Multimedia Tools and Applications
https://doi.org/10.1007/s11042-023-15520-9

Improved YOLOv5l for vehicle detection: an application to estimating traffic density and identifying over speeding vehicles on highway scenes

Navjot Singh[1] · Paras Saini[2] · Om Shubham[2] · Rituraj Awasthi[2] · Anurag Bharti[2] · Neetesh Kumar[3]

Received: 11 January 2022 / Revised: 6 July 2022 / Accepted: 19 April 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Vehicle detection and counting are getting progressively significant in highway administration due to the varying sizes of vehicles. This paper proposes a vision-based vehicle detection and counting framework. Initially, the COCO and BDD100K datasets are trained, employing improved YOLOv5l algorithm, using GhostBottleneck, for vehicle detection. Subsequently, the Centroid Tracking Algorithm counts the vehicles in videos uniquely. Later, vehicles are categorized as Light and Heavy vehicles. The model is evaluated over UCSD dataset for traffic estimation. Then, the speed of the recognized vehicle is computed to identify overspeeding. Lastly, number plate identification of the over speeding vehicles is performed. Vehicle detection performed well in precision, recall, F-score, and mAP. The centroid tracking algorithm had the best multiple object tracking accuracy and precision for vehicle tracking. The proposed model performed well in terms of detection rate, false alarm rate, and mean time to traffic detection.

✉ Navjot Singh
navjot@iiita.ac.in

Paras Saini
parassaini16@gmail.com

Om Shubham
omshubham99@gmail.com

Rituraj Awasthi
ra972472@gmail.com

Anurag Bharti
anuragbharti936@gmail.com

Neetesh Kumar
neetesh@cs.iitr.ac.in

1 Indian Institute of Information Technology Allahabad, Prayagraj 211015, India

2 Motilal Nehru National Institute of Technology Allahabad, Prayagraj 211004, India

3 Indian Institute of Technology Roorkee, Roorkee, Uttarakhand 247667, India

Published online: 31 May 2023

Keywords Vehicle detection · Vehicle counting · YOLO · Centroid tracking algorithm

1 Introduction

The rapid growth in the automobile industry and its impact on the highway and urban roads has impeded the traffic authorities from monitoring and controlling the traffic flow effectively. Based on the application of computer vision techniques [36], the system can solve many contemporary problems in traffic control and management; these applications can capture the traffic on an extensive scale and be analyzed in real-time. The necessary information to check, control and manage the traffic would be provided to the concerned authorities. Such systems are the essential components of intelligent transportation systems (ITS) [52]. The elevated vehicle density on the road network puts immense pressure on the smooth traffic flow and is responsible for frequent traffic jams, increased traffic accidents, increased travel time, etc. Such problems can be solved by observing the traffic states utilizing vision-based techniques. Moreover, vision-based techniques can also help detect and prevent violations of traffic rules like jumping traffic lights, tracking illegal turns, driving in no entry, etc. Traffic Surveillance [7] is used to analyze the traffic captured using a camera-based on human involvement. Manual traffic surveillance is most commonly implemented where a human operator is needed to monitor the traffic changes by watching different traffic streams from various cameras. The main disadvantage of using such a system is that there is too much information for a human operator to process at a given time. Moreover, the operator needs to be vigilant to address a sudden disruption that makes it a tiresome task to perform. On the other hand, intelligent traffic monitoring systems can automatically detect, recognize and extract useful information from the given input (image/video stream) via machine learning based on advanced computer vision techniques. Such systems are easy to install and can operate without any human intervention.

Considering this demand of ITS, this work proposes an intelligent deep learning system to automatically capture the traffic states using a vision-based object detection technique by taking the input (images/videos) and analyzing the traffic density on highways. Further, the proposed system assists the highway authorities by detecting the over-speeding vehicle by reading its number plate, which can be used for online challan to avoid traffic congestion or smooth traffic flow. The proposed system is flexible, adaptive, and easy to implement in a complex environment. It has numerous applications in Intelligent Transportation Systems. However, the following are the major motivational factors for this work.

- **Traffic Data Analysis** [48]**:** Vehicle detection and tracking applications are essential in civilian and military applications such as highway traffic surveillance control, management, and urban traffic planning. Vehicle detection processes on the road are used for vehicle tracking, counting the average speed of each vehicle, traffic analysis, and vehicle categorizing objectives. These may be implemented under different environmental changes. This article presents a concise overview of image processing methods and analysis tools used in building these previously mentioned applications that involved developing traffic surveillance systems.
- **Autonomous Vehicle's Decision Making** [3, 9, 11, 13, 20, 27]**:** One of the best applications of the proposed system can be in autonomous vehicles to capture the traffic states and make intelligent driving decisions. It can help an autonomous vehicle in deciding the next step and controlling the speed considering the traffic density and over-speeding

vehicles as it needs to know where all the objects are around the vehicle and types of the objects such as cars, pedestrians, traffic lights, road signs, bicycles, motorcycles, etc.
- **Intelligent Traffic Light Control System (ITLCS)** [18]**:** In order to design an effective ITLCS, a real-time traffic monitoring is required to capture the traffic states in terms of vehicles speed, traffic density on the lanes, types of vehicles, i.e., light/medium/heavy/emergency, etc., and it also requires identification of over-speeding vehicle or rule-breaking vehicle to punish the driver through an online challan. The proposed system meets the requirement for the same.

The utmost essential module for the applications stated above is vehicle detection. Vehicle detection is the primary task for applications like vehicle counting for traffic estimation and vehicle tracking for over speeding. In the article, the focus is more on the detection part of the vehicles. Traffic estimation and over speeding are selected as an application to check the efficacy of the proposed vehicle detection module over them. We suggest a lightweight vehicle detection model based on YOLOv5l to cater to the features of varying vehicle images. YOLOv5l, the newest one-stage object identification algorithm, has strong feature extraction capabilities, high detection accuracy, and quick convergence. The GhostBottleneck module replaces the BottleneckCSP module in the YOLOv5l Backbone network, which lightens the network and cuts down on the number of parameters. We have specifically improved the extraction of shallow detail features using the GhostBottleneck module.

The significant contributions of the proposed model are itemized as follows.

- This work estimates real-time traffic density on the highways via moving heterogeneous (low/heavy weight) vehicle detection and counting by avoiding the issue of recounting vehicles in the streaming video.
- An improved version of YOLOv5l is created which is fast and better than the existing YOLOv5l, by replacing the BottleneckCSP module with GhostBottleneck module.
- To assist highway authorities, the model identifies the over-speeding vehicles by calculating the velocity of vehicles. Once an over-speeding vehicle is detected, it is informed to the highway authority by detecting its number plate.
- Experimental study is done using Coco data set [30], BDD100K data set [49] and UCSD traffic video data set [8] by varying the number of inputs, and the model is tested on real streamed data. Results are evident from its effectiveness.

The rest of the paper is organized as follows. Related works are specified in Section 2. The proposed model is discussed in Section 3. Experimental setup and result analysis are done in Section 4. Finally, the conclusion and future work is mentioned in Section 5.

2 Related works

2.1 Vehicle detection

Currently, vehicle object detection is isolated into conventional machine vision strategies and complex deep learning techniques. Traditional machine vision techniques utilize the movement of a vehicle to isolate it from a fixed background picture. This strategy can be separated into three classes [2]: the strategy for utilizing background subtraction [1], the strategy for utilizing a consistent difference in video frames [39], and the strategy for utilizing optical flow [33]. Utilizing the video outline contrast strategy, the fluctuation is

determined by the pixel estimations of two or, on the other hand, three successive video outlines. Besides, the moving frontal area district is isolated by the limit [39]. The vehicle's halting can likewise be distinguished [38]. When the background in the video is fixed, it is utilized to build up the foundation model [38]. Each frame is contrasted with the background model, and the moving vehicle can likewise be portioned. The strategy for utilizing optical flow can distinguish the movement area in the video. The created optical stream field speaks to every pixel's course of movement and pixel speed [33]. Vehicle location techniques utilizing vehicle highlights have generally been utilized, such as SIFT and SURF techniques. For instance, 3D models have been utilized to finish vehicle discovery and characterization errands [16]. Utilizing the connection bends of 3D edges on the external surface of the vehicle [21], the vehicles are partitioned into SUVs, cars, and minibusses.

The utilization of profound convolutional networks (CNNs) has made stunning progress in the field of vehicle object location. CNNs have a solid capacity to learn picture includes and can play out various related assignments, for example, bounding box generation and classification [51]. The location strategy can be commonly isolated into two classes. The two-stage strategy creates a bounding box of the object employing different calculations and afterward groups the item by a CNN. The one-stage technique does not produce a bounding box; however, it legitimately changes over the situating issue of the object's bounding box into a relapse issue for preparation. In the two-stage technique, Region-CNN (R-CNN) [19] utilizes a particular locale search [45] in the picture. The other variants may include R-FCN [12], FPN [31], and Mask RCNN [23]. The picture contribution to the CNN must be of fixed size, and the more profound structure of the organization requires a long preparation time; furthermore, it expends a lot of capacity memory.

Liu et al. [34] proposed a target detection algorithm called Single Shot Multi Box Detector (SSD). SSD uses sliding window scanning on feature maps of different convolutional layers to detect targets of varying sizes. For smaller objects, SSD uses features from shallower layers, while for larger objects, SSD uses features from deeper layers. Liu et al. [35] proposed the GhostNet-SSD architecture, which deploys a Ghost module that divides the original convolution layer into two parts, first using fewer convolution kernels to generate a small number of internal feature maps, and then using simple linear change operations to generate ghost feature maps more efficiently. Faster R-CNN is an extension of Fast R-CNN, in which Ren et al. [42] introduced a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, allowing near-free region proposals. Bhalla et al. [5] concentrated on the concept of Behavioural Cloning and modified the DenseNet201 CNN model for self-driving. Li et al. [28] proposed an innovative self-driving car navigation approach that combines a vector field histogram local path planner adapted for modern 3D laser scanners with a global path planner based on satellite positioning and a digital map, as well as a custom built proportional integral derivative controller. Song et al. [44] proposed a computer vision based vehicle detection and counting system in highway scenes that uses deep learning. Zhang et al. [50] proposed a Deep Learning powered blockwise moving object detection method with binary scene modelling. For object identification and further vehicle speed estimation on roads, Byun et al. [6] uses the EfficientDet deep neural network-based technique.Liu and Juang [32] used YOLOv4 to detect vehicles and DeepSORT algorithms to track them to estimate traffic flow in lanes. Garca-González et al. [17] used Faster-RCNN deep architecture to follow vehicles and estimate road air pollution; while a traffic incident identification approach based on a global trajectory spatiotemporal map was proposed by Liang et al. [29].

The customary machine vision strategy has a quicker speed when identifying the vehicle yet does not create a great outcome when the picture changes in brightness, there is an occasional movement out of sight, and there are sluggish vehicles or complex scenes. In contrast, the deep learning models are better in terms of accuracy but at the cost of computation time. To cater with the computation time, one can skip few consecutive frames which are generally similar.

2.2 Vehicle tracking

The motion of all objects in a scene is estimated using multi-object tracking (MOT). MOT must continuously associate the same object in different frames based on the detection results in order to form a continuous trajectory. Because the global optimal solution cannot be obtained in the real-time online MOT algorithm, when the object is blocked or the detection result is incorrect, the associated trajectory may be interrupted, resulting in multiple trajectory fragments. The Multiple Hypothesis Tracking (MHT) [41] algorithm is used to solve the above problems by retaining all hypotheses of the tracked object and eliminating current uncertainty from subsequent observation data. Bewley et al. [4] proposed the Simple Online and Real-time Tracking (SORT) tracking algorithm to ensure industrial applicability. The Kalman filter is used to predict the object position, and the Hungarian algorithm is used to match the trajectory with the object one by one. The SORT algorithm can simply achieve the effect of real-time tracking, but when the object's movement is uncertain, the Kalman filter cannot provide an accurate prediction position, resulting in poor performance. Wojke et al. [47] proposed the Deep-SORT algorithm to solve this problem. Deep-SORT, which is based on the SORT algorithm, added a matching cascade mechanism to focus on the object's movement and appearance information to ensure that the trajectory matched correctly in consecutive frames. MOTDT [10], which is based on Deep-SORT, employs neural networks to solve unreliable detection and occlusion problems and creates a trajectory scoring function to select the best candidates. Based on MOTDT, the Jointly Learns the Detector and Embedding model (JDE) [46] extracts the embedding vector from the feature map and uses triplet loss to achieve faster tracking than MOTDT. However, the JDE must still perform object detection before matching objects.

3 Proposed model

The proposed model is composed of five phases. In the first phase, the COCO data set and BDD100K data set is trained via the improved YOLOv5l algorithm for vehicle detection. In the second phase, the Centroid Tracking Algorithm is used but only for videos. The primary purpose of this algorithm is to count vehicles uniquely. Moreover, in the case of images, we have taken care of the number of vehicles that our algorithm is detecting, as counting vehicles in an image is the same as the number of vehicles detected. In the third phase, vehicles are classified into two categories, namely Light Vehicles and Heavy Vehicles, after counting and detecting vehicles in the particular input. This helps in estimating the traffic density. In the fourth phase, we calculated the velocity of the detected vehicle. Based on the velocity of every vehicle, the over speeding vehicles are identified. Finally, in the fifth phase, license plate detection of the over speeding vehicles is performed. The flowchart of the proposed model can be seen in Fig. 1.

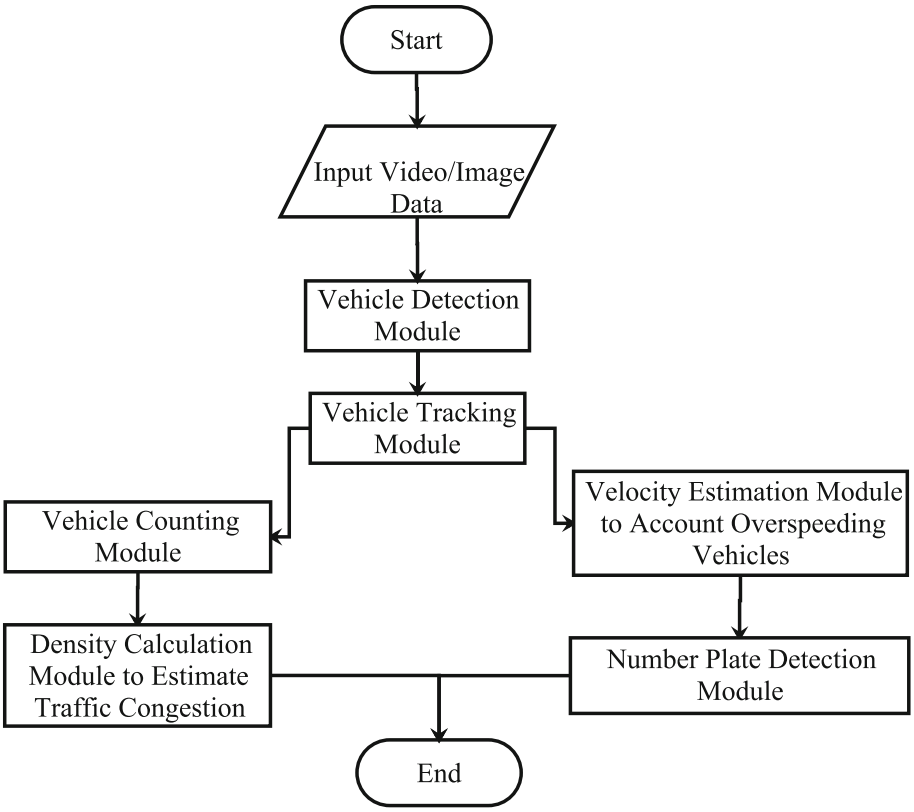

Fig. 1 Work flow of the proposed model

3.1 Vehicle detection

We are utilizing the improved version of YOLOv5l algorithm for detecting the vehicles. It utilizes a single convolutional neural network for both localization as well as classification of the object by drawing bounding boxes over it. The input image is divided into an $S \times S$ grids. The grid cell in which the center of an object lies is considered for that object. B bounding boxes are selected by each grid cell along with their confidence scores. The confidence score is calculated by computing the intersection over union (IOU) between the ground truth and the predicted box. Every bounding box comprises of 5 predictions: x, y, w, h, and confidence. The center of the box is represented by (x, y). The height h and width w are predicted relative to the image. Every grid predicts C conditional class probabilities, $Pr(Class(i)|Object)$. For a given grid cell only one set of class probabilities is considered irrespective of the number of boxes B. During testing, for each box the class-specific confidence scores are computed by multiplying individual box confidence predictions with the conditional class probabilities. The loss function for training shall include Bounding Box Position Loss (BBPL), Bounding Box Size Loss (BBSL), Confidence Loss (CFL), and Classification Loss (CL), which are computed by (1)–(4) and

detailed underneath [25, 40].

$$BBPL = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (x_i - \hat{x_i})^2 + (y_i - \hat{y_i})^2 \tag{1}$$

It computes the loss related to the predicted bounding box position (x, y), and λ is a constant. The function computes a sum over each bounding box predictor $(j = 0..B)$ of each grid cell $(i = 0..S^2)$. $\mathbb{1}_{ij}^{obj}$ is 1 if an object is present in grid cell i and the j-th bounding box predictor is responsible for that prediction and 0 otherwise. YOLO predicts various bounding boxes per grid. During training, only one predicted bounding box is answerable for each object, which has the highest IOU with respect to the ground truth.

(x, y) are the predicted bounding box position and $(\hat{x}, \hat{y})$ are the actual position from the training data. The bounding box size loss is computed as

$$BBSL = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w_i}})^2 + (\sqrt{h_i} - \sqrt{\hat{h_i}})^2 \tag{2}$$

The confidence loss is computed as

$$CFL = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C_i})^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C_i})^2 \tag{3}$$

C is the confidence score and $\hat{C}$ is the intersection over union of the predicted bounding box with the ground truth. $\mathbb{1}^{obj}$ equals one when there is an object in the cell, and 0 otherwise. $\mathbb{1}^{noobj}$ is just the opposite. λ is the weighing parameter necessary to increase model stability. The classification loss is calculated as

$$CL = \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \epsilon classes} (p_i(c) - \hat{p_i}(c))^2 \tag{4}$$

It looks similar to a standard sum-squared error for classification, except for the $\mathbb{1}^{obj}$ term. This term is used not to penalize classification error when no object is present on the cell. The detection network has 24 convolutional layers, followed by two fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers.

3.1.1 Improved YOLOv5l

YOLOv5 is the most recent and lightest version of previous YOLO algorithms, and it uses PyTorch instead of Darknet. As a backbone, it uses CSPDarknet53. This backbone handles the repeating gradient information in big backbones and combines gradient change into feature map, which speeds up inference, improves accuracy, and decreases model size by reducing parameters. It boosts information flow by using a path aggregation network (PANet) as a neck. PANet uses a new feature pyramid network (FPN) with many bottom-up and top-down layers. This enhances the model's low-level feature propagation. PANet improves the object's localization accuracy by improving localization in lower levels. Furthermore, the head in YOLOv5 is the same as in YOLOv4 and YOLOv3, resulting in three separate feature map outputs for multiscale prediction. It also aids in the efficient prediction of small to large items in the model. The image is first given to CSPDarknet53 for feature extraction, then to PANet for feature fusion. Finally, the results are generated by the YOLO layer. Figure 2 depicts the architecture of the improved YOLOv5l algorithm. The Focus

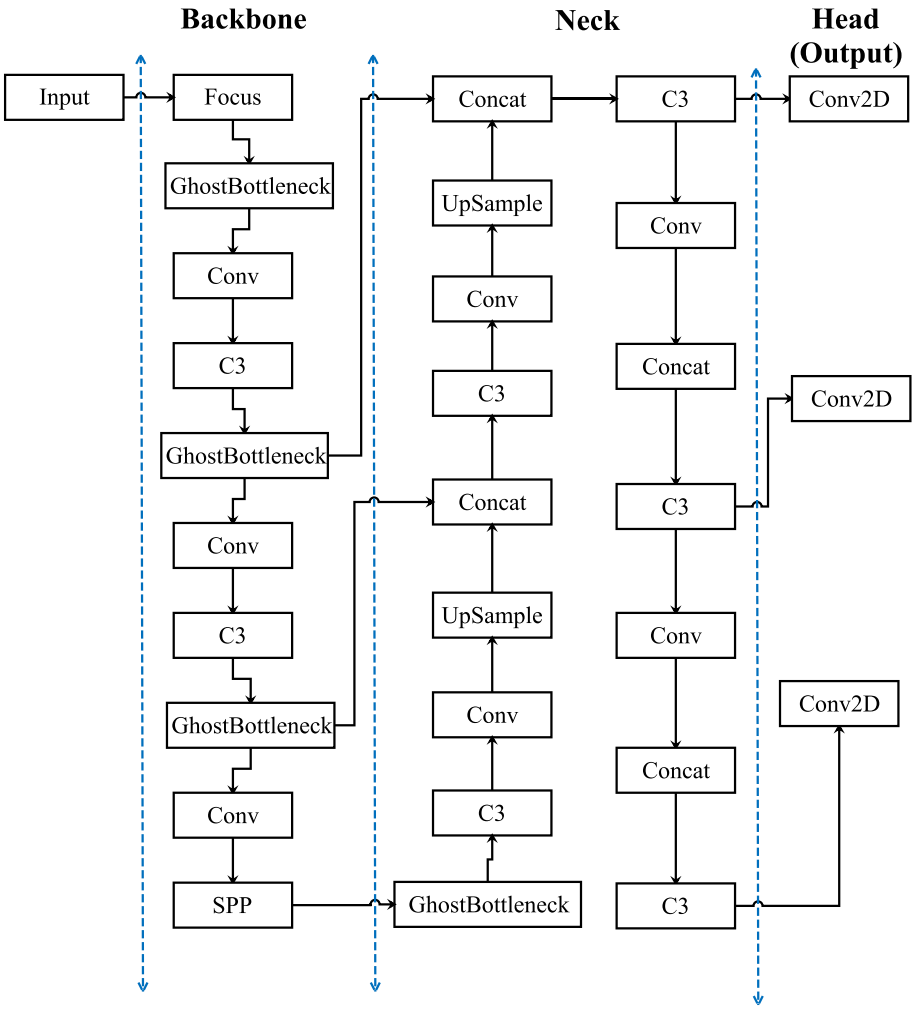

Fig. 2 Improved network architecture of YOLOv5l

layer [25] is an evolution of the YOLOv3 structure. It takes the first three layers of YOLOv3 and replaces them with a single layer in YOLOv5. In addition, Conv denotes a convolution layer. C3 is made up of three convolution layers and a bottleneck-cascaded module. Spatial pyramid pooling (SPP) is a pooling layer that is used to remove the network's fixed size constraint. Upsample is used in the nearest node to upsample the previous layer fusion. Concat is a slicing layer that slices the previous layer. The last three Conv2d are detection modules that are used in the network's head.

The CSP bottleneck is used in the original YOLOv5 network to deepen and enhance feature extraction. However, we have discovered that using some modules with reduced computing costs to simplify the structure of the model can also produce acceptable experimental results in the job of vehicle detection. We implemented a lightweight Ghost-Bottleneck [22] in place of the original network's CSP bottleneck to minimize the model's size and speed up inference to enable our model to be deployed on low-performance devices

with limited memory, such as track inspection cars or embedded devices. The fundamental concept behind the Ghost bottleneck is to create many richly detailed feature maps using low-cost linear operations. To be more precise, the feature map is first subjected to a small number of conventional convolution operations in order to produce intrinsic features. Next, the feature map is subjected to some low-cost linear transformation to produce another portion of the feature. Finally, the two components are integrated to produce the final output feature.

In order to extract rich shallow and deep feature information while lowering model parameters, the GhostBottleneck module in the Backbone network fully exploits the information redundancy properties of feature mapping. Essentially, two Ghost [22] modules are stacked to create the GhostBottleneck module. As seen in Fig. 3, the first Ghost module is used as an expansion layer to raise the number of channels. The second Ghost module then uses the input feature maps from the residual edges to lower the number of channels and add them. The Ghost module first collects a portion of the feature maps using a minimal amount of regular convolution, and then creates additional feature maps from these partial feature maps using a series of linear operations. The two sets of feature maps are then concatenated in the desired dimension. The generated feature maps are computed using (5).

$$Y' = X * f' + b \tag{5}$$

where $X \in \Re^{w \times h \times c}$ represents the input feature maps and, w, h, c its width, height, and number of channels respectively. $f' \in \Re^{c \times m \times k \times k}$ represents total $c \times m$ kernels of size $k \times k$ each, $*$ is a convolution operator and b is a bias. $Y' \in \Re^{w' \times h' \times m}$ represents m generated feature maps of width w' and height h'. Then, for each feature map of Y', we use a series of linear operations to regenerate s feature maps, which are computed using (6).

$$y_{ij} = \Phi_{ij}(y'_i), \forall i \in [1, m], j \in [1, s] \tag{6}$$

where y' is the i-th feature map of Y', $\Phi_{ij}(y'_i)$ is the j-th linear operation used by y' to generate the j-th Ghost feature mapping. The output of the Ghost module are the $m \times s$ feature maps $P = [p_{11}, p_{12}, ..., p_{ms}]$.

The input feature map's input channel count can be increased by s times, and the number of convolution kernels can be flexible. The tolerance to changes in the geometric features of the vehicles can be successfully increased, and the parameter redundancy can be decreased by combining the two Ghost modules with a deep-wise convolution. To increase the expressive power of the neural network, batch normalization (BN) is introduced after the convolutional layer of each module, and the hard-Swish [24] activation function is applied after the convolutional layer of the two Ghost modules.In order to obtain the same number of feature maps with fewer calculations, the Ghost module performs a small number of regular convolution operations and uses linear operations to reduce feature redundancy effectively. This reduces the number of parameters and calculations required by the entire model. Additionally, we have increased the thickness of the GhostBottleneck module in the shallow network of the Backbone, strengthened the network's feature extraction capabilities, and achieved multi-scale feature extraction from shallow layer to deep layer to maximize the use of the high-resolution features extracted in the shallow network.

YOLO algorithm and its variants [37] can detect numerous types of objects present in an image. Nevertheless, our focus is on vehicles, which include classes like cars, buses, trucks, and many more. Once the objects are identified by YOLO, vehicles are extracted out from them by selecting only those bounding boxes which lie in the different vehicle categories. The outcome of the vehicle detection phase can be seen in Fig. 4.

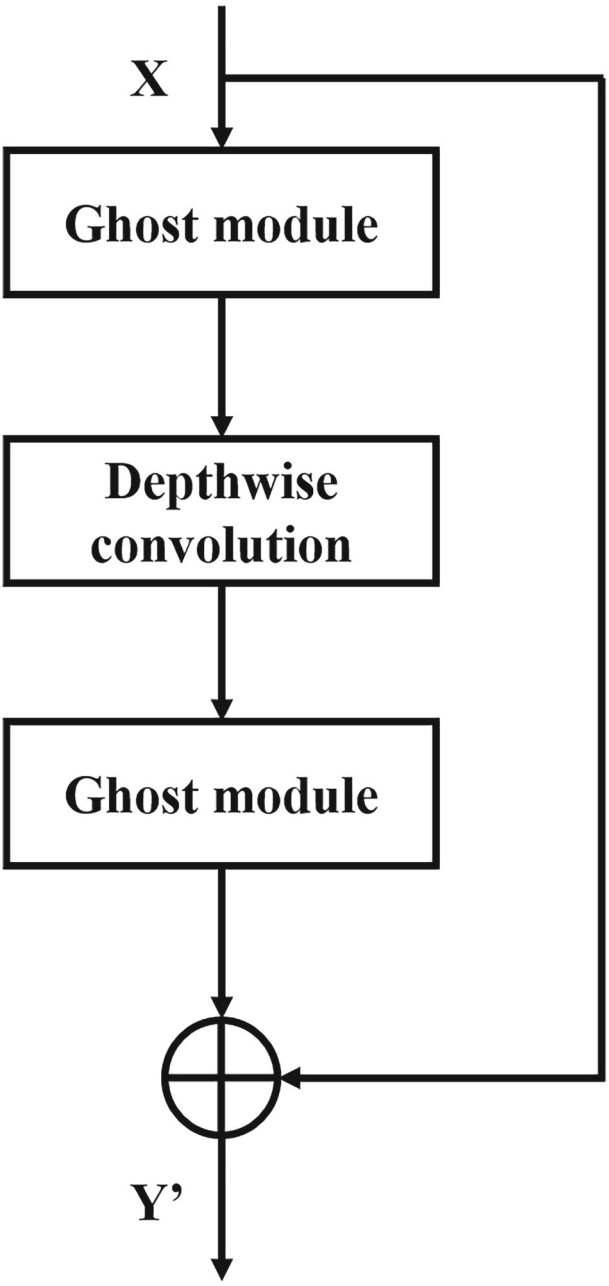

Fig. 3 GhostBottleneck module

3.2 Vehicle tracking and counting

In this phase, the vehicles are to be counted. The problem appears simple in case of an image but it is not easy to comprehend in case of a video. For an image, the vehicles detected

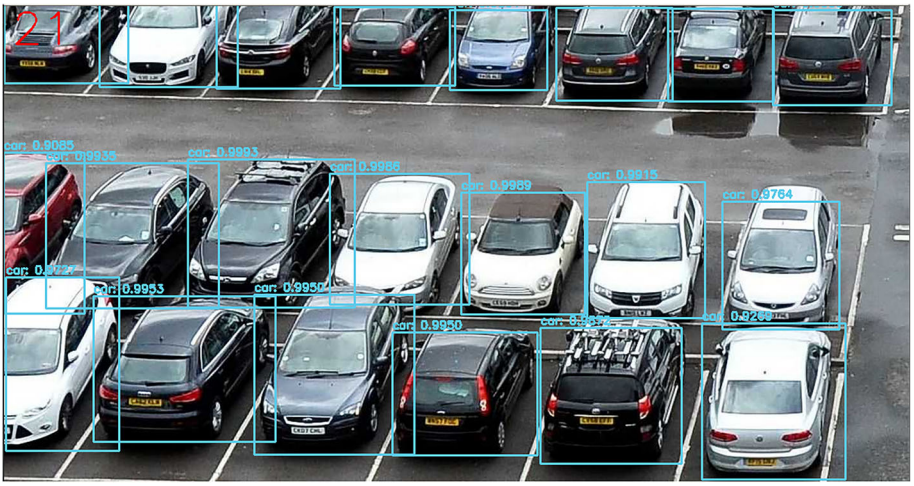

Fig. 4 Detection using improved YOLOv5l

in Section 3.1 equals the total vehicle count. But in case of a video, we cannot consider the count as the sum of total vehicles detected in all the frames of the video as we would have recounted the same vehicles appearing in various frames. We need to count the vehicles present in the video uniquely. For this purpose a centroid tracking algorithm [43] is employed. The centroid tracking algorithm is a multi-step process.

- Step 1-Accept bounding box coordinates and compute centroids: The bounding box for objects are generated by YOLO algorithm for every frame. A unique ID is given to every bounding box generated initially. The centriods of these bounding boxes are then computed. Figure 5 depicts this scenario.
- Step 2-Compute Euclidean distance between new bounding boxes and existing objects: Step 1 is applied to the upcoming frames and new unique IDs are given to only those objects which are coming the the frame for the first time. To handle this situation a scenario is shown in Fig. 6. The centriod of bounding boxes generated in the next frame are shown in yellow, while for the previous frame the old centriods are shown in purple. Then the Euclidean distance is computed between every pair of old and new centriods as shown in green and red arrows.
- Step 5-De-register old objects: Any sensible object following calculation should have the option to deal with when an item has been lost, vanished, or left the field of view. Precisely how would we handle these circumstances, is truly reliant on where our object tracker is intended to be sent, yet for this usage, we will de-register old items when they can't be coordinated to any current items for an aggregate of N ensuing frames.
- Step 3-Update (x, y)-coordinates of existing objects: Only those pair of points are considered which have minimum Euclidean distance between them as shown by green arrows in Fig. 7.
- Step 4-Register new objects: Those new centriods which could not be paired with the old centriods due to higher Euclidean distance in comparison to the other centriods are given new unique IDs as shown in Fig. 8. The above steps can be repeated for all the frames.

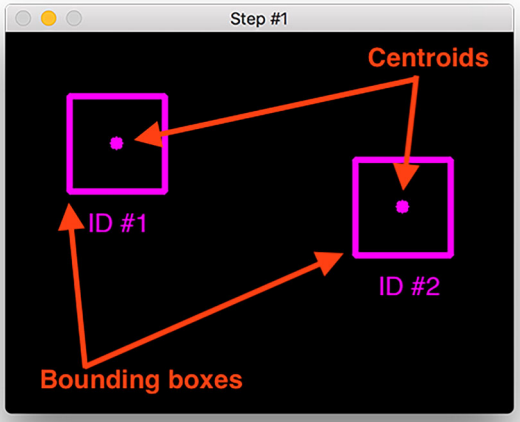

Fig. 5 Accepting bounding box coordinates and computing centroids [43]

3.3 Density calculation to determine traffic congestion

Density of the traffic is the number of vehicles in a certain area of the road. To calculate density, we define a fixed area on the road lane under consideration using two lines same as our counting lines, which will tell us the number of vehicles present in the area enclosed by the lines. By fixing a certain threshold on the number of vehicle in the area, we can define whether the traffic on the road is high or low. This is the case considered assuming the equal sizes of vehicles. however, we know that a heavy vehicle (bus or truck) covers more area in comparison to the light vehicles (car). Thus, in order to bring heterogeneity in vehicles, the

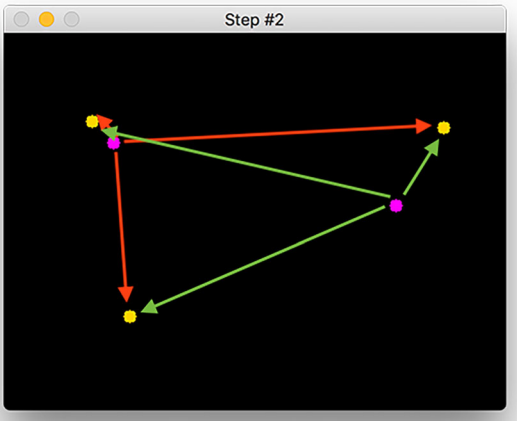

Fig. 6 Computing Euclidean distance between new bounding boxes and existing objects [43]

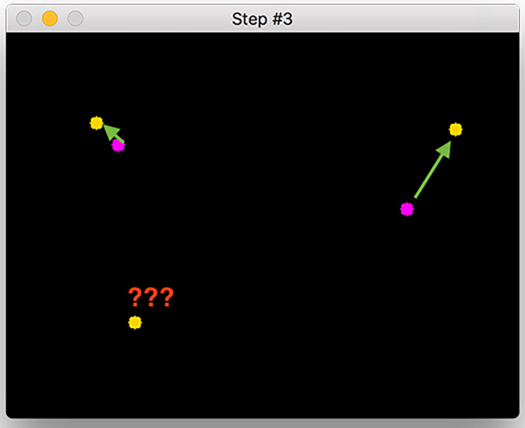

Fig. 7 Updating (x, y)-coordinates of existing objects [43]

counting is also done under two categories: Heavy vehicles and Light vehicles. An appropriate threshold on the combination of vehicles may be applied, taking into consideration the road size, to decide over density.

3.4 Velocity calculation for identifying over-speeding vehicles

We have analysed different approaches for the calculation of velocity of vehicles, but the calculation of distance is the prior step to calculate velocity. The steps to calculate distance are as follows.

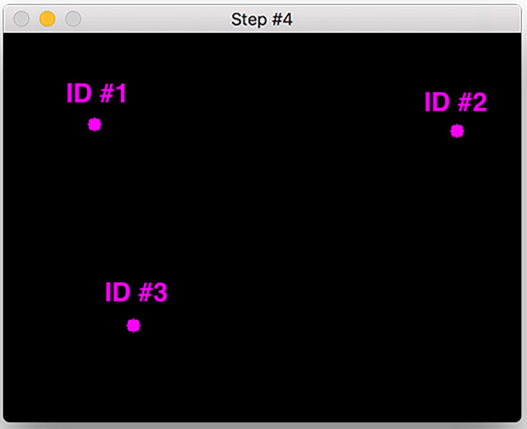

Fig. 8 Registering new objects [43]

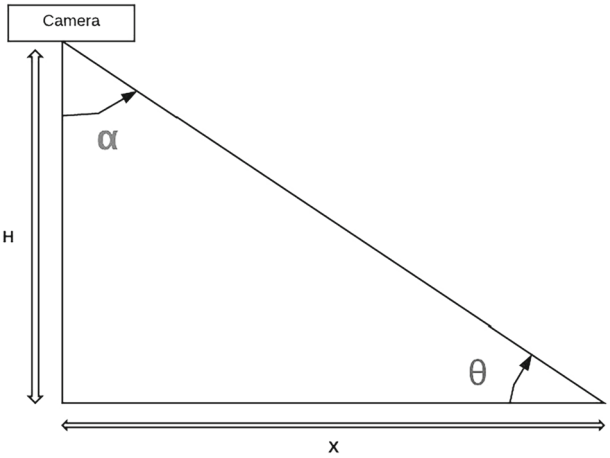

Fig. 9 Distance calculation using trigonometry

- **By taking the actual measurements:** It is a very simple way of getting distance, as we have taken two lines and also know the positions of two lines. This way, actual distance is calculated by the measurement of vehicle between two lines.
- **Using Trigonometry:** If H represents height of camera from the ground which is known to us, we have to calculate X, which is the distance travelled by the vehicle, as shown in Fig. 9, using trigonometric (7) and (8).

$$\tan\theta = \frac{H}{X} \tag{7}$$

$$X = H\cot\theta \tag{8}$$

the angle α is decided while installing the camera. The angle θ can be calculated using (9).

$$\theta = 90^o - \alpha \tag{9}$$

Therefore, by using some easy trigonometric equations we can calculate the value of the distance travelled by vehicle.

- **Mapping pixel to actual distance:-** We are calculating the distance between two lines in pixels. The two lines are prior known to us, therefore, we can easily get the distance as we know the coordinates of two lines already. Let D be that distance in pixels. We are calculating a ratio to convert the pixel distance to actual distance unit.
  - **Ratio Calculation**:- Let x be the width of car in pixels that we are getting by using boundary box and let y be the average width of car in feet, so the ratio R is calculated using (10).

$$R = \frac{x}{y}, \tag{10}$$

  - **Unit Conversion**:- Now, we need to convert the distance in pixels to feets. So, we are using the ratio (R) to solve that purpose. CD, as computed in (11), is the final distance that we will be using to calculate velocity of vehicle.

$$CD = D * \frac{y}{x} \tag{11}$$

Further the velocity is calculated as follows.

- **Frame By Frame Calculation:-** Here, we store the coordinates of the vehicle for each frame, we are calculating distance for the vehicle in each consecutive frame using the time difference between two frames. Since, we have both distance and time to get the velocity.
  - **Limitation:-** It is very time consuming, as there will be only negligible difference between the positions of a vehicle in two consecutive frames, therefore, it is worthless to calculate the distance between consecutive frames.
  - **Advancement:-** Instead of calculating distance between position of vehicle in consecutive frame, we can calculate distance by skipping a threshold amount of frames. It speeds up the algorithm.
- **By Choosing Entering And Crossing Lines:-** We use this algorithm to calculate the velocity of moving vehicle. In this, we have two fixed lines, one is entering and another is crossing line. We note the centroid coordinates of vehicle when vehicle centroid intersects with the entering line. We also have a frame counter which is used to find out the frame number. when the same vehicle crosses the crossing line, we calculate the distance between the positions of centroid. We store frames per second (fps) whenever the vehicle intersects entering line, and when vehicle intersects the crossing line we check the frame number, and by using fps we calculate the time in seconds to reach that particular frame. Thus, we have both time and distance to calculate velocity using these parameters.

Since, every road has a speed limit. Any vehicles whose velocity is above the limit specified, shall be an over-speeding vehicle. The screenshot of the vehicle can be captured using the coordinates of the bounding box of that vehicle.

3.5 Number plate detection

This phase involves two steps: first, Detection of the rectangular number plate region and extraction from the image. Second, Recognizing the characters of a number plate. Following are the steps used to determine the Number Plate region in an image:-

- Image Enhancement: A picture with a complex foundation generally contains noise that prompts undesired edge densities in the edge picture as demonstrated in Fig. 10. In this way, noise should be disposed of before edge discovery. For image enhancement, histogram equalization is used.
- Vertical Edge Detection: Sobel operator [26] is applied to find the approximate absolute gradient magnitude at each point in the image, as shown in Fig. 11.
- Eliminating Redundant edges: At times, vertical edges separated from those in the plate district, might be more extraordinary. Thus, we need to take out all vertical edges separated from those in the plate district, as shown in Fig. 12. Noise reduction, image enhancement and edge detection play important role in vehicle detection approaches like suggested in [15].
- Rectangular Number Plate Region Localization: The neighborhood district that has the most elevated vertical edge thickness in the dispensed with vertical edge picture is the Number Plate area. The dispensed with vertical edge picture is separated into neighborhood 16×16 squares. Each square in the edge picture compares to a pixel in the new picture, as shown in Fig. 13.

Fig. 10 Edge detection without image enhancement

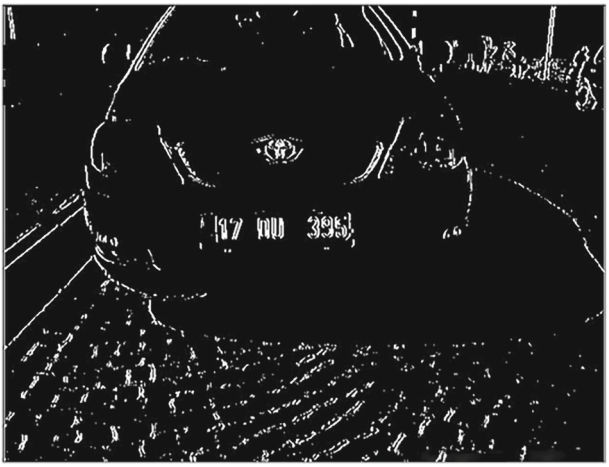

Fig. 11 Vertical edge detection using Sobel operator

Fig. 12 Eliminating redundant edges

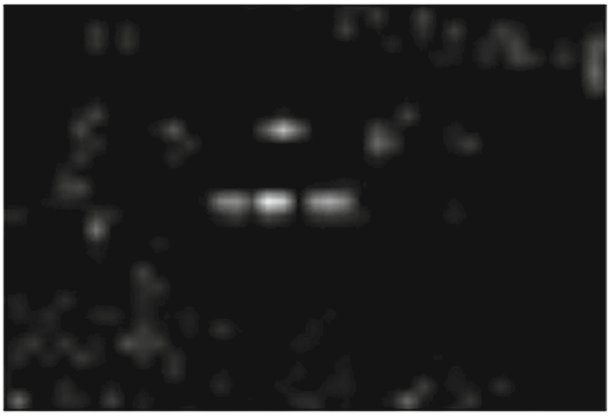

Fig. 13 Rectangular number plate region localization

- The other, undesirable light pixels are ignored, so the qualities that are lower than a predefined limit are set to nothing. The image obtained after the elimination is shown in Fig. 14.
- After the disposal, the picture is convolved with a channel that comprises of a framework of ones to find the up-and-comer of Number Plate district, as shown in Fig. 15
- Since the most splendid district in the picture is the Number Plate locale, resetting values that are more modest than the greatest estimation of the network to nothing, as seen in Fig. 16.
- To decide the genuine area on the original picture, the pixel estimations of the nearby areas (16×16) in the original picture comparing to each white pixel in new picture are set to one, and the pixels of the excess neighborhood districts are set to nothing. At last, by deciding the directions of the rectangular plate area, we can acquire the rectangular Number Plate district, as seen in Fig. 17.
- Character recognition [15] is done using Artificial Neural Network. The recognised Number Plate image will be the input to the network. The Network is based on a

Fig. 14 Removing unwanted light pixels

Fig. 15 Locating candidate number plate

Fig. 16 Resetting values that are smaller than maximum value

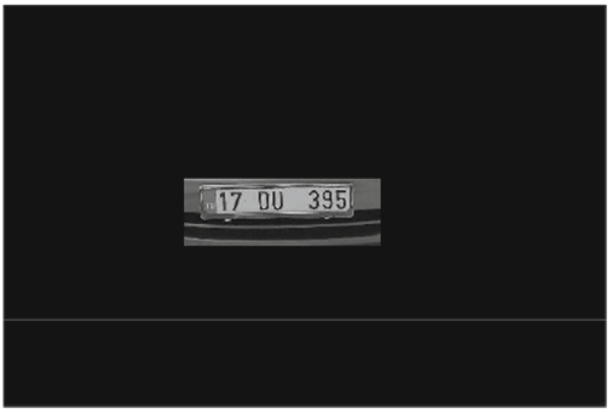

Fig. 17 Determining the actual location on the original image

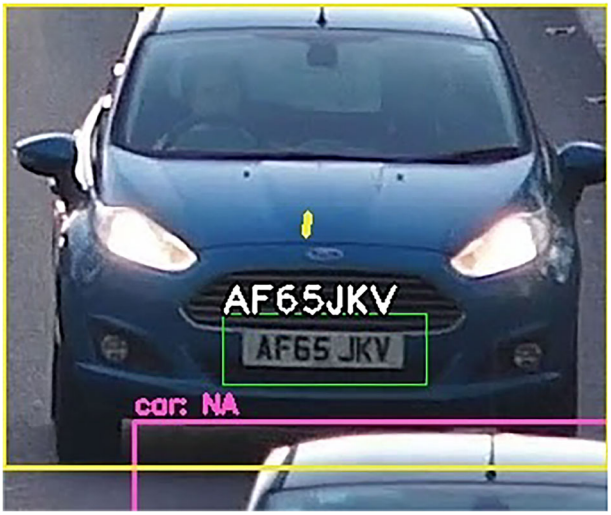

Fig. 18 Final output image with detected number plate

pre-trained model for character recognition [14]. The final output image with detected number plate can be seen in Fig. 18.

4 Experimental study

4.1 Dataset and libraries used

COCO Data Set: COCO is a Common Objects in Context dataset [30]. COCO was an activity to gather normal pictures, the pictures that reflect ordinary scene and gives logical data. In ordinary scene, numerous items can be found in a similar picture and each ought to be named as an alternate article and portioned appropriately. COCO dataset gives the marking and division of the items in the pictures. It contains 300K pictures with more than 200K labels. There are around 1.5 million item examples with 80 object classes.
BDD100K Data Set: With 100K videos (40 seconds each) and 10 tasks to evaluate the exciting progress of image recognition algorithms on autonomous driving, it is the largest driving video dataset [49]. The dataset is diverse in terms of geography, environment, and weather, which is useful for training models that are less likely to be surprised by new conditions. In this paper, BDD100K dataset is used for two tasks:

- Vehicle detection: 100K images are extracted from the 100K videos, which are the frames at the 10th second in the videos. The images are divided into three sets: training (70K), validation (10K), and testing (20K).
- Vehicle tracking: BDD100K also provides a multiple object tracking (MOT) dataset that includes 2,000 videos with approximately 400K frames. Each video is about 40 seconds long and annotated at 5 frames per second, for a total of about 200 frames per video. The dataset is divided into 1400 videos for training, 200 for validation, and 400 for testing.

UCSD Traffic Video Data Set: It includes 20-minute surveillance videos with a resolution of 320*240 of a highway in Seattle, Washington, USA. The dataset [8] categorizes traffic congestion into three categories (congestion, low speed, and normal speed) and takes rainy and sunny weather into account.

Language and Libraries Used: The experiments were conducted over Python using the following libraries: OpenCv, Numpy, SciPy, Tensorflow.

4.2 Result analysis

4.2.1 Calculating accuracy on various input videos

For the testing of the model, several videos were taken as input. The accuracy in case of a video is computed by (12).

$$Accuracy = \frac{No.\ of\ vehicles\ counted}{No.\ of\ vehicles\ actually\ crossed} \tag{12}$$

where,

- **Number Of Vehicles Counted:** It shows the number of vehicles that are counted by our detection system.
- **Number Of Vehicle actually crossed:** It shows the combination of both number of vehicles detected and number of vehicles that have not been detected.

The following varieties of images were preferred for evaluation.

- Many vehicles in a frame as shown in Fig. 20 with less area for background.
- Frames taken in night as shown in Fig. 19.
- Background covering majority of the area in the frame with less area covered by roads, as shown in Fig. 21.

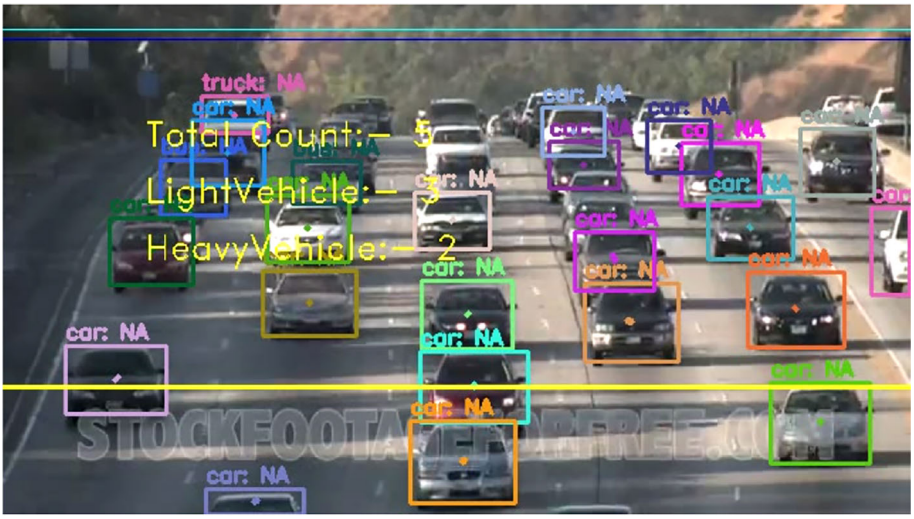

Fig. 19 Detecting and counting vehicles in video (Result-1)

Table 1 Analysing the accuracy for various input videos

Input videos	Number of vehicles actually crossed	Number of vehicles counted	Accuracy(%)
Input 1	27	25	92.59
Input 2	8	8	100
Input 3	20	16	80
Input 4	28	26	92.86
Input 5	32	28	87.5

The outcome of the analysis of accuracy over the sample videos can be seen in Table 1 and qualitative evaluation in Figs. 19, 20, 21 and 22. From the results, it can be inferred that the accuracy of the proposed model is acceptable to act in different traffic conditions. As the proposed model works under different lighting conditions and offering effective results.

4.2.2 Calculating accuracy on various input images

Further, to test the model on various images, a set of experiments were done by taking multiple images as input. For this scenario, the accuracy of the proposed model is computed by (13).

$$Accuracy = \frac{No.\ of\ vehicles\ detected}{No.\ of\ vehicles\ actually\ present} \tag{13}$$

where,

- **Number Of Vehicles Detected:** It shows the number of vehicles that are detected by our detection system.
- **Number Of Vehicle actually Present:** It shows number of vehicles that are present in the image.

The analysis of accuracy over sample images can be seen in Table 2 and qualitative evaluation in Figs. 23, 24 and 25. The following varieties of images were preferred for evaluation.

- Many vehicles in an image were partially occluded as shown in Figs. 23, 24, 25 and 26.
- Blurred images where the vehicles are not clearly discernible from the background as shown in Fig. 27.

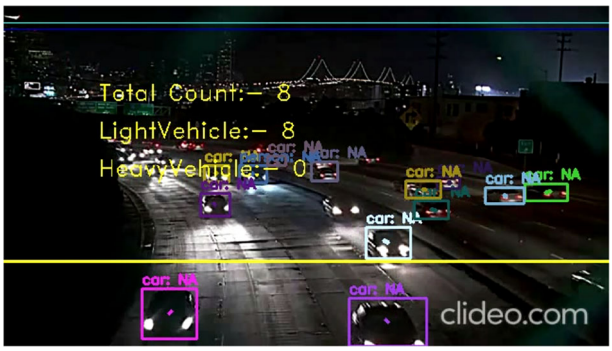

Fig. 20 Detecting and counting vehicles in night video (Result-2)

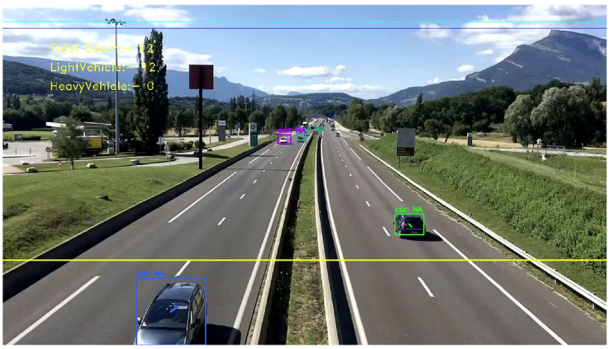

Fig. 21 Detecting and counting vehicles in video (Result-3)

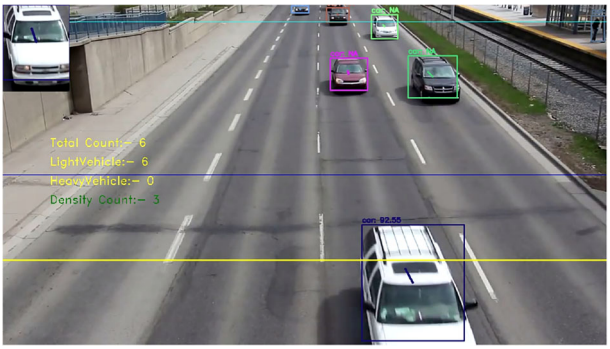

Fig. 22 Density count to determine traffic

Table 2 Analysing the accuracy for various input images

Input images	Number of vehicles detected	Number of vehicle actually present	Accuracy(%)
Input 1	73	75	97.33
Input 2	23	27	85.18
Input 3	21	21	100
Input 4	15	16	93.75
Input 5	28	28	100

Fig. 23 Detecting and counting vehicles in an image(Result-1)

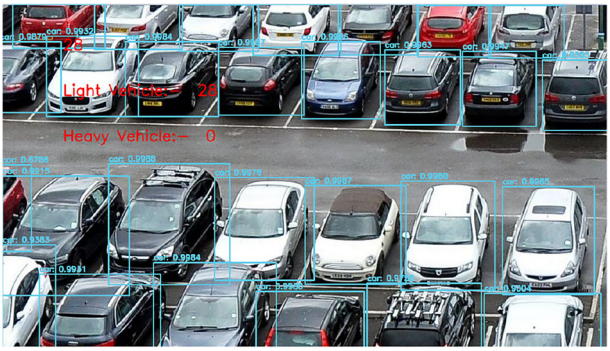

Fig. 24 Detecting and counting vehicles in an image(Result-2)

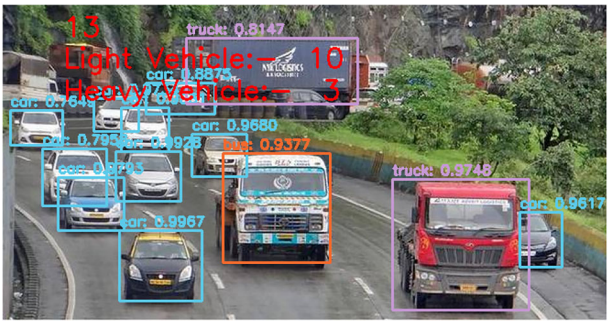

Fig. 25 Detecting and counting vehicles in an image(Result-3)

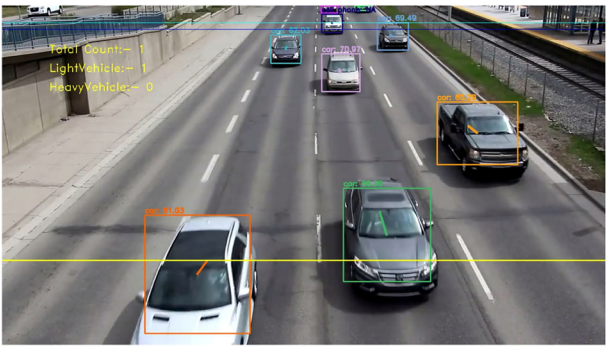

Fig. 26 Vehicles velocity detection (Result-1)

From the resulted figures and Table 2, it can be analyzed that the accuracy of the proposed model is quite good despite of complex structure of the images. Since, the proposed model works under different lighting conditions and offers convincing results. It is even capturing vehicles that are partially occluded.

4.2.3 Comparative analysis of vehicle detection

In this paper YOLOv5 is selected for vehicle detection. In order to show its efficacy, a comparison with other popular benchmark models can be seen in Table 3. The performance is measured in terms of precision, recall, F1 score and mean average precision (mAP). The average value of all categories of APs is defined as mAP, where AP is the integral value of the P–R curve. Precision, recall and F1 score are computed using (14)–(16).

$$Precision = \frac{TP}{TP + FP} \tag{14}$$

$$Recall = \frac{TP}{TP + FN} \tag{15}$$

$$Precision = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{16}$$

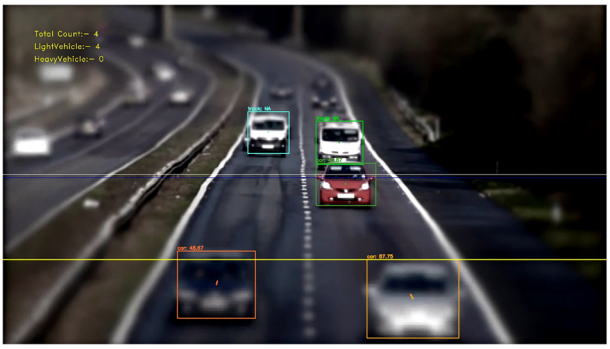

Fig. 27 Vehicles velocity detection (Result-2)

Table 3 A comparison of cutting-edge vehicle object detectors

Model	Precision (↑)	Recall (↑)	F1 score (↑)	mAP (↑)
SSD [34]	0.322	0.284	0.302	0.316
GhostNet-SSD [35]	0.351	0.308	0.328	0.322
FasterRCNN-ResNet50 [42]	0.407	0.428	0.417	0.423
FasterRCNN-VGG16 [17]	0.554	0.434	0.487	0.409
EfficientDet [50]	0.527	0.509	0.518	0.481
Improved YOLOv4 [29]	0.501	0.487	0.494	0.483
Improved YOLOv5l	**0.682**	**0.621**	**0.650**	**0.589**

Symbol ↑ indicates that the higher values are preferred

where TP, FP and FN represents true positives, false positives and false negatives respectively.

The following can be deduced from Table 3, where the best results are marked in bold:

- YOLOv5 algorithm has the best performance in terms of precision.
- In case of recall, YOLOv5 performs better.
- Since F1 score is the harmonic mean of precision and recall, YOLOv5 has outperformed the rest of the models.
- YOLOv5 also has the highest mAP value.

We have used a t-test to compare the performance of Improved YOLOv5l with six state-of-the-art algorithms at a significance level of 0.05. Initially, a hypothesis is made on the distribution similarity generated by the various vehicle detection algorithms. The comparison can be seen in Table 4. The results show that Improved YOLOv5l has outperformed in all the cases, as the results are positive and within the significance level. By clearing the t-test, it can be stated that Improved YOLOv5l is different from the six existing state-of-the-art vehicle detection algorithms.

4.2.4 Comparative analysis of vehicle tracking

Various object tracking benchmark models are evaluated on the MOT dataset using Multiple Object Tracking Accuracy (MOTA) and Multiple Object Tracking Precision (MOTA)

Table 4 t-test based comparison of state-of-the-art models with Improved YOLOv5l over benchmark evaluation functions

Models/improved YOLOv5l	Precision	Recall	F1 score	mAP
SSD [34]	+0.000885	+0.000946	+0.000916	+0.001168
GhostNet-SSD [35]	+0.000963	+0.001019	+0.00099	+0.001194
FasterRCNN-ResNet50 [42]	+0.00116	+0.001654	+0.001369	+0.001923
FasterRCNN-VGG16 [17]	+0.002497	+0.001707	+0.001956	+0.001773
EfficientDet [50]	+0.00206	+0.002855	+0.002421	+0.002961
Improved YOLOv4 [29]	+0.001763	+0.002384	+0.002047	+0.003017

(MOTP). MOTA and MOTP are calculated using (17) and (18).

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDS_t)}{\sum_t GT_t} \tag{17}$$

$$MOTA = 1 - \frac{\sum_{i,t} d_t^i}{\sum_t c_t} \tag{18}$$

where for the t-th frame, FN_t is the false negatives, FP_t is the false neagtives, IDS_t is the total number of times a trajectory changes its matched ground truth identity, and GT_t is the ground truth of detections. d_t^i is the distance in frame t between the predicted detection and the i-th ground truth detection, and c_t is the number of matched trajectories. MOTA displays the overall tracking accuracy, whereas MOTP displays the percentage alignment between the predicted bounding box and the ground truth.

The comparative analysis of vehicle tracking can be seen in Table 5, where the best results are marked in bold. The following can be deduced.

- Centroid tracking algorithm has the best Multiple Object Tracking Accuracy.
- Centroid tracking algorithm outperforms the other models in terms of Multiple Object Tracking Precision too.

4.2.5 Comparative analysis of traffic detection

We used the UCSD traffic video dataset for traffic detection. The experimental setup is similar to [29], with incident detection every 5000 frames, a speed limit of 90 km/h, and a congestion density threshold of 0.002 per frame. The camera's height is estimated to be 12 m, the lane line's length is 6 m, and the lane width is 3.75 m. The quantitative evaluation for traffic detection include Detection Rate (DR), False Alarm Rate (FAR), and Mean Time to Detect (MTTD) calculated using (19)–(21).

$$DR = \frac{\#\, detected\ traffic\ incidents}{\#\, incidents} \tag{19}$$

$$FAR = \frac{\#\, false\ alarms}{\#\, non\text{-}incidents\ detected} \tag{20}$$

$$MTTD = \frac{Total\ time\ elapsed\ between\ detecting\ incidents}{Total\ number\ of\ correctly\ detected\ incidents} \tag{21}$$

Table 5 A comparison of cutting-edge vehicle tracking algorithms

Model	MOTA (↑)	MOTP (↑)
MHT [41]	0.652	0.698
SORT [4]	0.794	0.728
DEEP-SORT [47]	0.813	0.775
MOTDT [10]	0.822	0.791
JDE [46]	0.843	0.802
Liang-MOT [29]	0.852	0.809
Centroid Tracking [43]	**0.881**	**0.815**

Symbol ↑ indicates that the higher values are preferred

Table 6 A comparison of cutting-edge traffic detection algorithms

Model	DR (↑)	FAR (↓)	MTTD (↓)
Liang et al. [29]	0.721	0.446	5.74
Proposed Model	**0.752**	**0.399**	**4.63**

Symbol ↑ indicates that the higher values are preferred, while the symbol ↓ indicates that the higher values are preferred

The comparative analysis of traffic detection can be seen in Table 6, where the best results are marked in bold. The following can be deduced.

– The proposed model outperforms the state-of-the-art model in terms of detection rate, false alarm rate and mean time to detect.

4.2.6 Velocity detection and over speeding vehicle identification

This subsection is dedicated to detect the velocity of the vehicles from the video output. To do this, we took various input videos to detect the velocity of vehicles. These streamed videos were fed into the model, the resulted estimated velocity of each vehicle is written on the bounding box itself. Few samples can be visualized in Figs. 26, 27 and 28. Even the vehicles that are blurred, the proposed model is able to capture its velocity. Based on the velocity calculations, a threshold pertaining to the speed limit is decided to identify the over-speeding vehicles. As an example, one such over-speeding vehicle can be seen in Fig. 29. Further the number plate detection algorithm is applied over the bounding box of the over-speeding car to identify the vehicle's registration number, as shown in Fig. 30.

5 Conclusion and future work

In natural situations, object detecting technology has advanced significantly. Images of traffic, however, differ from those of natural settings. Standard object detection techniques would not produce the expected result when applied directly to these photos. In this study, we provide a YOLOv5l-based lightweight vehicle identification model. The original

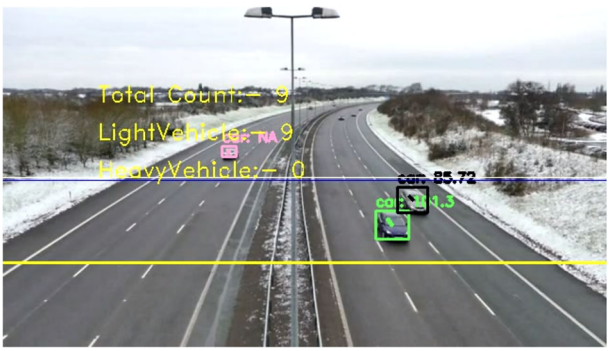

Fig. 28 Vehicles velocity detection (Result-3)

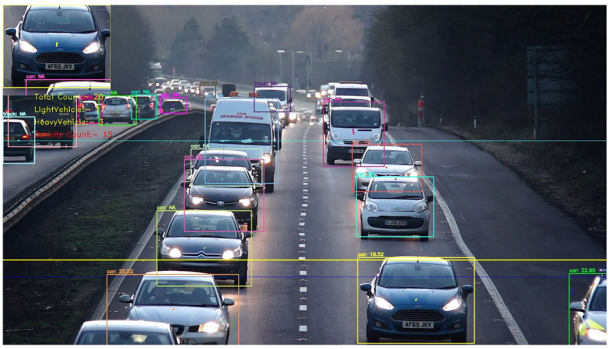

Fig. 29 Capturing over-speeding vehicles

YOLOv5l network has undergone some changes. We have increased an item detection scale to enhance the effectiveness of small object detection. While the GhostBottleneck module is expanded to concentrate on shallow feature information, the BottleneckCSP module is replaced with the GhostBottleneck module to lighten the model. This paper presents a productive approach to build the proficiency of the general recognition and tracking of vehicles, in actuality, circumstances. The proposed technique is tried on a different contribution under various climate caught by traffic observation camera. The proposed improvement in YOLOv5l is evaluated over three areas: vehicle detection, vehicle tracking and traffic detection. From experimentation it can be seen that the proposed model has the best values of precision, recall, F-score and mAP for vehicle detection over BDD100K image data set. The centroid tracking algorithm when combined with the improved YOLOv5l algorithm has the best multiple object tracking accuracy and multiple object tracking precision over BDD100K MOT data set. The proposed model performed well over UCSD traffic video data set in terms of detection rate, false alarm rate, and mean time to traffic detection.

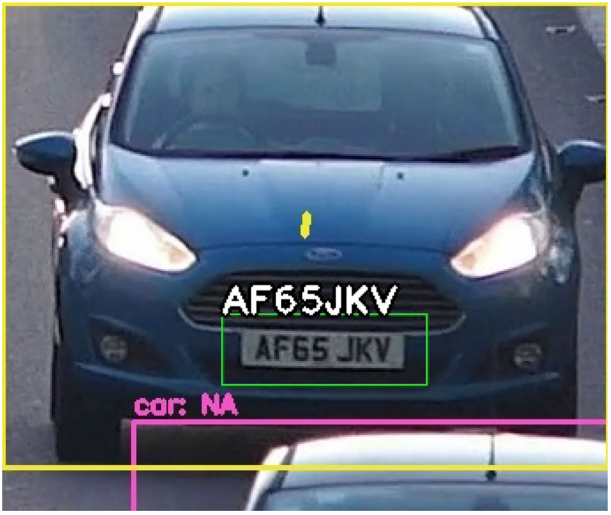

Fig. 30 Number plate detection of over-speeding vehicle

The proposed framework can be utilized for ongoing execution to identify track and check the quantity of vehicles that are available in the given information video or can be actualized on live stream of traffic. The framework is responsive in a few rush hour gridlock condition, and the outcomes with respect to vehicle recognition and following are palatable. Nonetheless, the exhibition of the framework straightforwardly relies on the nature of the info video and the camera position. It can be improved to provide better accuracy for detection and counting number of vehicles. It can be improved to detect the collision impact during any accident on road. It can be enhanced to detect the speed of vehicle more accurately.

Acknowledgements The authors express their gratitude to Department of Science & Technology (DST), India for the obtained financial support in performing this research work. This work is one of the outcomes of the project entitled "IoT based real time and effective traffic signal scheduling for smart city" with sanction no. DST/ICPS/CPS-Individual/2018/678(G) dated 03.01.2019, sponsored by DST.

Data availability Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Al-Smadi M, Abdulrahim K, Salam RA, Manikandan R, Ramakrishnan R (2016) 2013. Video object extraction by using background subtraction techniques for sports applications. Digit Image Process 5(9):435–440
2. Al-Smadi M, Abdulrahim K, Salam RA (2016) Traffic surveillance: a review of vision based vehicle detection, recognition and tracking. Int J Appl Eng Res 11(1):713–726
3. Badue C, Guidolini R, Carneiro RV, Azevedo P, Cardoso VB, Forechi A, Jesus L, Berriel R, Paixao TM, Mutz F, de Paula Veronese L (2021) Self-driving cars: a survey. Expert Syst Appl 165:113816
4. Bewley A, Ge Z, Ott L, Ramos F, Upcroft B (2016) Simple online and realtime tracking. In: 2016 IEEE International conference on image processing (ICIP). IEEE, pp 3464–3468
5. Bhalla A, Nikhila MS, Singh P (2020) Simulation of self-driving car using deep learning. In: 2020 3rd International conference on intelligent sustainable systems (ICISS). IEEE, pp 519–525
6. Byun S, Shin IK, Moon J, Kang J, Choi SI (2021) Road traffic monitoring from UAV images using deep learning networks. Remote Sens 13(20):4027
7. Cai Y, Wang H, Chen X, Jiang H (2015) Trajectory-based anomalous behaviour detection for intelligent traffic surveillance. IET Intell Transp Syst 9(8):810–816
8. Chan AB, Vasconcelos N (2005) Classification and retrieval of traffic video using auto-regressive stochastic processes. In: IEEE Proceedings. Intelligent vehicles symposium. IEEE, pp 771–776
9. Chen S (2019) Multimedia for autonomous driving. IEEE Multimed 26(3):5–8
10. Chen L, Ai H, Zhuang Z, Shang C (2018) Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In: 2018 IEEE International conference on multimedia and expo (ICME). IEEE, pp 1–6
11. Chen S, Chen Y, Zhang S, Zheng N (2019) A novel integrated simulation and testing platform for self-driving cars with hardware in the loop. IEEE Trans Intell Veh 4(3):425–436
12. Dai J, Li Y, He K, Sun J (2016) R-fcn object detection via region-based fully convolutional networks. In: Advances in neural information processing systems, pp 379–387
13. Daily M, Medasani S, Behringer R, Trivedi M (2017) Self-driving cars. Computer 50(12):18–23
14. Das TK, Tripathy AK, Mishra AK (2017) Optical character recognition using artificial neural network. In: 2017 International conference on computer communication and informatics (ICCCI). IEEE, pp 1–4
15. Fekri-Ershad S, Tajeripour F (2017) Impulse-noise resistant color-texture classification approach using hybrid color local binary patterns and Kullback–Leibler divergence. Comput J 60(11):1633–1648

16. Ferryman JM, Worrall AD, Sullivan GD, Baker KD (1995) A generic deformable model for vehicle recognition. In: Proceedings of the British machine vision conference 1995. British Machine Vision Association. https://doi.org/10.5244/c.9.13
17. García-González J, Molina-Cabello MA, Luque-Baena RM, Ortiz-de-Lazcano-Lobato JM, López-Rubio E (2021) Road pollution estimation from vehicle tracking in surveillance videos by deep convolutional neural networks. Appl Soft Comput 113:107950
18. Ghazal B, ElKhatib K, Chahine K, Kherfan M (2016) Smart traffic light control system. In: 2016 Third international conference on electrical, electronics, computer engineering and their applications (EECEA). IEEE, pp 140–145
19. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: 2014 IEEE conference on computer vision and pattern recognition. IEEE. https://doi.org/10. 1109/cvpr.2014.81
20. Greenblatt NA (2016) Self-driving cars and the law. IEEE Spectr 53(2):46–51
21. Han D, Leotta MJ, Cooper DB, Mundy JL (2006) Vehicle class recognition from video-based on 3d curve probes. In: 2005 IEEE International workshop on visual surveillance and performance evaluation of tracking and surveillance. IEEE. https://doi.org/10.1109/vspets.2005.1570927
22. Han K, Wang Y, Tian Q, Guo J, Xu C, Xu C (2020) Ghostnet: more features from cheap operations. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1580–1589
23. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969
24. Howard A, Sandler M, Chu G, Chen LC, Chen B, Tan M, Wang W, Zhu Y, Pang R, Vasudevan V, Le QV (2019) Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 1314–1324
25. Jocher G, Nishimura K, Mineeva T, Vilariño R (2020) YOLOv5. GitHub repository: https://github.com/ultralytics/yolov5
26. Kanopoulos N, Vasanthavada N, Baker RL (1988) Design of an image edge detection filter using the Sobel operator. IEEE J Solid-State Circ 23(2):358–367
27. Kim JH, Huh JH, Jung SH, Sim CB (2021) A study on an enhanced autonomous driving simulation model based on reinforcement learning using a collision prevention model. Electronics 10(18):2271
28. Li J, Bao H, Han X, Pan F, Pan W, Zhang F, Wang D (2017) Real-time self-driving car navigation and obstacle avoidance using mobile 3D laser scanner and GNSS. Multimed Tools Appl 76(21):23017–23039
29. Liang H, Song H, Yun X, Sun S, Wang Y, Zhang Z (2022) Traffic incident detection based on a global trajectory spatiotemporal map. Complex Intell Syst 8(2):1389–1408
30. Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft coco Common objects in context. In: European conference on computer vision. Springer, Cham, pp 740–755
31. Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2117–2125
32. Liu CM, Juang JC (2021) Estimation of lane-level traffic flow using a deep learning technique. Appl Sci 11(12):5619
33. Liu Y, Yao L, Shi Q, Ding J (2014) Optical flow based urban road vehicle tracking. In: 2013 Ninth international conference on computational intelligence and security. IEEE. https://doi.org/10.1109/cis.2013.89
34. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssd: single shot multibox detector. In: European conference on computer vision. Springer, Cham, pp 21–37
35. Liu J, Cong W, Li H (2020) Vehicle detection method based on ghostnet-SSD. In: 2020 International conference on virtual reality and intelligent systems (ICVRIS). IEEE, pp 200–203
36. Mou L, Xie H, Mao S, Zhao P, Chen Y (2020) Vision-based vehicle behaviour analysis: a structured learning approach via convolutional neural networks. IET Intell Transp Syst
37. Nepal U, Eslamiat H (2022) Comparing YOLOv3, YOLOv4 and YOLOv5 for autonomous landing spot detection in faulty UAVs. Sensors 22(2):464
38. Park K, Lee D, Park Y (2007) Video-based detection of street-parking violation. In: International conference on image processing. https://www.tib.eu/en/search/id/BLCP
39. Qiu-Lin LI, Jia-Feng HE (2011) Vehicles detection based on three-frame-difference method and cross-entropy threshold method. Comput Eng 37(4):172–174
40. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
41. Reid D (1979) An algorithm for tracking multiple targets. IEEE Trans Autom Control 24(6):843–854

42. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks
43. Simple object tracking with OpenCV. https://www.pyimagesearch.com/2018/07/23/simple-object-tracking-with-opencv/ accessed on 17th June 2022
44. Song H, Liang H, Li H, Dai Z, Yun X (2019) Vision-based vehicle detection and counting system using deep learning in highway scenes. Eur Transp Res Rev 11(1):1–16
45. Uijlings JRR, van de Sande KEA, Gevers T, Smeulders AWM (2013) Selective search for object recognition. Int J Comput Vis 104(2):154–171
46. Wang Z, Zheng L, Liu Y, Li Y, Wang S (2020) Towards real-time multi-object tracking. In: European conference on computer vision. Springer, Cham, pp 107–122
47. Wojke N, Bewley A, Paulus D (2017) Simple online and realtime tracking with a deep association metric. In: 2017 IEEE International conference on image processing (ICIP). IEEE, pp 3645–3649
48. Yang H, Qu S (2017) Real-time vehicle detection and counting in complex traffic scenes using background subtraction model with low-rank decomposition. IET Intell Transp Syst 12(1):75–85
49. Yu F, Chen H, Wang X, Xian W, Chen Y, Liu F, Madhavan V, Darrell T (2020) Bdd100k: a diverse driving dataset for heterogeneous multitask learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2636–2645
50. Zhang Y, Li X, Zhang Z, Wu F, Zhao L (2015) Deep learning driven blockwise moving object detection with binary scene modeling. Neurocomputing 168:454–463
51. Zhao ZQ, Zheng P, Xu ST, Wu X (2018) Object detection with deep learning: a review. arXiv:1807.05511
52. Zhu L, Yu FR, Wang Y, Ning B, Tang T (2018) Big data analytics in intelligent transportation systems: a survey. IEEE Trans Intell Transp Syst 20(1):383–398

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.