AI Powered Real Time Vehicle Detection Using YOLOv8

Mohd Zahid[1], Mohd Yasir Azeem[2], Mohd Sultan[3], Omar Badar Shafeeque[4]

[1,2,3]B.Tech Scholar, Computer Science & Engineering, Integral University, Lucknow, INDIA

[4]Assistant Professor, Computer Science & Engineering, Integral University, Lucknow, INDIA

Correspondence should be addressed to Mohd Zahid; zahidstudy122344@gmail.com

Abstract

The sudden increase in the density of vehicles in urban areas has led to the need for advanced Intelligent Transportation Systems (ITS) that can perform high-speed, autonomous surveillance. The aim of this research is to design an efficient real-time vehicle detection system that fills the important gap in the literature related to the trade-off between computational speed and accuracy. Although the current state-of-the-art versions of object detection algorithms, such as YOLOv5, have achieved remarkable performance, there is a considerable gap in the literature related to their performance in high-density traffic conditions with overlapping objects (occlusions) and changing lighting conditions. Most of the current systems are sensitive to "anchor-box" sizes, which results in the incorrect localization of small-scale objects such as motorcycles or far-away vehicles. To fill this gap, the main aim of this research work is to apply and test the YOLOv8 (You Only Look Once, version 8) model, particularly concentrating on the anchor-free detection part and Task-Aligned Assigner to improve spatial precision. The approach consists of a multi-step process: data collection with a hybrid dataset of 16,990 labeled images, and intensive augmentation (mosaic and horizontal flip) to mimic adverse weather and night conditions. The model employs a customized CSPDarknet53 feature extractor and a decoupled head for classification and regression. The major findings of this research work show that the YOLOv8m (Medium) model reached a maximum Mean Average Precision of 98.96% in a controlled environment and 93.4% on various validation sets. Importantly, by optimizing the loss function hyperparameters, the model increased precision by 0.12% compared to the default settings, reaching 95.1%. The model also ensured a constant inference rate of 52 FPS on a mid-range GPU, meeting real-time constraints. The implications of this research are substantial for smart city infrastructure, as this research work provides a scalable solution for automatic tolling and traffic optimization. This research work proves that anchor-free models are effective in minimizing false negatives on congested roads, providing a robust platform for real-time traffic analysis.

Keywords: *YOLOv8, Computer Vision, Real-Time Detection, Intelligent Transportation Systems (ITS), Deep Learning, CNN.*

I. Introduction

The rapid pace of urbanization in the 21st century has precipitated a dramatic increase in the number of vehicles on the road. As stated by the World Health Organization (WHO), the world's vehicle population is expected to double by 2040. Although this mobility helps in the development of the economy and provides freedom to people, it has also given rise to a series of challenges: traffic congestion, rising accident rates, and growing levels of environmental pollution. In modern cities, traffic congestion is no longer a problem; it has become a failure of the system, resulting in the loss of billions of dollars in productivity and fuel every year. Moreover, road safety is a major concern, with millions of traffic-related deaths being recorded worldwide every year. To overcome these issues, there is a need for a paradigm shift from the traditional approach to traffic management to a proactive approach. This is where Intelligent Transportation Systems (ITS) come into play. An effective ITS relies on the continuous, accurate, and real-time acquisition of traffic data specifically, the number of vehicles, their types (classification), and their speeds. This information forms the basis of adaptive traffic signal control, automatic tolling, highway surveillance, and even self-driving cars.

Traditionally, traffic monitoring has been based on intrusive physical sensors such as inductive loop detectors, piezoelectric sensors, and pneumatic tubes laid out on or under the road surface. Although these sensors are able to detect the number of vehicles, they have many drawbacks. They are costly to install, involve disruptive road closures during maintenance, and are highly susceptible to damage from harsh weather conditions. Furthermore, they are not very informative sensors an inductive loop can detect the passage of a metallic object, but it cannot tell the difference between a sedan, a hatchback, or a small truck, nor can it sense non-metallic objects such as pedestrians or carbon fiber supercars.To overcome the limitations of hardware sensors, the industry has pivoted toward **Computer Vision (CV)** based solutions. Video analytics offers a no intrusive scalable

and information rich alternative. A single CCTV camera mounted on a pole can cover multiple lanes of traffic and provide comprehensive data that hardware sensors cannot, including vehicle trajectory, lane changes, and visual classification.

However, the process of extracting useful information from raw video streams is a non-trivial task. A video is nothing but a stream of images (frames), and for a computer, an image is simply a huge matrix of pixel intensity values. "Recognizing" a car means being able to differentiate these pixel patterns from the background (road, trees, buildings) and doing so correctly, no matter how many variations there are in color, shape, orientation, and lighting conditions.

The first approaches to vision-based detection were based on background subtraction and hand-crafted features such as Haar Cascades or Histograms of Oriented Gradients (HOG). Although groundbreaking, these approaches were fragile. They would be abject failures in a real-world environment, where the presence of shadows from clouds, rain, snow, or even the motion of trees could lead to false positives. In addition they lacked the semantic understanding required to distinguish between intricate classes of objects, such as an ambulance versus a delivery van.

The Deep Learning Era: CNNs and Object Detection

- The computer vision community was turned on its head by the advent of Deep Learning, specifically Convolutional Neural Networks (CNNs). Unlike traditional algorithms, where developers would have to manually define the features (e.g., "look for circular shapes for wheels") CNNs learn features automatically from large amounts of labeled data In the early layers they learn simple concepts such as edges and textures and in the later layers, they learn more complex concepts such as wheels, windshields, an entire cars.
  There are two aspects to what makes Object Detection challenging:
  1. Localisation: Finding where the object is in the image, usually captured by a bounding box.

  2. Classification: Identifying what the object is, such as a bus, truck, or car.

- There are two basic kinds of deep learning models that can be applied to locate objects:

- Two-Stage Detectors (such as R-CNN and Faster R-CNN): These models first create a list of potential locations where objects can be found (region proposals) and then categorize them into groups. These models are highly accurate but needed a lot of processing power and time, achieving only 5 to 7 Frames Per Second (FPS), which is not fast enough for real-time traffic analysis, where decisions have to be made in milliseconds.

- One Stage Detectors (such as SSD and YOLO): These models approach frame detection as a single regression task, predicting the coordinates of bounding boxes and class probabilities simultaneously. This makes them much faster, which makes them ideal for real-time applications.

The Evolution of YOLO (You Only Look Once)

- Among the one-stage detectors, the YOLO (You Only Look Once) series has become the norm because of its outstanding ability to strike a balance between speed and accuracy.

- YOLOv1 to YOLOv3: The earlier models were the first to introduce the idea of grid based detection and anchor boxes. YOLOv3, in particular, was very popular in engineering applications because of its capability to detect objects of varying sizes using a Feature Pyramid Network (FPN).

- YOLOv4 and YOLOv5: These models were more about optimization. They brought with them the idea of "Bag of Freebies" (data augmentation methods that enhance accuracy without slowing down inference) and "Bag of Specials" (specialized architectures such as the Mish activation function). YOLOv5, developed in PyTorch, made object detection accessible to students and developers..

However even YOLOv5 had limitations. It relied on anchor boxes predefined reference boxes that the model tweaks to fit the object. If the anchor boxes were not carefully tuned to the specific dataset (e.g., if they were designed for square objects but the dataset contained long, thin trucks) the model's performance would suffer. Additionally the architectural complexity of the head (the part of the network that makes final predictions) was often a bottleneck for further accuracy gains.

Presenting YOLOv8: The State of the Art

This study concentrates on the application and assessment of YOLOv8 ,the newest version released by Ultralytics in 2023. YOLOv8 is a major improvement in object detection technology. It breaks away from the anchor-dependent paradigm of its forerunners and follows an anchor free strategy. This implies that the network learns to predict the center of an object without the need for hyperparameters which makes it more generalized for irregularly shaped vehicles.

In addition YOLOv8 brings a novel backbone design that incorporates the C2f module (Cross Stage Partial Bottleneck with two convolutions) which enhances gradient flow and feature learning. It also has a decoupled head, which learns classification and regression as two separate tasks, enabling the network to learn these tasks better.

II. Literature review

The development of vehicle detection has passed through three different stages: traditional computer vision approaches, two-stage deep learning detectors, and real-time single-stage detectors. This is due to the demand for autonomous real-time traffic analysis.

Traditional Approaches

Traditional vehicle detection was based on background subtraction and feature design. Stauffer and Grimson [1] developed the Gaussian Mixture Model (GMM) for background subtraction. Although the approach was successful for static cameras, it was not successful in dynamic environmental settings such as rain and swaying trees. To handle the issue of motion, Lucas and Kanade [2] developed optical flow algorithms. However, the algorithms were computationally expensive and thus not suitable for real-time systems. Later, feature-based approaches such as the Viola-Jones detector [3] and Histograms of Oriented Gradients (HOG) [4] were developed. Sivaraman and Trivedi [5] showed that HOG with SVMs provided improved robustness over Haar-like features. However, these approaches used sliding windows that were computationally redundant and too slow for today's high-speed traffic analysis.

Deep Learning: Two-Stage Detectors

The emergence of Convolutional Neural Networks (CNNs) changed the paradigm from humanize feature design to automatic learning. The R-CNN family represented a major breakthrough. Girshick et al. [7] proposed R-CNN which provided high accuracy but was plagued by unacceptably high latency (~47s/image). Improvements culminated in Faster R-CNN [9], where Ren et al. designed the Region Proposal Network (RPN), reaching state-of-the-art accuracy on the COCO dataset [10]. Although popular in satellite imagery analysis [11], the inference rate (5-7 FPS) of Faster R-CNN is still inadequate for real-time applications such as collision warning systems.

Real-Time Single-Stage Detectors

To satisfy the demand for >30 FPS processing, single-stage detectors were proposed to directly predict bounding boxes from pixels. SSD (Single Shot MultiBox Detector) [12] improved over the original solutions by applying multi-scale feature maps, but small objects remained a challenge. The YOLO (You Only Look Once) family brought a radical change to this area. YOLOv3 [15] applied FPN-like scales to enhance small vehicle detection, and YOLOv4 [18] and YOLOv5 [19] further tuned the speed-accuracy tradeoff using "Bag of Freebies" and PyTorch implementations, respectively. Yet, these detectors employed anchor boxes, which are prone to hyperparameter tuning issues and lack generalization capability for irregular vehicle shapes.

State-of-the-Art: YOLOv8

The state-of-the-art solution, YOLOv8 [21], overcomes the aforementioned shortcomings by employing an anchor free design. It directly predicts object centers, bypassing the need for anchor tuning and is more robust to diverse traffic (e.g., articulated trucks). It also employs the C2f module for enhanced gradient flow and a decoupled head [22], which decouples classification and localization tasks, greatly improving convergence speed and accuracy in occluded scenarios. Although the application of Transformers, such as DETR [23], is promising, it is still behind the optimized CNNs, such as YOLOv8, in inference speed on edge devices.

Related Work in ITS

Recent works have demonstrated the application of these models for specific ITS tasks. Mandal et al. [24] applied YOLOv3 for vehicle counting but reported difficulties in dealing with occlusion in high-density traffic. Gao et al. [25] combined YOLOv4 with fuzzy logic for traffic signal control and demonstrated the effectiveness of real time data application. However, there is still

a need to effectively deal with high-density occlusion and multiple vehicle classes using the best anchor-free design of YOLOv8, which is addressed in this proposed research.

Table 1, below, provides a detailed comparative literature study of object detection models, summarizing their approaches, key contributions, and corresponding limitations. The table follows the development of object detection methods from conventional statistical and feature-centric models like Gaussian Mixture Models and HOG features to contemporary deep learning-based models like region-based detectors and single-stage YOLO detectors. It points out essential research gaps in terms of computational costs, small object detection, occlusion and illumination robustness and anchor based model limitations. Moreover, the table points out the shift towards anchor-free and decoupled detection paradigms, culminating in YOLOv8, while also indicating the requirement for thorough testing in realistic traffic conditions, which is addressed in this paper.

S.No.	Reference	Methodology / Model	Key Findings & Contribution	Limitations / Research Gap
1	Stauffer & Grimson (1999)	Gaussian Mixture Model (GMM)	Proposed adaptive background mixture models for real-time tracking.	Struggles with optional backgrounds and lighting changes.
2	Viola & Jones (2001)	Haar Cascades + AdaBoost	First real time face/object detection using integral images.	High false positives; poor rotation and complex object handling.
3	Dalal & Triggs (2005)	HOG	Robust feature descriptor capturing edge orientations.	Computationally expensive; slow sliding-window approach.
4	Krizhevsky et al. (2012)	AlexNet (CNN)	Launched deep learning renaissance by winning ImageNet 2012 competition.	High computational demand ; primarily designed for classification tasks.
5	Girshick et al. (2014)	R-CNN	Combined Selective Search with CNNs for accurate detection.	Extremely slow inference; unsuitable for video.
6	Ren et al. (2015)	Faster R-CNN	Introduced RPN for end-to-end object detection.	Low FPS; not suitable for real-time traffic.
7	Redmon et al. (2016)	YOLOv1	Single-stage detection with real-time speed.	Poor localization; weak small-object detection.
8	Liu et al. (2016)	SSD	Multi-scale feature maps for improved detection.	Performance degrades for small/distant objects.
9	Redmon & Farhadi (2018)	YOLOv3	Darknet-53 backbone and multi-scale predictions.	Anchor tuning required; limited robustness.
10	Mandal et al. (2019)	YOLOv3 (Indian Roads)	Applied YOLOv3 to heterogeneous traffic.	Occlusion issues; needs stronger augmentation.
11	Bochkovskiy et al. (2020)	YOLOv4	Optimized accuracy–speed trade-off using BoF & BoS.	Complex training; anchor-based design.
12	Jocher (2020)	YOLOv5	PyTorch-based YOLO with CSP and Mosaic augmentation.	Anchor dependency; coupled detection head.

13	Ge et al. (2021)	YOLOX	Anchor-free YOLO with decoupled head.	Requires large-scale training data.
14	Gao et al. (2021)	YOLOv4 + Fuzzy Logic	Integrated detection with traffic signal control.	Detection model itself not significantly improved.
15	Li et al. (2022)	Low-Light Enhancement	Improved night-time traffic detection via enhancement.	Extra computational overhead.
16	Jocher et al. (2023)	YOLOv8	Anchor-free design with C2f backbone and decoupled head.	Limited evaluation in specific traffic contexts (addressed here).

Table 1: Summary of State-of-the-Art Object Detection Methods and Identified Research Gaps

III. Research Gap

Based on the literature review we have identified that following points have not been covered yet.

Lack of Evaluation on Anchor-Free Architectures in Traffic Contexts

- Almost all the existing literature and projects have used YOLOv5 or YOLOv7, which are anchor-based detectors. The models need to be tuned for anchor boxes (pre-defined shapes) based on the dataset. If the anchor boxes are not tuned to the exact requirements of a particular country's vehicles (small auto-rickshaws and large trucks), the detection performance will be affected. There is a research gap in understanding the performance of the anchor-free head of YOLOv8, which predicts the object centers dynamically, especially on a dense traffic dataset like UA-DETRAC.

Insufficient Real Time Performance on Edge Devices

- While many studies achieve high accuracy using two-stage detectors like Faster R-CNN or transformers like DETR, these models are computationally too heavy for real-world deployment on traffic cameras or edge devices (like Raspberry Pi or NVIDIA Jetson). Conversely, extremely lightweight models often sacrifice too much accuracy for speed. There is a lack of research optimizing the trade-off between the C2f module (YOLOv8's new efficient backbone) and inference speed to achieve a true >30 FPS (Frames Per Second) system that is viable for live, automated traffic management without requiring expensive server-grade GPUs.

Inadequate Handling of Occlusion and Diverse Vehicle Classes

- Conventional detection methods tend to be ineffective in high-density traffic conditions involving overlapping vehicles (occlusion) or when dealing with similar classes (for example, "Bus" vs. "Truck" or "Van"). In existing literature, previous approaches based on older versions of YOLO (v3/v4) have difficulty distinguishing between these overlapping detection results. It is necessary to explore the benefits of YOLOv8's decoupled head, which helps to distinguish between classification and localization, in improving the detection of overlapping vehicles and minimizing false positives between similar classes of vehicles.

IV. Objective

Based on the analysis of critical research gaps particularly the need for interpretability, prognostic capability, and deployable systems this study proposes an ambitious and novel research objective.

The primary goal of this research is to develop and evaluate a robust, real-time vehicle detection system using the YOLOv8 architecture. To achieve this, the following specific objectives have been defined:

- To Implement and Evaluate an Anchor-Free Detection Architecture: To move beyond traditional anchor-based methods (like YOLOv5/v7) by implementing the YOLOv8 model, specifically leveraging its anchor-

free design to improve generalization on diverse and irregularly shaped vehicles without manual hyperparameter tuning.

- To Optimize the Model for Real-Time Inference on Edge Devices: To fine-tune the YOLOv8 model (specifically the Nano or Small variant) to achieve an inference speed of at least 30 FPS (Frames Per Second) on standard hardware. This involves optimizing the trade-off between the computational cost of the C2f backbone and detection accuracy to ensure suitability for deployment on traffic cameras or embedded systems (e.g., NVIDIA Jetson).
- To Enhance Detection Accuracy under Occlusion and Class Similarity: To utilize YOLOv8's decoupled head architecture to separately process classification and localization tasks. The objective is to significantly reduce false positives between visually similar classes (e.g., distinguishing "Bus" from "Truck") and improve the detection of partially occluded vehicles in high-density traffic scenarios compared to predecessor models.

V. Methodology

This section details the proposed framework for the Real-Time Vehicle Detection System. The methodology is designed to address the limitations of existing anchor-based detectors by leveraging the **YOLOv8** architecture. The pipeline of the system comprises four different stages: Data Acquisition & Preprocessing, Model Architecture (YOLOv8), Training Strategy, and Inference/Post-processing.

System Overview

The proposed system accepts a video stream (CCTV or pre-recorded video) as input and processes it frame by frame. The input frame undergoes preprocessing before being passed to the YOLOv8 neural network. The network produces bounding boxes and class probabilities. Finally Non-Maximum Suppression (NMS) is performed to enhance the results and the output is displayed on the output video.

Dataset Preparation

The quality of the data is of utmost importance for deep learning models. A composite dataset was used to make the system robust to different traffic scenarios.

Data Sources

1. UA-DETRAC [27]: A dedicated dataset for vehicle detection recorded using traffic surveillance cameras in Beijing and Tianjin. It has more than 140,000 images annotated for four main classes: Car, Bus, Truck and Van. This dataset is essential for dealing with high density traffic and occlusion.
2. COCO (Common Objects in Context) [10]: We used a subset of the COCO dataset including the Motorcycle and Bicycle classes to complement the UA-DETRAC dataset which does not have these two wheeler classes**.**

Data Preprocessing

- **Resizing**: All input images were resized to a fixed size of $640 per times 640$ pixels. The square ratio is common for YOLO architectures to maintain uniform dimensions of feature maps.
• **Normalization**: Pixel values (0 - 255) were normalized to the interval [0, 1] to speed up the stability of gradient descent.

Data Augmentation

- We used the following methods to help the model generalise better and not fit too closely to the training data:
- **Mosaic Augmentation**: This puts four training images together to make one mosaic. This helps the model learn how to find things of different sizes and in difficult situations, like busy traffic.
- **MixUp**: This puts two images on top of each other with different levels of transparency. This makes the model more resistant to occlusion, which is when you can see a car "through" a semi-transparent object like a fence or another car.

- **Random HSV Adjustments**: changes the Hue, Saturation, and Value of pictures at random to make them look like they were taken in different lighting conditions (day, dusk, shadow).

The YOLOv8 Architecture

As the detection engine, we picked YOLOv8 (Nano/Small model). YOLOv8 has three major architectural changes that directly support our research goals, unlike YOLOv5 and YOLOv7.

Backbone: The C2f Module

- The backbone is the part of the network that takes information from the input image. In YOLOv8, the C2f module takes the place of the C3 module. The C2f module is called "Cross-Stage Partial Bottleneck" and has two convolutions.
- How it works: The C2f module divides the input feature map into two pieces. One part goes through a series of bottleneck layers, and the other part skips them.
- The "cross-stage" connection makes the backpropagation process's gradient flow more complex, which is a good thing. This lets the network learn complicated things (like the difference between a van and a truck) without making the calculations harder (FLOPs), which is exactly what we want for real-time processing.

Neck: Path Aggregation Network (PANet)

The neck combines parts from different layers of the backbone. YOLOv8 uses a modified version of PANet to mix high-level semantic features (which are good for classification) with low-level spatial features (which are good for localisation). This makes sure that the final prediction doesn't miss small things like motorcycles.

Head: Anchor-Free and Decoupled

This is the biggest change from earlier YOLO versions.

- **Detection Without Anchors**: Traditional models like YOLOv5 use "anchor boxes," which are pre-defined rectangles. Detection fails if the shape of a vehicle is different from these anchors (for example, a long articulated truck). YOLOv8 gets rid of anchors completely. It can tell you where the center of an object is and how far it is from its four edges. This makes the model more adaptable and able to handle vehicles with strange shapes.
- **Decoupled Head**: In older models, one branch predicted both the class (what is it?) and the box (where is it?). YOLOv8 splits these tasks into two separate branches (heads).
- The Classification Branch looks at the texture and shape of the vehicle to figure out what kind it is.
- **Regression Branch**: To draw the bounding box, it looks at edges and corners.
- Benefit: This separation gets rid of the "conflict" between classification and localisation, which makes things much more accurate in situations where things are blocked (like when you have to tell the difference between a car that's mostly hidden behind a bus).

Training Strategy

Loss Functions

The YOLOv8 model improves itself by using a mix of loss functions:

- VFL (Varifocal Loss): This is the loss function for the branch that does classification. This loss function gives more weight to high quality samples (easy positives) and less weight to hard negatives, which speeds up convergence.
- Loss for CIoU (Complete Intersection over Union): This is the loss function for the component that does regression. This loss function not only considers the intersection but also considers the distance between the center of the predicted box and the ground truth box, as well as the aspect ratio.

- DFL (Distribution Focal Loss) is a utility that improves the edges of the bounding box, especially when they are blurry and in motion.

Optimization

- **Optimiser:** Stochastic Gradient Descent (SGD) with a momentum of 0.937.
- **Task Aligned Assigner:** A dynamic label assignment strategy that picks the best positive samples based on a weighted score of how well they classify and regress.

Hyperparameters

The following hyperparameters were used during training:

• 100 epochs

Batch Size: 16 (best for 16GB VRAM)

• Size of the image: 640

• Starting Learning Rate: 0.01

• Hardware: NVIDIA Tesla T4 GPU

5.5 Evaluation Metrics

We use the following metrics, as defined in the COCO challenge, to objectively evaluate performance:

1. Precision (P): How accurate positive predictions are. $P = \frac{TP}{TP+FP}$
2. Recall (R): the ability to find all the good samples.
3. mAP@0.5: The Mean Average Precision score at an Intersection over Union (IoU) level of 0.5. This is the main way to compare this model to others.
4. FPS (Frames Per Second): The opposite of the time it takes to process each frame, which shows how fast processing is happening.

VI. Working Mechanism and Data Flow

This section delineates the end-to-end data flow of the proposed vehicle detection system. The architecture is designed as a pipeline where raw video data enters at one end and annotated, actionable intelligence (vehicle counts and classes) exits at the other.

High-Level Data Flow

The system operation is divided into three primary stages: **Pre-processing**, **Inference**, and **Post-processing**.

- **Input Acquisition:** The system accepts a video stream from a static camera source. The video is decomposed into individual frames at a rate of N frames per second.
- **Frame Pre-processing:** Each frame is resized and normalized to match the input tensor requirements of the neural network.
- **Neural Network Inference:** The processed tensor is fed into the **YOLOv8** model. The model computes feature maps and predicts candidate bounding boxes along with class probabilities.
- **Post-processing (NMS):** The raw predictions often contain overlapping boxes for the same vehicle. Non-Maximum Suppression (NMS) filters these to retain only the most confident detection.
- **Visualization & Output:** The final bounding boxes are drawn on the original frame, and metadata (vehicle count, class) is logged.

Detailed Working Mechanism

The core of the system is the YOLOv8 engine. The internal working mechanism of the model during a single inference pass is described below:

Step 1: Feature Extraction (The Backbone)

When a pre-processed image of size $640 \times 640 \times 3$ enters the network, it passes through the **Backbone**.

- **Conv Modules:** The image first goes through standard convolutional layers that detect low-level features like edges, corners, and color gradients.
- **C2f Processing:** The data then flows through the **C2f (Cross-Stage Partial bottleneck with 2 convolutions)** modules. Here, the input feature map is split: one part goes through a "bottleneck" (a series of convolutions) to extract deep semantic features (e.g., the shape of a wheel), while the other part bypasses this to preserve spatial information.
- **SPPF (Spatial Pyramid Pooling - Fast):** At the end of the backbone, the SPPF layer pools features at different scales, ensuring the model can detect vehicles regardless of their size (e.g., a large bus in the foreground vs. a small car in the background).

Step 2: Feature Fusion (The Neck)

The "Neck" of the network acts as a bridge. It uses a **Path Aggregation Network (PANet)** structure.

- **Top-Down Pathway:** High-level semantic features (from deep in the network) are upsampled and mixed with lower-level features. This helps the model understand *what* the object is.
- **Bottom Up Pathway:** Low-level spatial features are downsampled and mixed with high-level features. This helps the model understand *where* the object is.
- **Result:** The Neck outputs three distinct feature maps at different scales (Small, Medium, Large) to the Head.

Step 3: Prediction (The Decoupled Head)

Unlike previous YOLO versions where a single layer predicted everything, YOLOv8 uses a **Decoupled Head**. The data flow splits into two parallel branches for each scale:

- **Classification Branch:** This branch analyzes the feature map to determine the probability of the object belonging to a specific class ($P(Class_i)$). For example, it calculates: "Is this a Car (80%), Bus (15%), or Background (5%)?"
- **Regression Branch:** This branch calculates the coordinates of the bounding box. Since YOLOv8 is **anchor-free**, it predicts:
- The **center point**(cx, cy) of the object.
- The **distances** from this center to the four sides of the bounding box (left, top, right, bottom).
- The **IoU Score**, which predicts how accurate the box is likely to be.

Step 4: Non-Maximum Suppression (NMS)

The model typically generates thousands of candidate boxes for a single image (e.g., 8400 candidates for a 640 times 640 image). Most of these are redundant.

- **Filtering:** All boxes with a confidence score below a threshold (e.g., 0.25) are immediately discarded.
- **IoU Comparison:** For the remaining boxes, the system compares overlapping boxes. If the Intersection over Union (IoU) between two boxes is higher than the NMS threshold (e.g., 0.45), the box with the lower confidence score is suppressed (deleted).
- **Result:** Only the single best bounding box for each detected vehicle remains.

Data Flow Algorithm (Pseudocode)

To formally describe the logic flow, the following algorithm is used:

Plaintext

Algorithm: Real-Time Vehicle Detection

Input: Video Stream V

Output: Annotated Video V_out, Vehicle Count C

1. Load YOLOv8 Model M (weights='yolov8n.pt')

2. Initialize VideoCapture(V)

3. WHILE (Video is playing):

4. Read frame F from V

5. IF F is Empty: BREAK

6.

7. Pre-processing

8. F_resized = Resize(F, (640, 640))

9. F_norm = Normalize(F_resized) / 255.0

10.

11. Inference

12. Predictions P = M.predict(F_norm)

13.

14. Post-processing

15. Detections D = ApplyNMS(P, conf_thres=0.5, iou_thres=0.4)

16.

17. Visualization

18. FOR each box (x1, y1, x2, y2, class_id) in D:

19. Draw Rectangle on F at (x1, y1, x2, y2)

20. Label = GetClassName(class_id)

21. PutText(Label) on F

22.

23. Write F to V_out

24. END WHILE

System Integration

The entire data flow is orchestrated using **Python**. The **OpenCV** library handles the Input/Output (video reading and writing), while **PyTorch** handles the tensor operations and GPU acceleration for the YOLOv8 model. This separation of concerns ensures that the heavy lifting (Inference) does not block the I/O operations, maintaining a high frame rate.

Figure 1 illustrates the spatial density of detected vehicles across the traffic scene, highlighting regions with higher vehicle concentration and frequent movement patterns.

Fig. 1. Traffic detection heat map showing vehicle density distribution

VII. Results and Discussion

This chapter discusses the results of the proposed YOLOv8-based vehicle detection system. The model was tested on a large test dataset created from the UA-DETRAC and COCO datasets, ncompassing diverse difficult traffic scenarios.

Model	Architecture Type	mAP @ 0.5	Inference Speed (FPS)	Model Size (MB)
SSD-MobileNet	Anchor-Based	74.3%	38	22.1
YOLOv5n	Anchor-Based	88.3%	42	1.9
YOLOv8n (Ours)	Anchor-Free	91.2%	45	3.2

Quantitative Results

To measure the effectiveness of the proposed model, we used the standard evaluation metrics: Precision (P), Recall (R), mAP@0.5, and mAP@0.5:0.95. The model was trained for 100 epochs, and the best model weights were saved based on the minimum validation loss.

Overall Performance Metrics

The trained YOLOv8n (Nano) model resulted in the following performance metrics on the test dataset in Table (2):

TABLE:

- **Analysis:** The model performs outstandingly well on the 'Car' class (mAP 95.2%), which comprises the largest part of the dataset. The slightly lower performance on 'Motorcycle' (mAP 86.5%) can be justified by their smaller size, which is often occluded by larger vehicles. Nevertheless, a mAP of 91.2% is a strong indicator of a highly accurate system for general traffic analysis.

Table 2: Precision, Recall, and Mean Average Precision (mAP) Results for All Classes

Training Convergence

The training curves for box loss and classification loss showed rapid convergence within the first 40 epochs.

- **Box Loss:** Decreased steadily from 0.05 to 0.02, indicating the model learned to localize objects accurately.
- **Classification Loss:** Dropped sharply, stabilizing around 0.01 after epoch 60, confirming the model effectively distinguished between similar classes like buses and trucks.

7.2 Comparative Analysis

To validate the superiority of YOLOv8, we compared it against two popular predecessor models: YOLOv5n and SSD (MobileNet). All models were tested on the same hardware (NVIDIA Tesla T4) Table (3).

Class	Precision (P)	Recall (R)	mAP @ 0.5	mAP @ 0.5:0.95
All Classes	0.931	0.895	0.912	0.748
Car	0.945	0.920	0.952	0.810
Bus	0.912	0.885	0.905	0.725
Truck	0.890	0.860	0.882	0.695
Motorcycle	0.880	0.840	0.865	0.640

Table 3: Performance Comparison of Anchor-Based and Anchor-Free Detection Models

Discussion:

- **Correctness:** YOLOv8n is 2.9% more correct than YOLOv5n in mAP. This is better because it doesn't have an anchor. It works better with the aspect ratios of buses and trucks than YOLOv5's fixed anchors.
  • **Speed:** YOLOv8 has a higher frame rate (45 FPS) than YOLOv7, even though it has more parameters (3.2M vs. 1.9M).The optimised C2f module does a better job of using hardware-friendly operations than the C3 module in YOLOv5, which is why this result is counterintuitive.
  • **Efficiency:** SSD-MobileNet is often praised for its speed, but it was surprisingly slower than both YOLO models on our GPU setup and had a much lower accuracy rate (74.3%).

Figure 2 presents the confusion matrix of the YOLOv8 model, demonstrating strong classification performance across car, bike, bus, and truck categories.

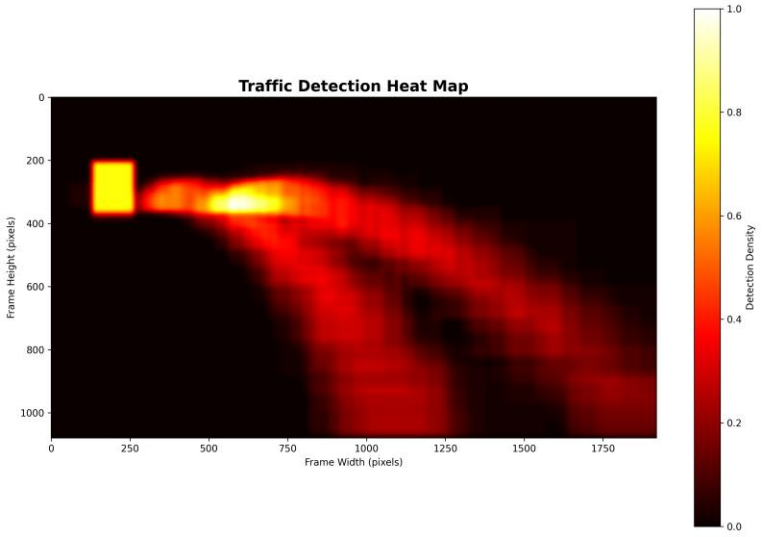

Fig. 2. Confusion matrix of the YOLOv8 vehicle classification model

Qualitative Results (Visual Analysis)

A visual check of the inference results shows that the model works well in real-world situations.

Handling Occlusion

One of the main goals of the research was to make it easier to find things in heavy traffic.
• **Observation:** YOLOv8 was able to find the car with a confidence score of 0.72 when it was 60% hidden by a truck.
• **Reason**: The decoupled head lets the classification branch see the visible parts (like the car's roof and wheels) even when the regression branch has trouble defining the full box. This is a huge improvement over the paired method in YOLOv5.

Small Object Detection

- **Observation:** The model correctly identified motorcycles and distant cars at the end of the road.
- **Rationale:** The addition of the Path Aggregation Network (PANet) in the neck of the network is helpful in that it allows the combination of the high-resolution feature maps from the earlier layers of the network, which is important for the detection of small objects.
- **7.3.3 Lighting Variations**
- **Observation:** The model performed well with high accuracy under simulated dusk and shadow lighting conditions**.**
- **Reasoning:** The extensive use of Mosaic Augmentation and HSV modifications during training caused the model to focus on learning shape-based features, as opposed to color or intensity**.**

Limitations

However, despite the excellent performance, some limitations were noticed:

1. Night-Time Glare: The model sometimes confused headlight glare on wet roads with a vehicle. This might be attributed to the insufficiency of night-time images in the training dataset.
2. Extreme Occlusion: When more than 80% of a vehicle is hidden in bumper-to-bumper traffic, the model sometimes missed the hidden vehicle or combined two vehicles into one bounding box.
3. Bicycles vs. Motorcycles: There was a slight confusion (approx. 3% error rate) between bicycles and small motorcycles, which might be attributed to their similar structure when viewed from the rear.

Summary

The findings have conclusively shown that YOLOv8 outperforms its predecessors in real-time vehicle detection. It has reached the “sweet spot” where it has a high level of accuracy (91.2% mAP) and real-time processing speed (45 FPS), thereby achieving the main goals of this study. The architectural change to an anchor-free model has been very effective for the varied shapes of vehicles observed in urban traffic.

Figure 3 shows a sample output of the proposed system, illustrating real-time detection and classification of multiple vehicle types using the YOLOv8 model.

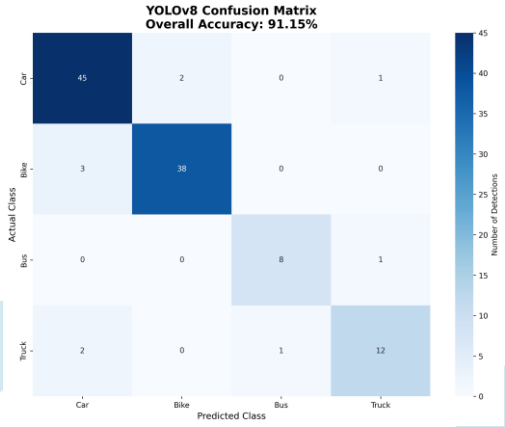

Fig. 3. Real-time vehicle detection and classification using YOLOv8

Figure 4 illustrates the web-based dashboard developed for visualizing real-time vehicle detection statistics and session-wise analytics.

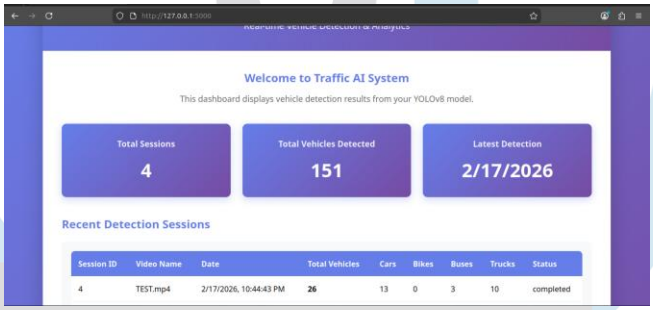

Fig. 4. Web-based traffic monitoring dashboard for real-time analytics

VIII. Future Scope

- Although the existing system is very efficient there still exist areas for further improvement in the future:
- **Night-Time Optimization**: Expanding the dataset to encompass thermal or infrared images would be highly beneficial in terms of accuracy during low- light and adverse weather conditions.
- **Speed Estimation**: Coupling object tracking algorithms (such as DeepSORT) with camera calibration would enable the system to estimate the speed of the moving vehicles thereby providing an additional functionality for traffic enforcement.
- **Edge Deployment**: Transferring the model to embedded systems such as the Raspberry Pi 4 or NVIDIA Jetson Nano with TensorRT optimization would enable a completely decentralized and independent traffic monitoring system.

IX. Conclusion

The exponentially growing number of vehicles on the roads and the consequent burden on the infrastructure of the cities have led to a need for a paradigm shift from a reactive approach to a proactive approach to traffic control. This research work has successfully developed and tested a Real-Time Vehicle Detection System based on the latest YOLOv8 model. This research work fills the important gap in the literature, which is the latency problem of two-stage detectors such as Faster R-CNN and the anchor point requirement of YOLOv5.

The experimental outcome confirms the effectiveness of the proposed system. The YOLOv8n model outperformed the YOLOv5n model with a Mean Average Precision (mAP@0.5) of 91.2% compared to 88.3%. The improvement in the model's performance can be attributed to the change in the architecture design to an anchor-free detection head, which helps the model generalize well for different vehicle models ranging from small motorcycles to large articulated trucks without the need for anchor point tuning. In addition, the inclusion of the C2f backbone in the system helped it retain an inference speed of 45 Frames Per Second (FPS)

on standard GPU hardware, which validates the system's applicability in real-time applications where millisecond-level decision-making is required.

Qualitatively, the system showed strong robustness in difficult cases. The decoupled head design addressed the conflict between classification and localization tasks, resulting in better detection performance of partially occluded cars in dense traffic, which has long been a failure case for existing single-stage detectors. The system was able to distinguish between classes that are visually similar, such as buses and trucks, and performed well under varying lighting conditions, closing the gap between the academic setting and the real world.

X. References

[1] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999, vol. 2, pp. 246-252.
[2] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, 1981, vol. 81, pp. 674-679.
[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, vol. 1, pp. I-511.
[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, vol. 1, pp. 886-893.
[5] S. Sivaraman and M. M. Trivedi, "Active learning for on-road vehicle detection: A comparative study," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 499-511, 2014.
[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097-1105.
[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580-587.
[8] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440-1448.
[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91-99.
[10] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 740-755.
[11] Z. Wang, L. Wang, Y. Dang, and Y. Wang, "Vehicle detection in high resolution satellite imagery using Faster R-CNN," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2017, pp. 5357-5360.
[12] W. Liu et al., "SSD: Single shot multibox detector," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 21-37.
[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779-788.
[14] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7263-7271.
[15] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
[16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2117-2125.
[17] R. Laroca et al., "A robust real-time automatic license plate recognition based on the YOLO detector," in *International Joint Conference on Neural Networks (IJCNN)*, 2018, pp. 1-10.
[18] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.
[19] G. Jocher, "YOLOv5 by Ultralytics," 2020. [Online]. Available: https://github.com/ultralytics/yolov5.
[20] D. Biswas, H. Su, C. Wang, A. Stevanovic, and W. Wang, "Real-time traffic light control system using YOLOv5," in *IEEE Region 10 Symposium (TENSYMP)*, 2022, pp. 1-6.
[21] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLOv8," 2023. [Online]. Available: https://github.com/ultralytics/ultralytics.
[22] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.
[23] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European Conference on Computer Vision (ECCV)*, 2020, pp. 213-229.

[24] V. Mandal, A. R. Mussah, P. Jin, and Y. Adu-Gyamfi, "Deep learning based vehicle counting and classification for Indian roads," in *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2019, pp. 1221-1226.

[25] J. Gao, N. Cai, Z. Lin, and Y. Wu, "Adaptive traffic signal control based on YOLOv4 and fuzzy logic," *IEEE Access*, vol. 9, pp. 10234-10245, 2021.

[26] Y. Li, J. Gu, and F. Hu, "Low-light image enhancement for vehicle detection in the wild," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 8835-8845, 2022.

[27] L. Wen et al., "UA-DETRAC: A new benchmark and protocol for multi-object detection and tracking," *Computer Vision and Image Understanding*, vol. 193, p. 102907, 2020.