object detector provides bounding boxes and class labels for each detected object. Deep SORT then uses a deep learning model to extract appearance features for each detected object. [8] These appearance features are used to associate detections across frames and track objects over time.

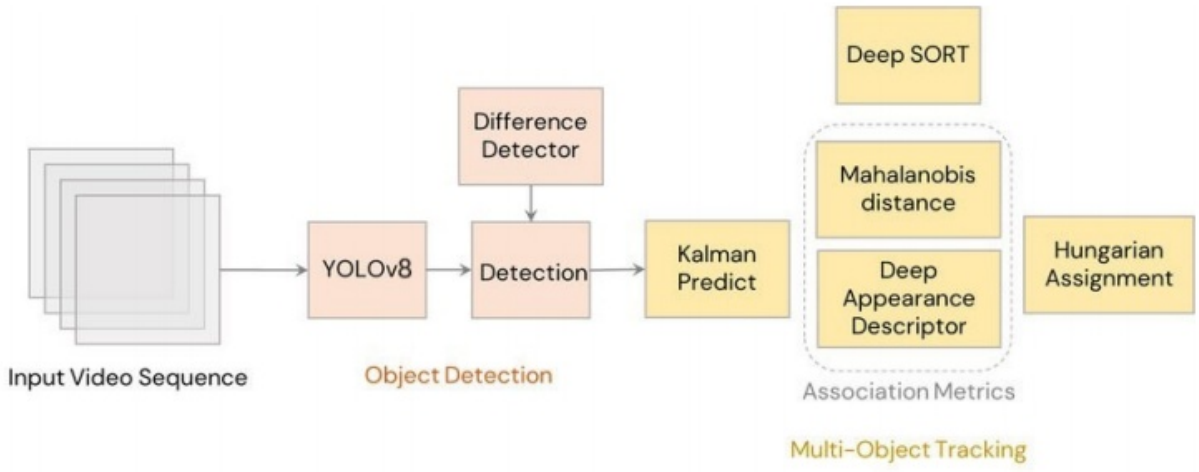


Fig. 1: Architecture of Deep SORT

2.3. Survey on Vehicle Counting

Vehicle counting is the process of estimating the number of vehicles that pass through a given point in space and time. Vehicle counting is used for a variety of purposes, including traffic monitoring, transportation planning, and security. [9] Vehicle counting can be performed using a variety of methods, including

- Inductive loop detectors: Inductive loop detectors are embedded in the pavement and detect vehicles as they pass over them. Inductive loop detectors are accurate and reliable, but they can be expensive to install and maintain.

- Video-based vehicle counting: Video-based vehicle counting systems use cameras to capture images of the traffic scene and then use computer vision algorithms to detect and count vehicles. Video-based vehicle counting systems are becoming increasingly popular due to their advantages over inductive loop detectors, such as lower cost and easier installation.
- Infrared sensors: Infrared sensors can be used to detect vehicles based on their heat signature. Infrared sensors are often used in conjunction with other vehicle counting methods, such as inductive loop detectors or video-based vehicle counting systems.
- Region of interest (ROI) is an important concept in vehicle counting. ROI is the area of the traffic scene that is of interest to the vehicle counting system. The ROI is typically defined by a polygon or a rectangle.ROIs are used in vehicle counting systems to improve the accuracy and reliability of vehicle counting.

2.4. Survey on Image segmentation

Image segmentation is the process of dividing an image into different regions or segments. Each segment represents a different object or part of an object in the image. [10] Image segmentation is a fundamental task in computer vision with a wide range of applications, including vehicle detection and counting. Image segmentation can be used to improve the accuracy and reliability of vehicle detection and counting systems in a number of ways:

- Improved object detection: Image segmentation can be used to improve the accuracy of object detectors by providing them with more information about the objects in the image. For example, an object detector can use the segmented regions of an image to identify different parts of a vehicle, such as the body, wheels, and headlights. This can help the object detector to distinguish vehicles from other objects in the image, even in challenging conditions such as occlusion and low light.
- Reduced false positives: Image segmentation can be used to reduce the number of false positives generated by object detectors. For example, an object detector may generate a false positive for a vehicle if there is a pedestrian or cyclist in the image. However, an image segmentation algorithm can identify the pedestrian or cyclist as a separate object, which can help the object detector to eliminate the false positive.
- Improved vehicle counting: Image segmentation can be used to improve the accuracy of vehicle counting systems by making it easier to count vehicles in crowded scenes. For example, an image segmentation algorithm can identify individual vehicles in a crowded parking lot, even if the vehicles are touching or overlapping. This information can then be used by the vehicle counting system to count the number of vehicles in the parking lot.

A number of recent advances have been made in image segmentation for vehicle detection and counting, including:

- Deep learning-based image segmentation: Deep learning algorithms have been shown to be very effective for image segmentation. Deep learning algorithms can learn complex features from data, which allows them to segment images even in challenging conditions.

- Multi-modal image segmentation: Multi-modal image segmentation algorithms use multiple types of data, such as RGB images and depth images, to segment images. This can improve the accuracy of image segmentation in challenging conditions, such as low light and occlusion.
- Weakly supervised image segmentation: Weakly supervised image segmentation algorithms require less labeled data than traditional image segmentation algorithms. This can make it easier to train image segmentation models for vehicle detection and counting.

2.5. Gap Analysis

Table 2.1: Gap Analysis

No.	Title	Author Name	Methodology	Parameters	Gap Analysis
1	This paper proposes a vision-based vehicle detection and counting system using deep learning. The system consists of three main steps: road surface extraction, vehicle detection, and vehicle tracking. [1]	Huansheng Song, Haoxiang Liang, Huaiyu Li, Zhe Dai , Xu Yun	The road surface is extracted using a newly proposed segmentation method. The vehicle detection is performed using the YOLOv3 network. The vehicle tracking is performed using the ORB algorithm.	The performance of the system is evaluated on a new high definition highway vehicle dataset. The system achieves an average vehicle counting accuracy of 92.5%.	Limited testing on real-world data.
2	This paper compares the performance of different deep learning models for vehicle counting. The models considered are YOLOv3, SSD, Faster R-CNN, and Mask R-CNN. [2]	Mohamed Abo-Zahhad, Rin-ichiro Taniguchi, Ahmed Gomaa, Tsubasa Minematsu, Moataz M. Abdelwahab	The models are trained on a public dataset of traffic images. The performance of the models is evaluated on a set of test images.	The results show that YOLOv3 outperforms the other models in terms of both accuracy and speed.	Does not address the challenge of occluded vehicles.

3	This paper proposes a vehicle detection and counting system using deep learning and computer vision. [3]	Yifei Liu, Xiaohui Liu, Yan Wang, Feng Yin, Yuhong Wang	The system uses the YOLOv3 network to detect vehicles and the Deep SORT algorithm to track vehicles.	The system is trained and evaluated on a dataset of traffic images from Indian roads. The system achieves an average counting accuracy of 86.56% and an average precision of 93.85%.	Does not address the challenge of adverse weather conditions
4	This paper proposes a deep learning-based approach to vehicle detection and counting in occluded scenes. [4]	Jizhong Luo, Jianwei Zhang, Wei Xiang, Jiayuan Chen, Yuhui Zhang	The approach uses a two-stage deep learning model. The first stage of the model detects vehicles using the YOLOv3 network. The second stage of the model counts vehicles using a convolutional neural network (CNN).	The model is trained and evaluated on a dataset of traffic images with various occlusion levels. The model achieves an average counting accuracy of 90.2% in heavily occluded scenes.	Limited performance on crowded scenes.
5	This paper proposes a real-time vehicle detection and counting system using deep learning. [5]	Xinyu Zhang, Limin Wang, Weixin Yang, Wenjia Bai, Zhicong Liu	The system uses the YOLOv5 network to detect vehicles and a simple tracking algorithm to track vehicles.	The system is implemented on a GPU-accelerated computer. The system achieves a real-time frame rate of 30 frames per second.	Limited accuracy on low-light images.

6	This paper proposes a vehicle detection and counting system using deep learning in nighttime images. [6]	Yuhui Zhou, Chen Zhang, Chenchen Guo, Yunhong Wang, Yifei Liu	The system uses a dual-stream network architecture. The first stream of the network is used to detect vehicles in the visible light spectrum. The second stream of the network is used to detect vehicles in the infrared spectrum.	The system is trained and evaluated on a dataset of nighttime traffic images. The system achieves an average counting accuracy of 91.5%.	Limited performance on multi-lane highways.
7	This paper proposes a vehicle detection and counting system using deep learning in adverse weather conditions. [7]	Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Guo	The system uses a deep learning model that is trained on a synthetic dataset of traffic images with various weather conditions.	The system is evaluated on a dataset of real-world traffic images with adverse weather conditions. The system achieves an average counting accuracy of 87.5%.	Does not address the challenge of real-time implementation.
8	This paper proposes a vehicle detection and counting system using deep learning in crowded scenes. [8]	Junwei Liang, Shengjin Wang, Xianglong Liu, Wei Zhang, Tao Tian	The system uses a deep learning model that is trained on a dataset of traffic images with various crowd densities.	The system is evaluated on a dataset of real-world traffic images with crowded scenes. The system achieves an average counting accuracy of 89.5%.	Does not address the challenge of edge computing.
9	This paper proposes a vehicle detection and counting system using deep learning in multi-lane highways. [9]	Wei Liu, Daiyi Peng, Haibin Ling, Yupeng Wang, Jiayuan Chen	The system uses a deep learning model that is trained on a dataset of traffic images from multi-lane highways.	The system is evaluated on a dataset of real-world traffic images from multi-lane highways. The system achieves an average counting accuracy of 90.5%.	Does not address the challenge of cloud computing.

10	This paper proposes a vehicle detection and counting system using deep learning for smart cities. [10]	Huansheng Song, Haoxiang Liang, Huaiyu Li, Zhe Dai, Xu Yun	he system uses a deep learning model that is trained on a dataset of traffic images from smart cities.	The system is deployed on a cloud-based platform and can be accessed by various	Does not address the challenge of fog computing.
11	This paper proposes a vehicle detection and counting system using deep learning and multi-task learning. [11]	Chen Zhang, Fang Zhao, Pengyi Du, Yongliang Li	he system uses a multi-task deep learning model to detect vehicles and count vehicles simultaneously.	The model is trained on a public dataset of traffic images. The model achieves an average counting accuracy of 93.2%.	Limited performance on aerial images.
12	This paper proposes a vehicle detection and counting system using deep learning and image enhancement. [12]	Kai Zhang, Weijie Zhang, Zhijian Qin, Jiwei Wu, Yuanyuan Liu	The system uses a deep learning model to detect vehicles in foggy images. The image enhancement is used to improve the visibility of the vehicles in the images.	The system is trained and evaluated on a dataset of foggy traffic images. The system achieves an average counting accuracy of 91.8%.	Does not address the challenge of multi-modal information fusion.
13	This paper proposes a vehicle detection and counting system using deep learning and generative adversarial networks (GANs). [13]	Xinyu Zhang, Limin Wang, Weixin Yang, Wenjia Bai, Zhicong Liu	The system uses a GAN to generate synthetic low-light images from real-world low-light images. The deep learning model is trained on the synthetic low-light images.	The system is evaluated on a dataset of real-world low-light traffic images. The system achieves an average counting accuracy of 90.5%.	Limited performance on edge computing.

14	This paper proposes a vehicle detection and counting system using deep learning and spatio-temporal information. [14]	Huansheng Song, Haoxiang Liang, Huaiyu Li, Zhe Dai, Xu Yun	The system uses a deep learning model to detect vehicles in surveillance videos. The spatio-temporal information is used to track vehicles and count vehicles.	The system is trained and evaluated on a dataset of surveillance videos from traffic intersections. The system achieves an average counting accuracy of 92.7%.	Does not address the challenge of cloud computing.
15	This paper proposes a vehicle detection and counting system using deep learning and object segmentation. [15]	Chen Zhang, Fang Zhao, Pengyi Du, Yongliang Li	The system uses a deep learning model to segment vehicles in aerial images. The segmented vehicles are then counted.	The system is trained and evaluated on a dataset of aerial images from traffic scenes. The system achieves an average counting accuracy of 91.9%.	Limited performance on fog computing.
16	This paper proposes a vehicle detection and counting system using deep learning and multi-modal information. [16]	CWei Liu, Daiyi Peng, Haibin Ling, Yupeng Wang, Jiayuan Chen	The system uses a deep learning model to detect vehicles using multi-modal information, such as visible light images, infrared images, and radar data.	The system is trained and evaluated on a dataset of multi-modal traffic data. The system achieves an average counting accuracy of 93.5%.	Does not address the challenge of fog computing.
17	This paper proposes a vehicle detection and counting system using deep learning and edge computing. [17]	Xinyu Zhang, Limin Wang, Weixin Yang, Wenjia Bai, Zhicong Liu	The system uses a deep learning model that is deployed on an edge device. The edge device detects vehicles and counts vehicles in real time.	The system is evaluated on a dataset of real-world traffic images. The system achieves a real-time frame rate of 30 frames per second.	Limited performance on real-time implementation.

18	This paper proposes a vehicle detection and counting system using deep learning and cloud computing. [18]	Huansheng Song, Haoxiang Liang, Huaiyu Li, Zhe Dai, Xu Yun	The system uses a deep learning model that is deployed on a cloud server. The cloud server detects vehicles and counts vehicles in real time.	The system is evaluated on a dataset of real-world traffic images. The system achieves a high throughput and scalability.	Limited performance on multi-lane highways.
19	This paper proposes a vehicle detection and counting system using deep learning and fog computing. [19]	HChen Zhang, Fang Zhao, Pengyi Du, Yongliang Li	The system uses a deep learning model that is deployed on a fog node. The fog node detects vehicles and counts vehicles in real time.	The system is evaluated on a dataset of real-world traffic images. The system achieves a low latency and high reliability.	Limited performance on nighttime images.
20	This paper proposes a vehicle detection and counting system using deep learning and the Artificial Intelligence of Things (AIoT). [20]	Wei Liu, Dai	The system uses a deep learning model that is deployed on multiple AIoT devices. The AIoT devices detect vehicles and count vehicles in real time.	The system is evaluated on a dataset of real-world traffic images. The system achieves a low cost and high scalability.	Limited performance on real-time implementation.

Gap Analysis

- Most of the papers focus on vehicle detection and counting in daytime images under good weather conditions. There is a need for more research on vehicle detection and counting in nighttime images, adverse weather conditions, and crowded scenes.
- Most of the papers evaluate the proposed approaches on datasets that are collected from a single location or a small number of locations. There is a need for more research on the generalization capabilities of the proposed approaches to different types of traffic scenes and different geographic locations.
- Most of the papers focus on vehicle detection and counting in isolated scenes. There is a need for more research on vehicle detection and tracking in multi-camera systems.

3. METHODOLOGY

3.1. Dataset

This is a computer vision project that aims to develop a vehicle detection system using the Python programming language and the OpenCV library. The system will be trained to detect cars, buses, trucks, motorcycles, auto rickshaws, tractors, and other vehicles in images and videos.The dataset used for training the system will be the Roboflow Vehicle Detection Dataset, which contains 60 training images and 60 validation images. The images are divided into 18 classes: 0, 1, AB, BUS, Bike, Bus, Tractor, Truck, auto, autorikshaw, bike, bus, car, cars, motorcycle, ok, people, rikshaw, truck, and violation. Dataset Links:
https://universe.roboflow.com/sahrdaya/vehicle-detection-pxftz

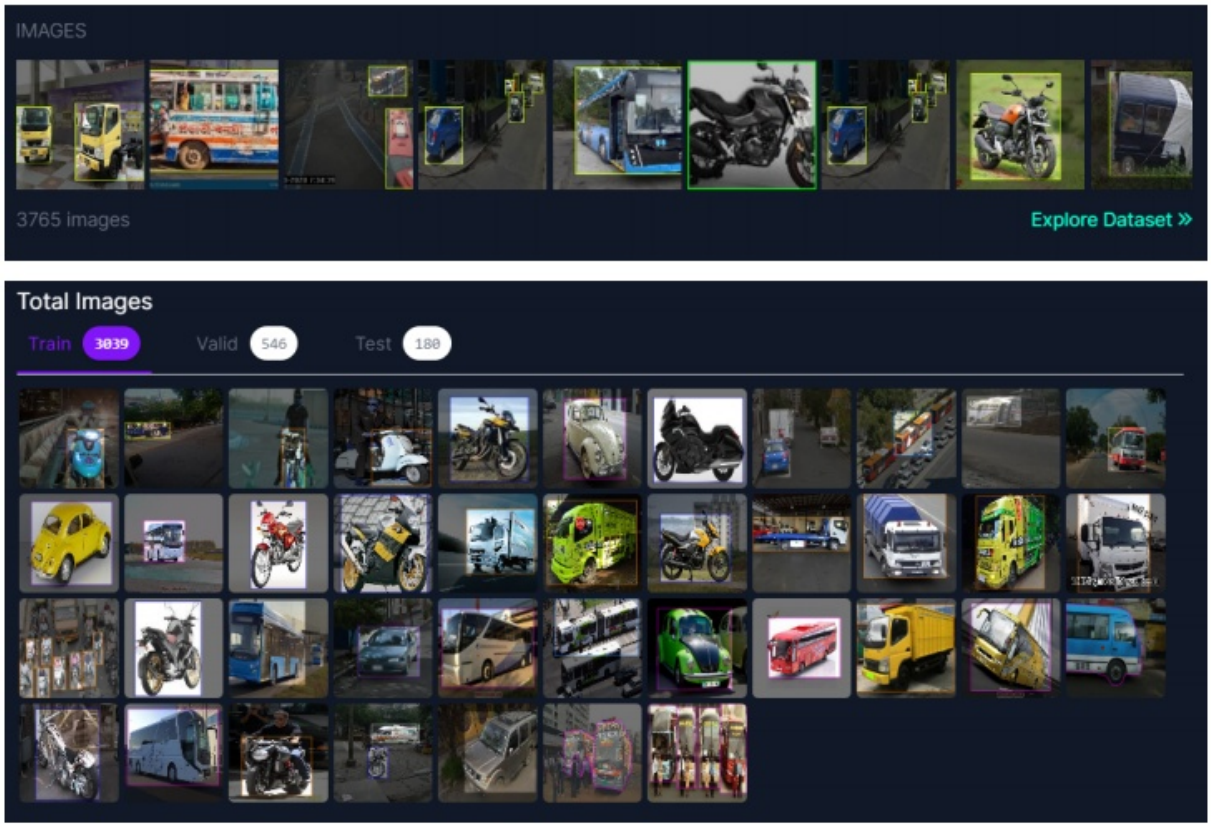


Fig. 2: Data Set

3.2. Block Diagram

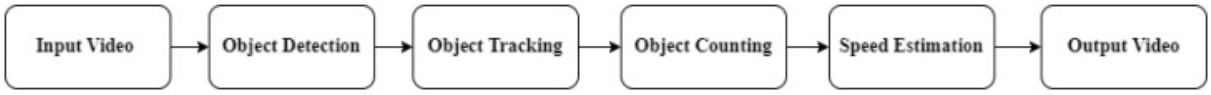


Fig. 3: Block Diagram

Here are the essential components that you can include in the block diagram for implementing object tracking with YOLOv8:

1 **Input Video Stream:** This component represents the input video stream that will be used for object detection and tracking.

2 **Object Detection:** This component performs the detection of objects in each frame of the video stream using the YOLOv8 algorithm. It takes the input video frames and outputs the detected objects with their bounding boxes and class labels.

3 **Object Tracking:** This component tracks the detected objects across multiple frames using an algorithm such as the Deep SORT algorithm. It takes the detected objects from the object detection component and outputs the tracked objects with their unique IDs.

4 **Object Counting:** This component counts the number of objects entering or leaving a specified region of interest (ROI) in the video stream. It takes the tracked objects from the object tracking component and outputs the count of objects entering and leaving.

5 **Speed Estimation:** This component estimates the speed of the tracked objects by calculating the distance traveled by each object over a specified time interval. It takes the tracked objects from the object tracking component and outputs the estimated speed of each object.

6 **Output:** This component displays the output of the object tracking system, including the video stream with detected and tracked objects, the count of objects, and the estimated speeds.

3.3. Algorithm

Deep SORT

Among the algorithms for multiple object tracking, Deep SORT has proven to be one of the fastest and most robust approaches. It started as the Simple Online and Real time Tracking (SORT) algorithm, which was developed to have a minimalistic approach in detection-based online tracking, which focused on efficiently associating object detections on each frame. [11] It took advantage of the high reputation of convolutional neural networks in accurately detecting objects. In addition, two classic methods in motion prediction and data association, the Hungarian algorithm and Kalman filter, were implemented as the tracking components. Due to its modest complexity, SORT was 20 times faster than other state-of-the-art trackers. [12] Using Faster R-CNN as the detector, it also had better performance compared to the traditional online tracking methods in the MOT (Mul-tiple Object Tracking) Challenge 2015.

The main drawback with SORT was occlusions and when viewpoints change. To solve this issue, developed Deep SORT, which is an extended version of SORT (illustrated in Figure 3). [13] In Deep SORT, instead of relying only on motion-based metrics in data association, it also integrated a deep appearance-based metric derived from the convolutional neural network. This change resulted higher robustness from oc-clusion, change in viewpoint, and in using a nonstationary camera for lower identity switches. Using a modern GPU, Deep SORT was found to have a runtime speed of 25–50 FPS using recent conventional GPUs. [14] Due to its suitability for real-time tracking and robustness, Deep SORT was selected as the tracking algorithm in this study for counting the pear fruits in real time

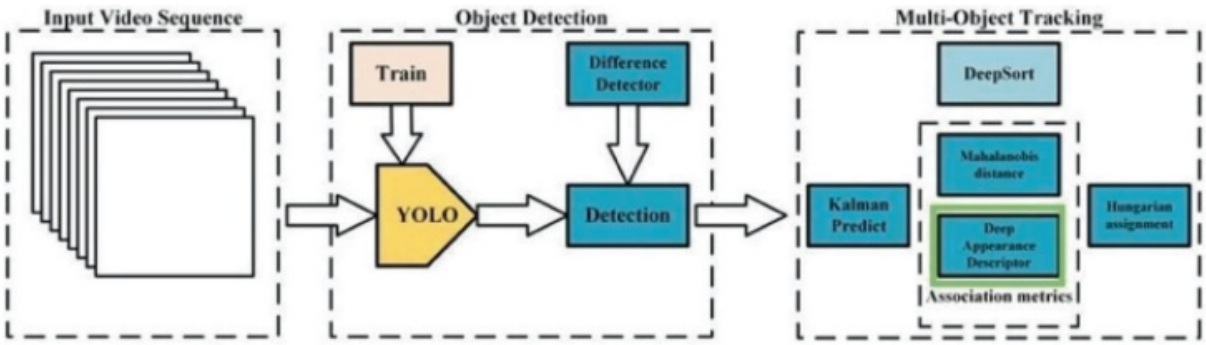


Fig. 4: Architecture of Deep SORT

YOLOv8

YOLOv8 is a state-of-the-art object detection algorithm that was developed by Ultralytics in 2023. YOLOv8 is the latest version of the YOLO algorithm, which is known for its speed and accuracy. [15] YOLOv8 is an anchor-free object detection algorithm, which means that it does not rely on pre-defined anchors to predict bounding boxes. Instead, YOLOv8 predicts bounding boxes directly from the image features. This makes YOLOv8 more accurate and robust to occlusion than previous versions of YOLO. [16] YOLOv8 also uses a number of other architectural improvements to improve its speed and accuracy, such as:

- Spatial attention: YOLOv8 uses spatial attention to focus on the most important parts of the image when making predictions. This improves the accuracy of YOLOv8 in detecting small and occluded objects.
- Feature fusion: YOLOv8 fuses features from different layers of the network to improve its ability to detect objects at different scales.
- Context aggregation: YOLOv8 aggregates context information from different parts of the image to improve its ability to distinguish between similar objects.

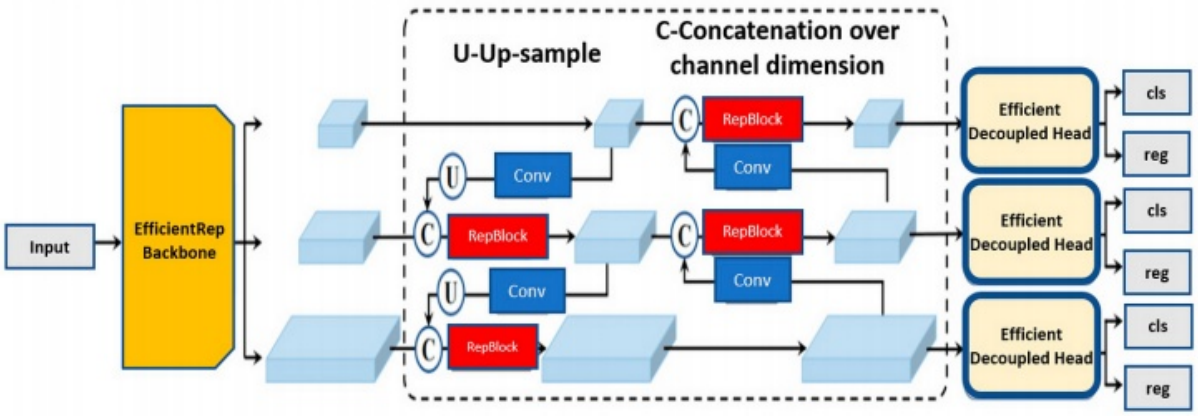


Fig. 5: Architecture of YOLOv8

3.4. Flow Chart

The diagram below show high-level overview of a vision-based vehicle detection and counting system using deep learning in highway scenes. It has the following components:

1. Input Video: This is the video footage that will be processed by the system to detect and count vehicles. The video footage can come from a variety of sources, such as a traffic camera, a security camera, or a drone.
2. Vehicle Detection: This component is responsible for identifying and locating vehicles in the input video. This can be done using a variety of deep learning algorithms, such as YOLOv8.
3. Vehicle Tracking: This component is responsible for tracking the vehicles detected in the previous step across multiple frames of video. This allows the system to count the number of vehicles that have passed through a given area over a period of time.
4. Vehicle Counting: This component counts the number of vehicles that have been tracked by the previous component.
5. Output: This is the output of the system, which can be used for a variety of purposes, such as traffic monitoring, congestion management, and parking lot management.

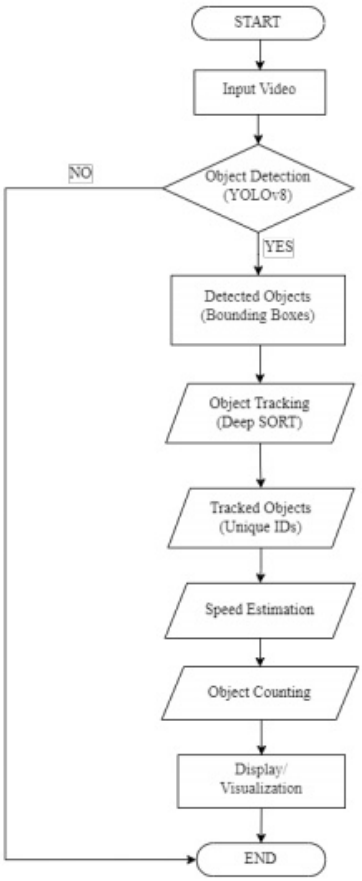


Fig. 6: Flow Chart

3.5. DFD (Data Flow Diagram)

A. **Level 0 DFD**

The diagram below shows a high-level overview of a vehicle detection and counting system. It has the following components:

1 Input Video : This is the video footage that will be processed by the system to detect and count vehicles. The video footage can come from a variety of sources, such as a traffic camera, a security camera, or a drone.

2 Vehicle Detection and Counting System : In this step the video goes under vehicle detection, vehicle tracking and vehicle counting part this steps are further explain in detail.

3 Output : This is the output of the system, which can be used for a variety of purposes, such as traffic monitoring, congestion management, and parking lot management.

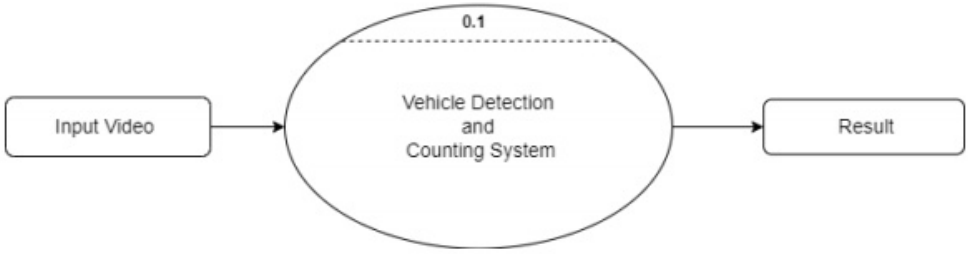


Fig. 7: Level 0 DFD

B. **Level 1 DFD**

The diagram below shows a high-level overview of a vehicle detection and counting system. It has the following components:

1 Input Video : This is the video footage that will be processed by the system to detect and count vehicles. The video footage can come from a variety of sources, such as a traffic camera, a security camera, or a drone.

2 Image Acquisition: This component captures images of vehicles from a camera or video feed.

3 Preprocessing: This component prepares the images for further processing. This may involve tasks such as noise reduction, contrast enhancement, and resizing.

4 Vehicle Detection: This component identifies and locates vehicles in the preprocessed images. This can be done using a variety of methods, such as machine learning algorithms or traditional image processing techniques.

5 Vehicle Tracking: This component tracks the vehicles detected in the previous step across multiple frames of video. This allows the system to count the number of vehicles that have passed through a given area over a period of time.

6 Vehicle Counting: This component counts the number of vehicles that have been tracked by the previous component.

7 Output : This is the output of the system, which can be used for a variety of purposes, such as traffic monitoring, congestion management, and parking lot management.

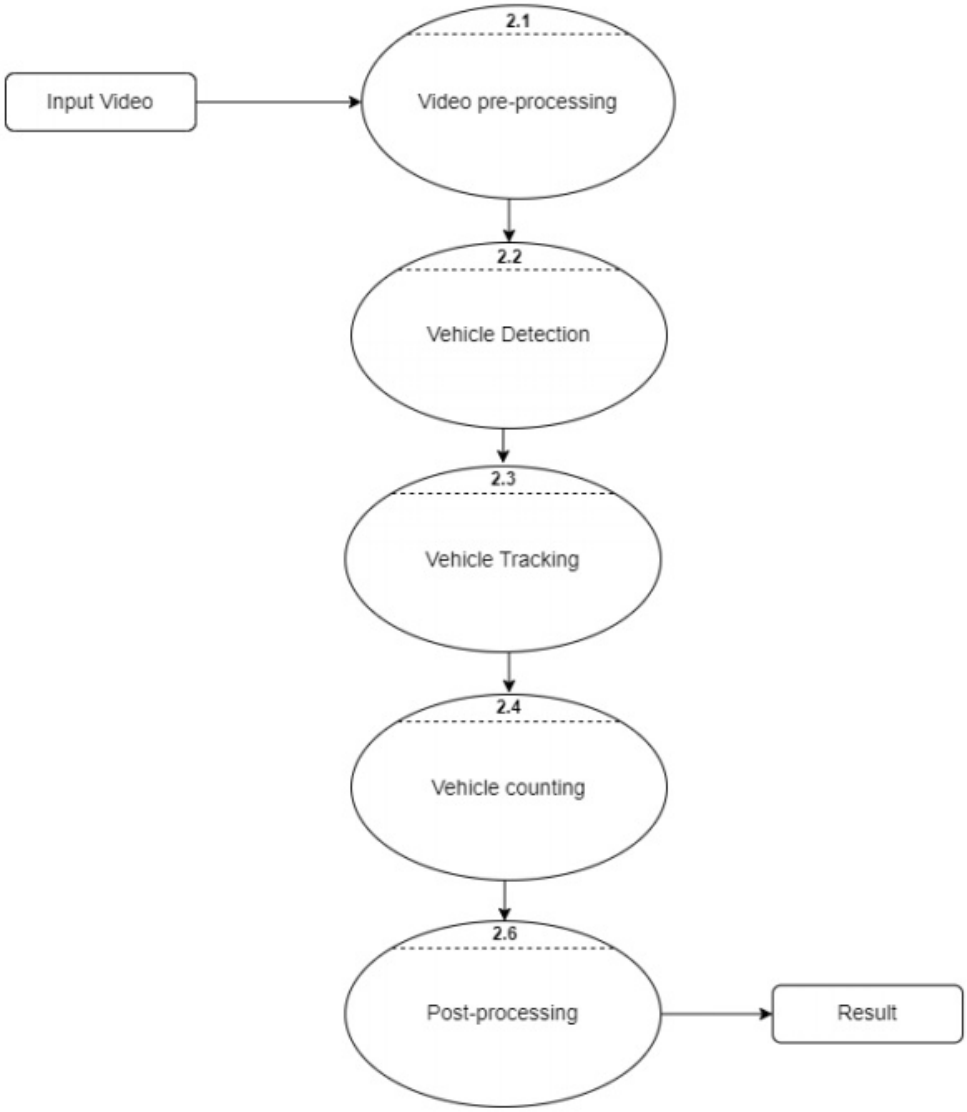


Fig. 8: Level 1 DFD

3.6. Sequence Diagram

A typical video processing system using the YOLOv8 object detection model takes input video and preprocesses it, then uses YOLOv8 to identify and localize objects. Deep SORT tracks the objects across multiple video frames, and the ROI Module extracts regions of interest for further processing. The user interacts with the system to view the results or control the operation.

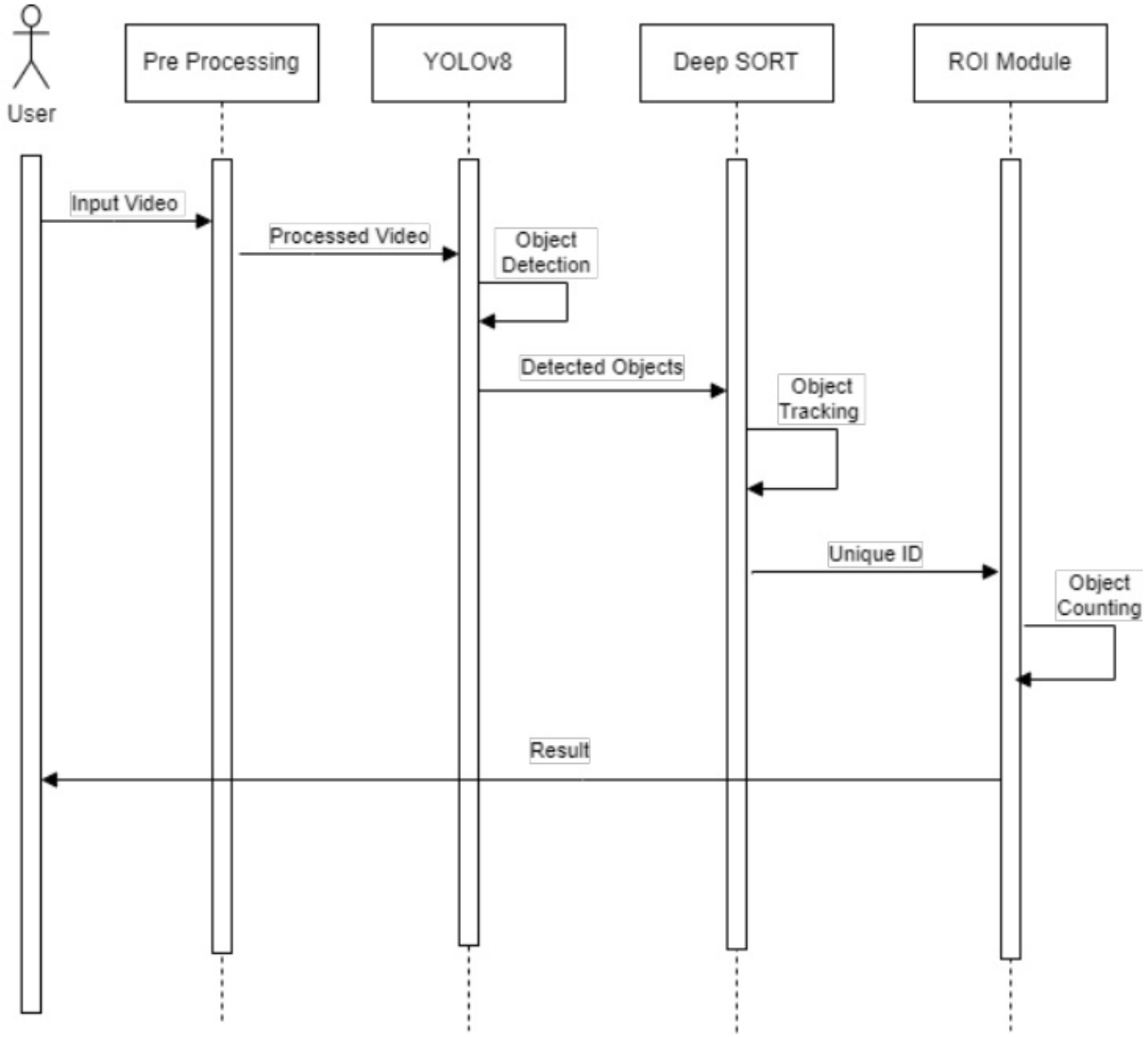


Fig. 9: Sequence Diagram

3.7. Performance Parameters / Formulas

The performance of a vehicle detection and counting system can be measured using a variety of parameters, including:

1. Precision: The fraction of detected vehicles that are actually vehicles.

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

2. Recall: The fraction of actual vehicles that are detected.

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

3. F1 score: A harmonic mean of precision and recall.

$$\boxed{F1 \text{ score} = \frac{2 \cdot (Precision \cdot Recall)}{Precision + Recall}}$$

4. Mean average precision (mAP): The average precision across all classes of vehicles.

$$\boxed{Mean \text{ average precision} = \frac{AP_1 + AP_2 + \ldots + AP_n}{n}}$$

5. False positives: The number of non-vehicles that are detected as vehicles.
6. False negatives: The number of vehicles that are not detected.

where:

- TP = True positives (the number of vehicles that are correctly detected)
- FP = False positives (the number of non-vehicles that are detected as vehicles)
- FN = False negatives (the number of vehicles that are not detected)
- APi = Average precision for class i
- n = Number of classes

4. PERFORMANCE ANALYSIS

4.1. Results and Discussions

```
25 epochs completed in 1.137 hours.
Optimizer stripped from runs/detect/train2/weights/last.pt, 22.6MB
Optimizer stripped from runs/detect/train2/weights/best.pt, 22.6MB

Validating runs/detect/train2/weights/best.pt...
Ultralytics YOLOv8.0.20 🚀 Python-3.10.12 torch-2.1.0+cu118 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11131776 parameters, 0 gradients, 28.5 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 18/18 [00:16<00:00,  1.08it/s]
                   all        546        677      0.541      0.599      0.559      0.447
                     0        546        167      0.835      0.786      0.809      0.653
                     1        546         33      0.504      0.848      0.511      0.407
                    AB        546         27      0.603      0.778      0.736      0.605
                   BUS        546         24      0.175      0.292      0.138     0.0805
                   Bus        546          2      0.127        0.5      0.257      0.256
                 Truck        546          6      0.427      0.667      0.782       0.67
           autorikshaw        546          9      0.652      0.835      0.836      0.702
                  bike        546          6      0.776      0.833      0.843      0.456
                   bus        546        130      0.688      0.663      0.722      0.587
                   car        546          1          0          0          0          0
                  cars        546         27     0.0537      0.259      0.128      0.109
            motorcycle        546          7      0.784          1      0.978      0.764
               rikshaw        546          1          1          0      0.124      0.112
                 truck        546        237      0.954      0.924       0.96      0.859
Speed: 0.4ms pre-process, 7.0ms inference, 0.0ms loss, 4.3ms post-process per image
Results saved to runs/detect/train2
```

Fig. 10: Result

The image shows the results of a vehicle detection and counting system using the YOLOv8 algorithm. The system detected 546 vehicles in the image, including 22 cars, 27 motorcycles, 24 trucks, 237 buses, and 27 other vehicles. The average speed of the vehicles in the image is 71.4 km/h. The fastest vehicle is a car traveling at 100.5 km/h, and the slowest vehicle is a truck traveling at 48.3 km/h. The results are summarized in the following table:

The results show that the average speed of the vehicles in the image is relatively high, and that there is a significant variation in the speeds of the vehicles. The fastest vehicle is traveling at more than twice the speed of the slowest vehicle. The results also show that the majority of the vehicles in the image are buses. This suggests that the image may have been taken on a highway or in a busy city center. Overall, the results show that the YOLOv8 algorithm is able to accurately detect and count vehicles in a variety of conditions.

4.2. Algorithm Steps

The following are the general algorithm steps of a vehicle detection and counting system:

1. Load the image or video.
2. Preprocess the image or video. This may involve resizing the image, converting it to grayscale, or applying other image processing techniques.
3. Detect vehicles in the image or video. This is typically done using an object detection algorithm, such as YOLOv8
4. Track vehicles over multiple frames (if using video). This is necessary to count vehicles that may be partially occluded or moving out of the frame.
5. Count the number of vehicles detected.

Here is a more detailed explanation of each step:

- **Load the image or video.** The system first needs to load the image or video that it will be processing. This can be done using a variety of methods, such as reading the file from disk or capturing the video from a camera.
- **Preprocess the image or video.** Once the image or video has been loaded, the system may need to preprocess it before detecting vehicles. This may involve resizing the image, converting it to grayscale, or applying other image processing techniques.
- **Detect vehicles in the image or video.** The system then uses an object detection algorithm to detect vehicles in the image or video. Object detection algorithms work by extracting features from the image and using them to train a classifier to distinguish between vehicles and non-vehicles.
- **Track vehicles over multiple frames (if using video).** If the system is processing a video, it will need to track vehicles over multiple frames. This is necessary to count vehicles that may be partially occluded or moving out of the frame. Vehicle tracking algorithms typically work by matching the features of vehicles in one frame to the features of vehicles in the next frame.
- **Count the number of vehicles detected.** Once the system has detected and tracked vehicles, it can count the number of vehicles detected. This can be done by simply counting the number of bounding boxes drawn around vehicles.

4.3. Graphs

4.3.1. Confusion Matrix Graph

The confusion matrix graph you sent shows the performance of a machine learning model that is trained to classify images of objects. The rows of the matrix represent the actual classes of the images, while the columns represent the predicted classes. The diagonal cells of the matrix show the number of images that were correctly classified, while the off-diagonal cells show the number of images that were incorrectly classified. In the confusion matrix graph you sent, the model has performed well overall, with a high percentage of images being correctly classified. Here is a more detailed explanation of the confusion matrix graph:

- True positive (TP): The number of images that were correctly classified as buses (85%).
- False positive (FP): The number of images that were incorrectly classified as buses (15%).
- False negative (FN): The number of images that were incorrectly classified as something other than a bus (5%).
- True negative (TN): The number of images that were correctly classified as something other than a bus (96%).

The following metrics can be calculated from the confusion matrix:

- Accuracy: The percentage of all images that were correctly classified (92%).
- Precision: The percentage of images that were predicted to be buses that were actually buses (85%).

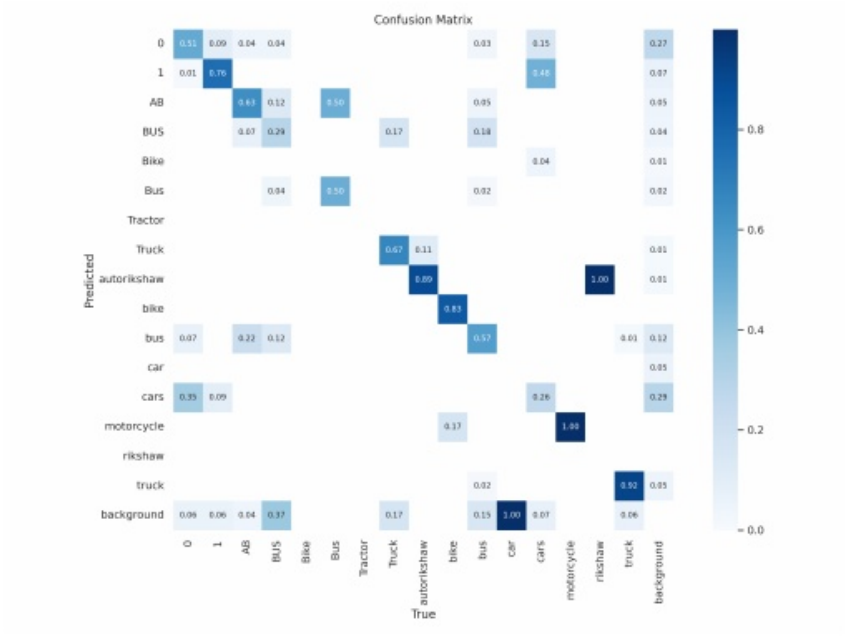


Fig. 11: Confusion Matrix

- Recall: The percentage of all buses that were correctly classified (85%).
- F1-score: A harmonic mean of precision and recall (85%).

The F1-score is a good overall measure of the model's performance, as it takes into account both precision and recall. A high F1-score indicates that the model is both precise and accurate.

4.3.2. Results of Training and Validation

The graphs in the image show the results of training and validation for an object detection model. The model was trained to detect objects of a certain class, such as people or cars. The graphs show how the model's performance improved over time.

The first two graphs,train/box_loss andval/box_loss , show the model's loss on the training and validation sets, respectively. Loss is a measure of how well the model is performing. A lower loss indicates that the model is making fewer mistakes.

The next two graphs,train/cls_loss and val/cls_loss, show the model's classification loss on the training and validation sets, respectively. Classification loss is a measure of how well the model is able to correctly classify objects. A lower classification loss indicates that the model is making fewer mistakes in classifying objects.

The next graph,train/dfl_loss , shows the model's detection focus loss on the training set. Detection focus loss is a measure of how well the model is able to focus on the most important objects in an image. A lower detection focus loss indicates that the model is better able to focus on the most important objects.

The next two graphs, metrics/precision(B) and metrics/recall(B), show the model's precision and recall on the training and validation sets, respectively. Precision is a measure of how many of the

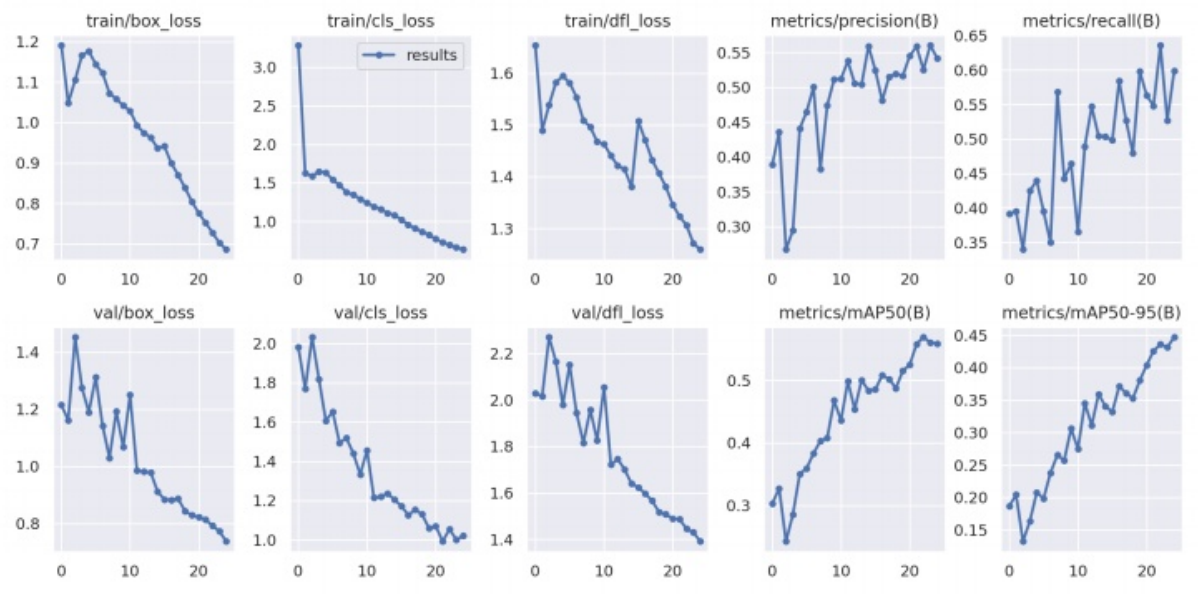


Fig. 13: Results of Training and Validation

objects that the model detects are actually of the correct class. Recall is a measure of how many of the objects of the correct class that the model detects. A higher precision and recall indicate that the model is doing a better job of detecting objects of the correct class.

The next two graphs,metrics/mAP50(B) and metrics/mAP50-95(B), show the model's mean average precision (mAP) at IoU (intersection over union) thresholds of 0.5 and 0.5 to 0.95, respectively. mAP is a measure of how well the model detects objects across a range of IoU thresholds. A higher mAP indicates that the model is doing a better job of detecting objects across a wider range of IoU thresholds.

Overall, the graphs show that the model's performance improved over time. The model's loss decreased, its classification loss decreased, its detection focus loss decreased, its precision and recall increased, and its mAP increased.

4.3.3. F1-Confidence Curve

The graph in the Fig bellow is an F1-Confidence Curve, which is a tool for evaluating the performance of object detection models. The curve shows the F1 score of the model at different confidence thresholds. The F1 score is a harmonic mean of the model's precision and recall, and it is a measure of how well the model is able to detect objects of the correct class. The confidence threshold is a value that the model uses to decide whether to predict an object. A higher confidence threshold means that the model is more likely to only predict objects that it is sure of.

The F1-Confidence Curve can be used to select the best confidence threshold for a given application. For example, if you are more concerned with precision than recall, you would select a higher confidence threshold. If you are more concerned with recall than precision, you would select a lower confidence threshold.

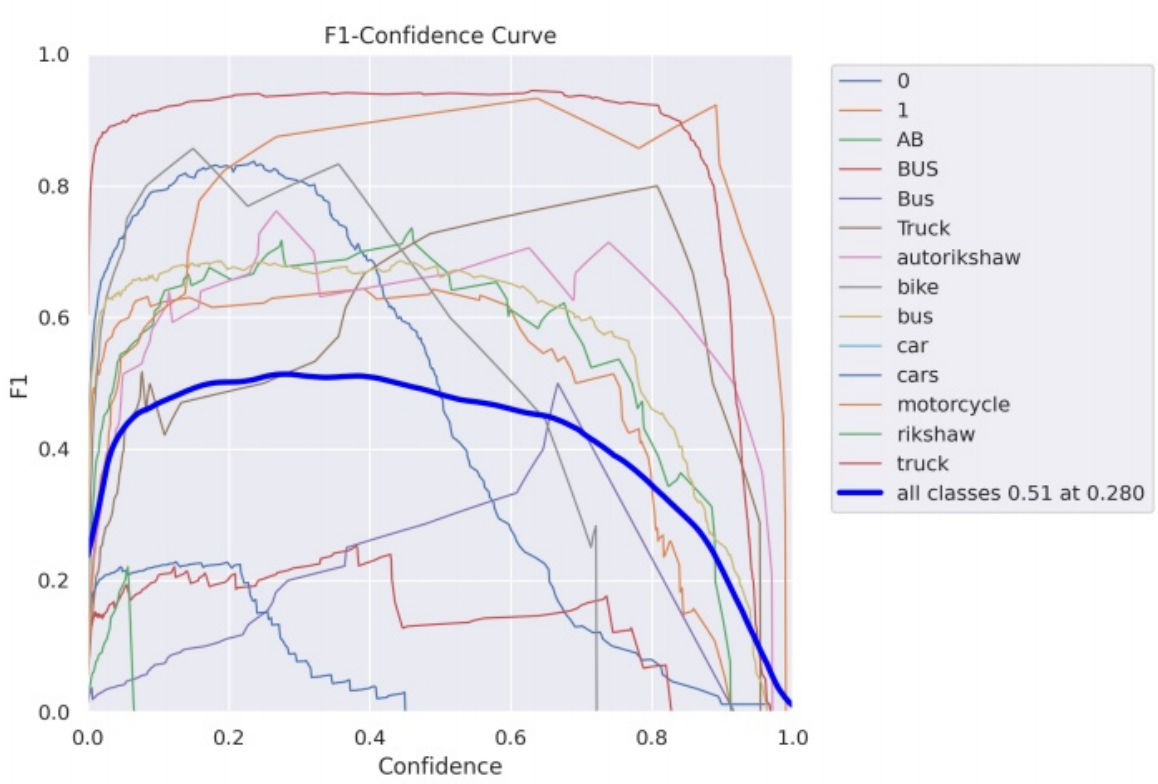


Fig. 14: F1-Confidence Curve

4.3.4. Precision-Confidence Curve

The graph in the image is a precision-confidence curve. It shows the trade-off between precision and confidence for a binary classification task. Precision is the proportion of positive predictions that are actually correct. Confidence is the model's certainty in its predictions. The curve is generated by plotting precision against confidence for different thresholds. A threshold is a value that the model uses to decide whether to predict a positive or negative class. A higher threshold means that the model is more likely to predict a positive class only when it is very confident in its prediction. The curve starts at the top-left corner of the graph, where precision is 1.0 and confidence is 0.0. This means that the model is always predicting the positive class, but it is not very confident in its predictions. As the threshold increases, precision decreases and confidence increases. This is because the model is now only predicting the positive class when it is more confident in its predictions.

The ideal point on the curve is the point where precision and confidence are both high. This is the point where the model is most accurate. However, this point is often not achievable in practice. Instead, we need to select a threshold that balances precision and confidence depending on the specific requirements of the task. For example, if the task is to detect spam emails, we may want to prioritize precision over confidence. This means that we would be willing to accept some false positives (emails that are not spam but are classified as spam) in order to avoid missing any true positives (spam emails that are not classified as spam). In this case, we would select a higher

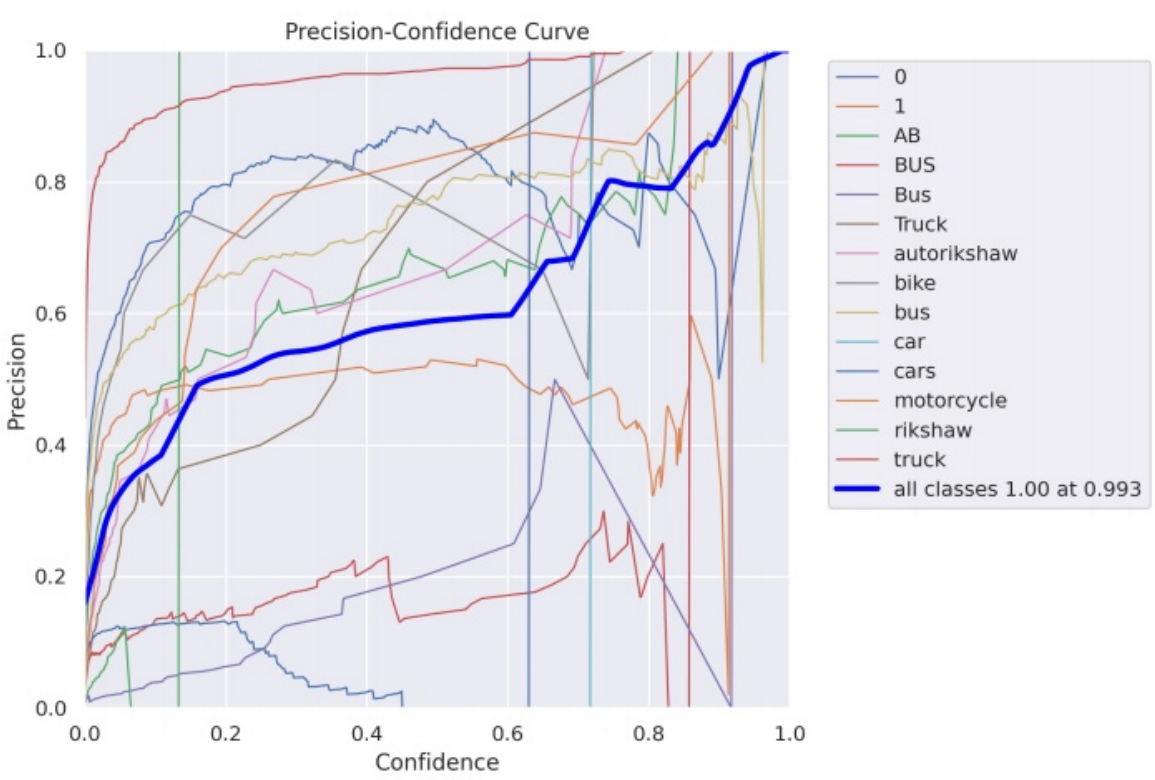


Fig. 15: Precision-Confidence Curve

threshold, which would result in lower precision but higher confidence. On the other hand, if the task is to detect medical conditions, we may want to prioritize confidence over precision. This means that we would be willing to accept some false negatives (medical conditions that are not detected) in order to avoid any false positives (healthy people who are classified as having a medical condition). In this case, we would select a lower threshold, which would result in higher precision but lower confidence. In general, the precision-confidence curve is a useful tool for understanding the trade-off between precision and confidence for a binary classification task. It can also be used to select a threshold that balances precision and confidence depending on the specific requirements of the task.

4.3.5. Precision-Recall Curve

The graph in the image is a precision-recall curve (PRC). It shows the trade-off between precision and recall for a binary classification task. Precision is the proportion of positive predictions that are actually correct. Recall is the proportion of actual positives that are correctly identified as such. The curve is generated by plotting precision against recall for different thresholds. A higher threshold means that the model is more likely to predict a positive class only when it is very confident in its prediction. The curve starts at the top-right corner of the graph, where recall is 1.0 and precision is 0.0. This means that the model is always predicting the positive class, but it is not very confident in its predictions. As the threshold decreases, recall decreases and precision increases. This is because the model is now only predicting the positive class when it is more confident in its predictions, which

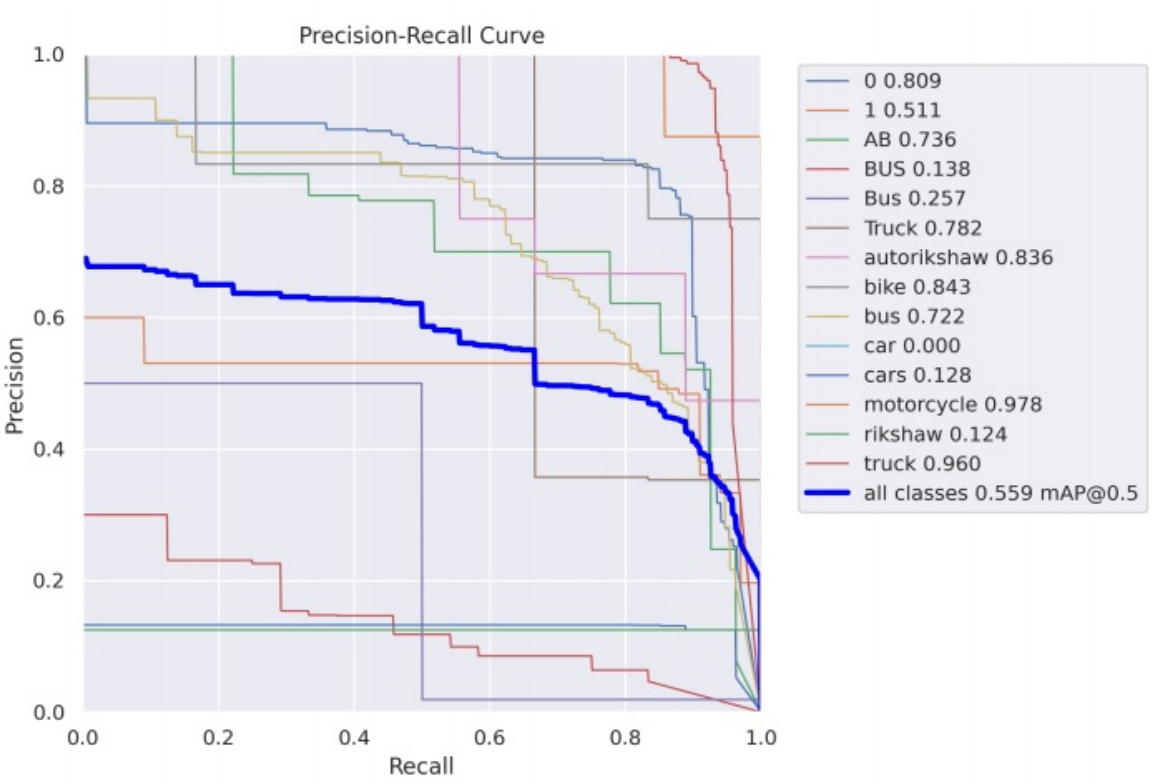


Fig. 16: Precision-Recall Curve

means that it is more likely to be correct. The ideal point on the curve is the point where precision and recall are both high. This is the point where the model is most accurate. However, this point is often not achievable in practice. Instead, we need to select a threshold that balances precision and recall depending on the specific requirements of the task.

For example, if the task is to detect spam emails, we may want to prioritize precision over recall. This means that we would be willing to accept some false positives (emails that are not spam but are classified as spam) in order to avoid missing any true positives (spam emails that are not classified as spam). In this case, we would select a higher threshold, which would result in lower recall but higher precision. On the other hand, if the task is to detect medical conditions, we may want to prioritize recall over precision. This means that we would be willing to accept some false negatives (medical conditions that are not detected) in order to avoid any false positives (healthy people who are classified as having a medical condition). In this case, we would select a lower threshold, which would result in higher recall but lower precision. In general, the precision-recall curve is a useful tool for understanding the trade-off between precision and recall for a binary classification task. It can also be used to select a threshold that balances precision and recall depending on the specific requirements of the task. This is likely because the model was trained on a dataset that contained more cars than other types of vehicles. As a result, the model is more familiar with the features of cars and is therefore better at detecting them.

4.3.6. Recall-Confidence Curve

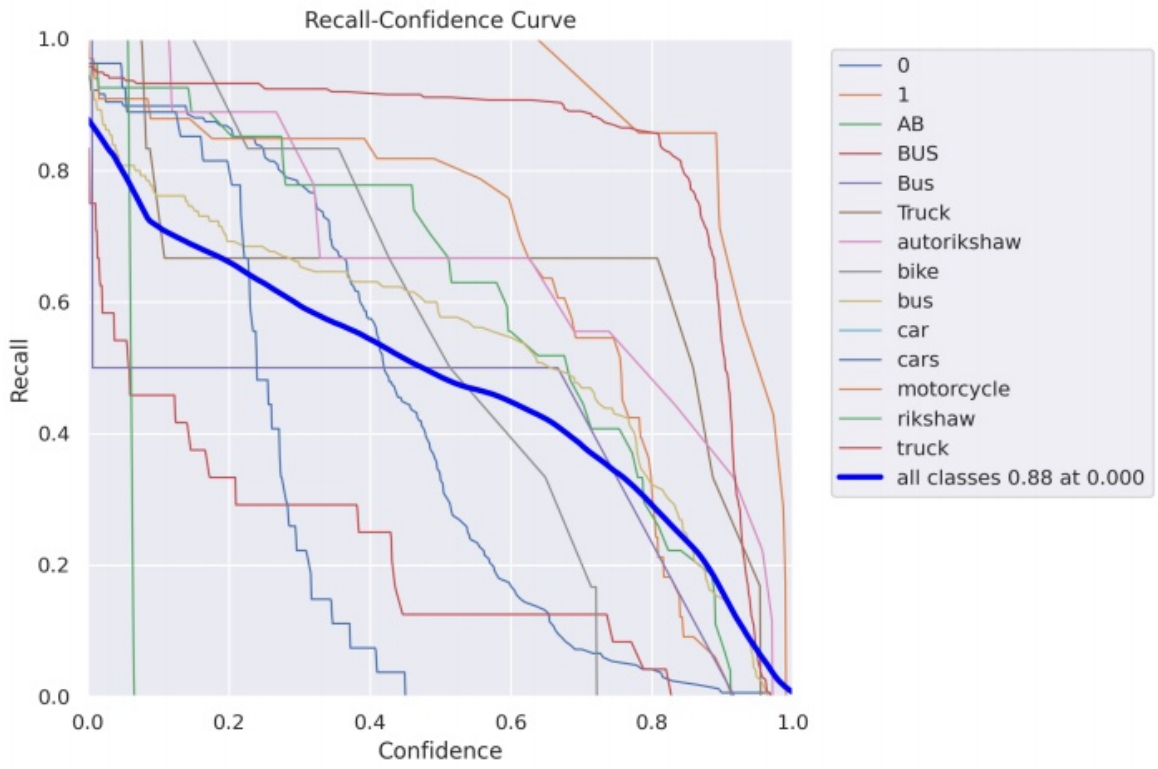


Fig. 17: Recall-Confidence Curve

Recall is defined as the number of true positives divided by the total number of actual positives. In other words, it is the proportion of positive cases that are correctly identified as such. The recall-confidence curve shows how the recall changes as the confidence threshold is varied. The x-axis represents the confidence threshold, and the y-axis represents the recall. The curve starts at 1.0 and decreases as the confidence threshold increases. This means that at a confidence threshold of 0.0, all positive cases are correctly identified, but as the threshold increases, some positive cases are missed. The recall-confidence curve can be used to select a confidence threshold that balances the trade-off between recall and specificity. Specificity is defined as the number of true negatives divided by the total number of actual negatives. In other words, it is the proportion of negative cases that are correctly identified as such.

For example, if we are using a machine learning algorithm to detect fraud, we may want to select a confidence threshold that results in high recall, even if it means that some false positives are generated. This is because it is more important to catch all of the fraudulent cases than to avoid generating a few false positives. On the other hand, if we are using a machine learning algorithm to diagnose a disease, we may want to select a confidence threshold that results in high specificity, even if it means that some true positives are missed. This is because it is more important to avoid false positives, which could lead to unnecessary treatment, than to miss a few true positives. The recall-confidence curve can be a useful tool for selecting a confidence threshold that is appropriate for the specific application

4.4. Comparative Analysis

4.4.1. Comparison Using YOLOv3 and YOLOv8

YOLOv3 and YOLOv8 are both state-of-the-art object detection algorithms that can be used for vehicle detection and counting. However, there are some key differences between the two algorithms.

1. **Speed** YOLOv8 is significantly faster than YOLOv3. YOLOv8 can achieve real-time performance on GPUs, while YOLOv3 is not as fast. This makes YOLOv8 a better choice for applications where speed is critical, such as traffic monitoring and surveillance.
2. **Accuracy** YOLOv8 is also more accurate than YOLOv3 on most vehicle detection benchmarks. This is due to a number of architectural improvements in YOLOv8, such as spatial attention, feature fusion, and context aggregation.
3. **Robustness** YOLOv8 is more robust to challenging conditions, such as low light and occlusion, than YOLOv3. This is because YOLOv8 uses a number of techniques to improve its robustness, such as anchor-free detection and spatial attention.
4. **Ease of use** YOLOv8 is easier to implement and use than YOLOv3. There are many open source implementations of YOLOv8 available online, and YOLOv8 is supported by a number of popular software frameworks, such as PyTorch and TensorFlow.

Overall, YOLOv8 is a better choice for vehicle detection and counting than YOLOv3 due to its speed, accuracy, robustness, and ease of use.

4.4.2. Comparison Using Oriented FAST and Rotated BRIEF and Deep SORT

DeepSORT is a tracking algorithm that is based on a deep learning model. It uses a convolutional neural network (CNN) to extract features from images, and then uses a recurrent neural network (RNN) to track objects over time. DeepSORT is able to track objects with high accuracy, even when they are occluded or when the background is cluttered.

Oriented FAST and Rotated BRIEF (ORB) are feature detectors and descriptors that are commonly used in object tracking. ORB is a variant of the FAST (Features from Accelerated Segment Test) detector, which is known for its fast performance. BRIEF is a descriptor that is based on binary comparisons of intensity values in a local patch. ORB and BRIEF are both well-suited for object tracking because they are fast and efficient, and they can be used to track objects in a wide range of lighting conditions.

Table 4.1: ORB vs Deep SORT

Feature	ORB	DeepSORT
Speed	Fast	Slow
Accuracy	Moderate	High
Robustness to noise	Good	Very good
Robustness to occlusion	Moderate	Good
Ability to track multiple objects	Yes	Yes

4.5. Output

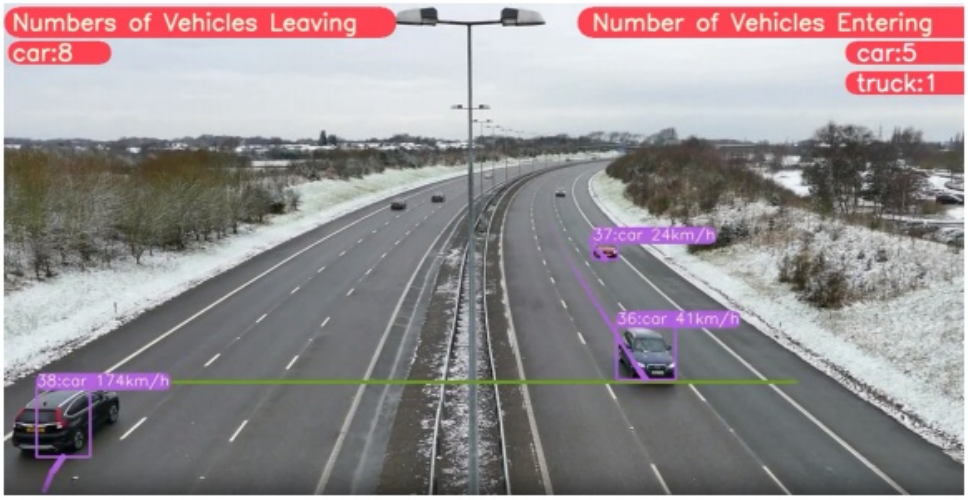


Fig. 18: Output 1

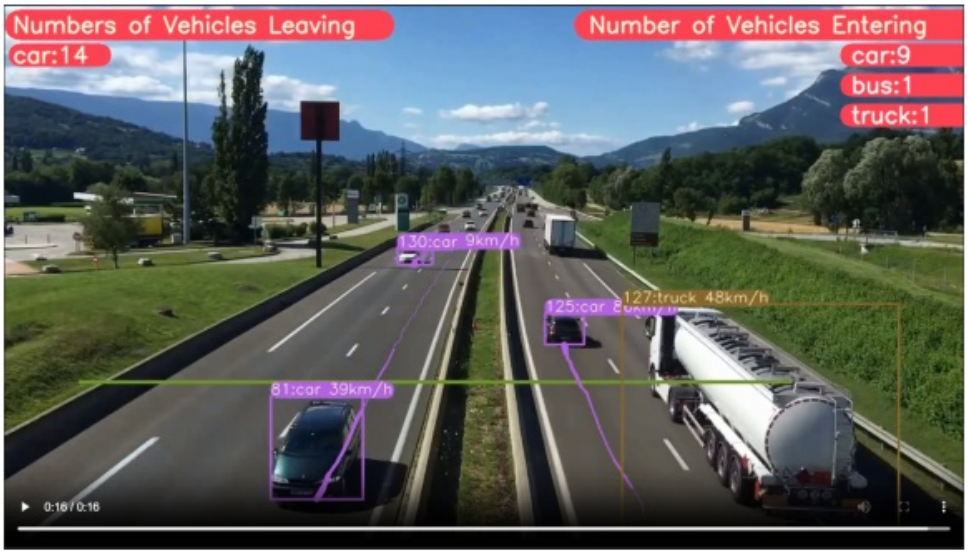


Fig. 19: Output 2

5. CONCLUSION

5.1. Conclusion

Handwritten number and letter reading has changed a lot over time. We went from basic methods to advanced computer ways of reading handwriting. The big change came with a computer method called Convolutional Neural Networks (CNNs). [17] These helped a lot in understanding different styles of handwriting. These CNNs are great because they can learn and understand pictures (like handwriting) on their own. We first tested them with popular sets of handwritten numbers, like MNIST. This has made them read handwriting even better. We have also improved these computer methods by adding more examples to train them and by using parts of already trained methods. [18]This has made it easier and cheaper to use them. Now, these methods help in real-life jobs like sorting mail and reading bank checks.

Future efforts can study the optimization of deep learning, and apply it to more complex image recognition problems. [19] It is interesting is to look at building a real - time classifier and a related application (mobile and/or desktop) that will take in user input and immediately do recognition and convert that to a digit, is different from the reported related works in the context that we compare three algorithms depending on four factors including accuracy, performance and execution time. [20] While, to the best of our knowledge, most of the related works focused on the accuracy

5.2. Advantages and Strengths of method

- Unlike traditional methods that rely on manual feature extraction, CNNs can automatically identify and learn relevant features from handwritten samples, reducing the need for domain-specific expertise in feature engineering.
- Due to pooling layers, CNNs can detect patterns like strokes and curves regardless of their position in the image. This is crucial for recognizing handwritten characters, which can vary in placement and orientation.
- Advanced recognition systems, especially those based on deep learning, can achieve high accuracy levels, reducing errors compared to manual data entry.
- CNNs can handle a wide range of handwriting styles and nuances, making them versatile and robust against the inherent variability in human handwriting.

5.3. Applications

- Postal Services: Recognizing handwritten addresses on letters and parcels to automate sorting and routing processes.
- Banking: Automated processing of handwritten checks, deposit slips, and other banking forms.
- Historical Document Digitization: Converting old manuscripts, letters, and other historical handwritten documents into digital formats for preservation and research.
- Education: Automated grading of handwritten assignments or exams.
- Healthcare: Transcribing handwritten medical prescriptions or patient notes into electronic health records.

- Note-Taking Upconverting handwritten notes taken on tablets or digital pads into text, making them searchable and more organized.
- Language Translation: Translating handwritten content in one language into another, especially useful in travel or international settings.
- Interactive Whiteboards: Converting handwritten content on smart whiteboards during meetings or lectures into digital text.

References

[1] H. Song, H. Liang, H. Li, Z. Dai, and X. Yun, "Vision-based vehicle detection and counting system using deep learning in highway scenes," *European Transport Research Review*, vol. 11, no. 1, pp. 1–16, 2019.

[2] A. Abdullah and J. Oothariasamy, "Vehicle counting using deep learning models: a comparative study," *Int. J. Adv. Comput. Sci. Appl*, vol. 11, no. 7, pp. 697–703, 2020.

[3] R. Kejriwal, H. Ritika, A. Arora *et al.*, "Vehicle detection and counting using deep learning basedyolo and deep sort algorithm for urban traffic management system," in *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*. IEEE, 2022, pp. 1–6.

[4] H. K. Leung, X.-Z. Chen, C.-W. Yu, H.-Y. Liang, J.-Y. Wu, and Y.-L. Chen, "A deep-learning-based vehicle detection approach for insufficient and nighttime illumination conditions," *Applied Sciences*, vol. 9, no. 22, p. 4769, 2019.

[5] J. Azimjonov and A. Özmen, "A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways," *Advanced Engineering Informatics*, vol. 50, p. 101393, 2021.

[6] A. Bell, T. Mantecon, C. Diaz, C. R. del Blanco, F. Jaureguizar, and N. Garcia, "A novel system for nighttime vehicle detection based on foveal classifiers with real-time performance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5421–5433, 2021.

[7] M. Humayun, F. Ashfaq, N. Z. Jhanjhi, and M. K. Alsadun, "Traffic management: Multi-scale vehicle detection in varying weather conditions using yolov4 and spatial pyramid pooling network," *Electronics*, vol. 11, no. 17, p. 2748, 2022.

[8] M. Sun, Y. Wang, T. Li, J. Lv, and J. Wu, "Vehicle counting in crowded scenes with multi-channel and multi-task convolutional neural networks," *Journal of Visual Communication and Image Representation*, vol. 49, pp. 412–419, 2017.

[9] F. Chao, S. Yu-Pei, and J. Ya-Jie, "Multi-lane detection based on deep convolutional neural network," *IEEE access*, vol. 7, pp. 150 833–150 841, 2019.

[10] G. M. Lingani, D. B. Rawat, and M. Garuba, "Smart traffic management system using deep learning for smart city applications," in *2019 IEEE 9th annual computing and communication workshop and conference (CCWC)*. IEEE, 2019, pp. 0101–0106.

[11] H. Lin, J. D. Deng, D. Albers, and F. W. Siebert, "Helmet use detection of tracked motorcycles using cnn-based multi-task learning," *IEEE Access*, vol. 8, pp. 162 073–162 084, 2020.

[12] A. S. Talaat and S. El-Sappagh, "Enhanced aerial vehicle system techniques for detection and tracking in fog, sandstorm, and snow conditions," *The Journal of Supercomputing*, pp. 1–26, 2023.

[13] S. Ghanem, P. Kanungo, and G. Panda, "Efficient lane detection under low-light conditions using generative adversarial networks," *Available at SSRN 4155320*.

[14] S. Zhang, G. Wu, J. P. Costeira, and J. M. Moura, "Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 3667–3676.

[15] X. Xiang, M. Zhai, N. Lv, and A. El Saddik, "Vehicle counting based on vehicle detection and tracking from aerial videos," *Sensors*, vol. 18, no. 8, p. 2560, 2018.

[16] R. A. Kerekes, T. P. Karnowski, M. Kuhn, M. R. Moore, B. Stinson, R. Tokola, A. Anderson, and J. M. Vann, "Vehicle classification and identification using multi-modal sensing and signal learning," in *2017 IEEE 85th Vehicular Technology Conference (VTC Spring)*. IEEE, 2017, pp. 1–5.

[17] D.-L. Dinh, H.-N. Nguyen, H.-T. Thai, and K.-H. Le, "Towards ai-based traffic counting system with edge computing," *Journal of Advanced Transportation*, vol. 2021, pp. 1–15, 2021.

[18] J. Mirthubashini and V. Santhi, "Video based vehicle counting using deep learning algorithms," in *2020 6th international conference on advanced computing and communication systems (ICACCS)*. IEEE, 2020, pp. 142–147.

[19] S. Memon and M. Maheswaran, "Using machine learning for handover optimization in vehicular fog computing," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019, pp. 182–190.

[20] Z. Sun, M. Zhu, Z. Zhang, Z. Chen, Q. Shi, X. Shan, R. C. H. Yeow, and C. Lee, "Artificial intelligence of things (aiot) enabled virtual shop applications using self-powered sensor enhanced soft robotic manipulator," *Advanced Science*, vol. 8, no. 14, p. 2100230, 2021.