RESEARCH ARTICLE

INTERNATIONAL RESEARCH JOURNAL OF MULTIDISCIPLINARY TECHNOVATION

Asian Research Association

DOI: 10.54392/irjmt2459

Deep Learning based Road Traffic Assessment for Vehicle Rerouting: An Extensive Experimental Study of RetinaNet and YOLO Models

Anand John [a, *], Divyakant Meva [b], Nidhi Arora [c]

[a] Department of Computer Applications, Christ College, Rajkot 360005, Gujarat, India
[b] Faculty of Computer Applications, Marwadi University, Rajkot 360003, Gujarat, India.
[c] Department of Computer Science, Kalindi College, University of Delhi, Delhi 110008, India
* Corresponding Author Email: anand_john@yahoo.com
DOI: https://doi.org/10.54392/irjmt2459
Received: 19-03-2024; Revised: 14-09-2024; Accepted: 20-09-2024; Published: 23-09-2024

Abstract: Currently, there is a notable prevalence of substantial traffic congestion and frequent vehicular accidents on roadways in contemporary times. Amalgamation of latest front-line technologies involving Internet of Things (IoT) and image classification has immense potential to advance the progress of a proficient traffic regulation system. To mitigate the occurrence of vehicular accidents, our research endeavors revolve around the comprehensive analysis of the prevailing road conditions. This meticulous examination allows us to effectively automate traffic routes orchestration, ensuring smooth vehicular movement across all lanes of the road network. The analysis of traffic patterns is conducted through the utilization of visual data images. The real time captured traffic images undergo processing using various object detection models named RetinaNet and the YOLO (You Only Look Once) models. A series of comparative evaluations suggests an improved traffic object identification capacity for the RetinaNet model as compared to all YOLO models.

Keywords: Road Traffic, RetinaNet, YOLO models, Object detection, IoT, Convolution Neural Network (CNN)

1. Introduction

The pursuit of real-time object recognition in images has garnered considerable interest from researchers and technical professionals worldwide in recent times. The scenario can be attributed to the application of high end computing strategies of Machine as well as Deep Learning to the field. Deep Learning specifically has shown amazingly accurate results in the automation of object by making machines self-train and develop highly flexible, dynamic, complex and self-learnable models [1]. The area has been successfully applied in image categorizing, challenges in image categorizing and object detection to aid administration in taking real time decisions. It has opened new possibilities for many applications [2-3]. As the applications of object discovery are increasing, the requirement for smaller models that can be installed on mobile and embedded systems are going to grow rapidly [4-6]. A problem detection method is proposed by integration of intelligent Convolutional Neural Networks based deep learning techniques with Kalman Filter based Gaussian-mixture model to improve cyber physical systems security, protection of actual material and finds cyber security attack in smarter way. It gives a mixed model describing the risk probabilities in Cyber Physical Systems [7]. Moreover, there are various challenges in utilizing deep learning models for IoT (Internet of Things) security using big data technologies. These pose a challenge for additional research in IoT security features for better data handling as IoT devices generate large sizes of data [8].

The Australian Center for Cyber Security has built the UNSW-NB15 dataset which contains complex and modern data illustrations of risks grounded on cyber security. An invasion discovery technique is proposed that uses predictive machine learning models using weight based learning model in a network traffic scenario. The risk data is found from IoT devices, gets confirmed as risk data by getting checked through the device known as fog computing and finally the actual data is sent to cloud storage, where the model is trained on the actual data to improve the accuracy. Similarly, it reduces the hardware usage and the usefulness is enhanced. It helps in implementing the development of real time invasion discovery system in smart traffic management system and hence it gives way to additional research in network invasion discovery system [9]. The usage of fog computing has increased speedily due to its features of handling network bottleneck and absence of regional independence.

However, fog computing has limitations of certain privacy data problems and inefficiency to deal with corrupting attacks in the network. To solve the problems, the block-chain based secure federated learning in traffic system is proposed to find the traffic flow in urban development and decrease the breach. The efficiency of traffic flow discovery is improved and more exploration is needed to augment the improvement in reducing the problems of discretion and safety of data [10]. Since deep learning has revealed promising outcomes in many areas of object discovery, its applicability to traffic management is significant in nature, as real time traffic volume prediction can help in suggesting rerouting and avoid accidents. Deep learning methods are applied in traffic network management system to supervise and investigate the traffic by classification and discovery of vehicles. But there are limitations of deep learning methods: - in making deep learning methods learn different network, it takes considerable time and large amount of storage is used [11]. GluonCV and GluonNLP Toolkit are used to process state of the art (SOTA) deep learning methods in picture discovery built on Apache server. It also encourages constant research. These two toolkits can be installed on any platform and with diverse software programing languages. These toolkits are executed in python and provide numerous opportunities to implement new ideas with deep learning [12]. A deep reinforcement learning-based improved traffic control system is suggested. This method aids in the formulation of better decision-making strategies by calculating the length of the vehicle queue and identifying other traffic flow characteristics. The limitations found in this method is that some random study gives insignificant output [13]. Deep learning methods has limitations that they have convergence problem, as it builds a straight connection between each traffic-matrix and routing strategy. It is suggested to join transformer with deep reinforcement learning to address the convergence issue and improve traffic-matrix performance in software-defined networking in real time altering traffic conditions. A multi characteristic embedding segment is designed to attain positional encodings in the transformer model. Good routing outcomes are produced using transformer-based deep learning after the convergence problem is improved [14]. However, researchers have found that in some environmental locations, identifying a vehicle in a picture or video can be exceedingly challenging. One of the most essential objectives for effective traffic management is to manage traffic routes in real time, avoiding accident scenes and heavy traffic chocked roadways, to make life easier, save time, and reach perfection. To achieve this, the vehicle mass in the traffic is checked by recognizing and locating the vehicle from the traffic scenes or videos. The area involves techniques of computer vision as well as deep learning. Computer vision enabled deep learning frameworks work towards minimizing human efforts while emulating the direction and technique of natural eyes. Usually, traffic lights at each intersection operate according to the times set for traffic on each side by electronic devices. The primary problem is that even if there is no traffic on one route, traffic from other routes must stay until its traffic signal time becomes nil. A method can be built which removes the waiting period after discovering the total vehicles while doing traffic supervision. Correspondingly, its advantage is that the police do not have to intervene in the city's roads, and traffic supervision can be automated effectively. One significant issue with traffic management is breach of traffic which is the major causes behind accidents. To reduce this, vehicle identification methods can be used to find wrong side vehicles. By inspecting various vehicles on the road, it is possible to determine the traffic situation and location and provide the precise details to manage traffic with the well-organized methods.

1.1 Motivation for Conducted Research

In current research being done for traffic based object detection, researchers have presented the application of specific deep learning based models to this area. However, a need for detailed empirical investigation of the available wide range of deep learning architectures is required using a common traffic image dataset. The research effort presented in this paper therefore is a step in this direction to conduct an extensive evaluation of five well-known deep learning architectures on a common traffic image dataset. In the experimental evaluation conducted in this paper, we propose to use live traffic facts from CCTV cameras and extract sensitive facts such as vehicle types, traffic density from live traffic data. For this purpose, 300 live traffic images were collected from the different locations of Rajkot (Gujarat, India) in the early months of 2023. Various traffic light timings are assigned depending on the density of that definite road. Various deep learning object detection models such as four “You-Only-Look-Once (YOLO)” and RetinaNet models are trained on the real time collected traffic images. All these algorithms are executed in the tensorflow platform having keras module to find the effective vehicle identification. Outwardly, we narrate the exactness as well as practice of several algorithms like “RetinaNet”, “YOLO - v5”, “YOLO - v6”, “YOLO - v7” and “YOLO - v8” for identifying vehicles after processing the photographs in these algorithms. The conducted in-depth comparative analysis for several significant YOLO models and RetinaNet model on a common dataset will be beneficial for researchers to examine all algorithms irrationally.

Organization of Paper: Related work is presented in next section 2; Methodologies followed are defined in section 3 trailed by the demonstration of Results and Discussion in section 4; and Conclusion of performed research work is presented in section 5 followed by references.

2. Related Work

An innovative traffic management system was developed to quantify and predict traffic flows as well as to classify types of vehicles. Diverse vehicle discovery methods such as YOLO models are examined and comparison of yolo models is made. The YOLO models examined and compared were YOLOv3, YOLOv4, YOLOv5, YOLOv6 and YOLOv7 specifically for vehicle discovery to develop an effective traffic management system. The researcher specifies that yolo models have limitations. Small vehicles are not discovered using yolo models. For traffic mass calculations and vehicle identifications, a counting approach using YOLOv4 small architecture and was given by Cheng-Jian Lin *et al*. [15-16]. In the paper, it was projected that by using the CFNN (Convolutional Fuzzy Neural Network) model along with the fusion method increased the accuracy of vehicle identification and location. It emphasized that YOLO-CFNN have a high mAP rate, and real time vehicle accurate counting and categorization features. In another study conducted by Yi-Qi Huang *et al.* [17], a regression technique was developed to find out the total number of vehicles in a traffic environment using YOLO v3 model. In YOLO model, first pictures are taken from present road areas, then it is processed through the YOLO structure. The pictures are separated into a grid of 3x3 matrices. Consequently, each grid classifies and locates the vehicle objects. YOLO finds the confidence score for each grid. The confidence or assurance score is one, if the precise vehicle is identified in the box area and zero if the precise vehicle is not identified in the box area. Likewise, the preciseness is increased to identify vehicles by using anchor boxes. In this paper also it was presented that YOLO v3 model has some limitations and could not discover some objects. The results depicted in the paper suggested that YOLO v3 model can be improved for vehicle identification and traffic analyses by using YOLOv3-DL deep neural network. The network augments the feature extraction method thereby effectively reducing intersection over union loss error and hence improves the YOLOv3 model.

Traffic density is found depending on the discovery of vehicles and its total as shown in the paper [18]. The vehicle discovery method is enhanced by using four distinct models. When the experiment is compared to other models, the accuracy is shown higher. Better techniques can be used in future research to yield better outcomes and a faster inference time.

Luyang Zhang *et al.* [19] demonstrated the performance of an improved RetinaNet model by replacing the CNN with an octave convolution for vehicle discovery. The presented method was effective in increasing the preciseness for the discovery of different vehicles and displayed steady benefits over low resolution issues. In another study by Azizi Abdullah *et al.* [20], effectiveness of tiny-YOLO was examined. Comparative analysis concluded that YOLO was better model for finding vehicles rather than YOLO tiny. Further, Abuelgasim Saadeldin *et al.* [21] in their study explained that vehicle counting method used by YOLOv5n with deep sort method helped to generate accurate and efficient results for vehicle discovery in the traffic environment to manage present time situations. In another study by Yidan Chen *et al.* [22], vehicle discovery by applying deep learning methods such as YOLOv3 and SSD under actual rush-hour situations is explained. It implies that there is still opportunity for improvement and that a wide range of additional techniques can be made accessible for use in vehicle discovery.

Yu Zhang *et al.* [23], in their paper, stated that the highway traffic scene was recorded in different areas of the road to find strong revelation of vehicle discovery where the vehicles are moving at a very high speed. They used an improved YOLOv5 model for vehicle detection. Using this model, the flip-mosaic data improvement technique was calculated, that meaningfully increased the detection of smaller vehicle size displayed in the video. There were few samples used in this investigation. Pictures with better lighting can be use, and different weather situations need to be further explored and researched. In some other studies by Cheng-Jian Lin *et al*. [24], a real-time traffic control system based on YOLOv4, Gaussian mixture model, and virtual detection zone is suggested for finding the vehicle, counting the vehicles in the video, finding the speed of the vehicle, and in another studies by Alexey Bochkovskiy *et al.* [25] effectiveness of YOLOv4 model is used for finding the objects accurately. Also, comparison of different models is done showing their accuracy and performance. It suggests further enhancement in the study for work in future.

The conducted literature survey has depicted the application of only selected specific YOLO algorithms in different traffic object detection experiments by different researchers. However, there is a need for an in-depth comparative analysis of varied YOLO models and RetinaNet model on a single dataset, to enable the stakeholders to examine all algorithm's performance for a common set of traffic data. Therefore, to substantiate research in traffic object detection, this paper presents in-depth experimental evaluations of 4 major YOLO models and RetinaNet model along with their comparative analysis.

3. Methodology

Single shot detection (SSD) algorithm does a single process in the input pictures to do object discovery. It processes the whole picture in one execution only. But SSD is less accurate than other models and can only be used for less memory requirement. Despite using CNN as their primary architecture, Faster R-CNN and YOLO have distinct frameworks. The region proposal classification network

(Faster R-CNN), makes several predictions and performs detection on different regions. However, YOLO only makes one prediction for each input and just half of the errors, which allows it to retain a high average precision.

3.1 RetinaNet Architecture

One of the highest one-stage object discovery algorithms, RetinaNet is acknowledged for performing well with dense and tiny objects. It is a widespread object discovery model which is mostly implemented for object detection using aerial as well as satellite images. Figure 1 depicts the architecture of RetinaNet model.

During training, focal loss function is used to solve class imbalance issues. RetinaNet contains one main backbone network and additionally two subnetworks. The backbone called as self-convolutional network, calculates the feature map for a given captured picture.

On the output of the backbone, the initial subnet carries out convolutional object categorization, while another subnet carries out convolutional bounding box regression. There are four major factors of a RetinaNet model architecture as revealed in Figure 1:

- Bottom-up direction: - The Residual Neural Network (ResNet) which analyses the feature maps at unlike measures, irrespective of the picture size.
- Top-down direction and adjacent networks:- The top-down direction increases the sampling rate in the feature maps from advanced pyramid planes, and the lateral networks combines the top and bottom layers with the same area size.
- Grouping sub-network:- It finds the object and shows the location for every anchor box and the item class.
- Regression sub-network: - It decrease the loss by adjusting the values of bounding boxes with the anchor boxes to discover the real object.

The featured pyramids constructed upon image pyramids are known as featured image pyramids. Deep learning is used to distinguish objects of various sizes in any pictures, and makes use of the featured image pyramids. RetinaNet utilizes a focal loss function which is a firm sized cross entropy loss, whereas the confidence in the proper class grows the size deteriorate to zero. When the picture is passed through deep ResNet, the feature is initial taken out, and afterwards the feature pictures of various sections are combined by feature pyramid network (FPN) bottom-up, top-down connection and horizontal connection, and finally directed to categorization and subnetwork.

3.2 YOLOv5 Architecture

YOLOv5 architecture is depicted in Fig. 2, where Cross-stage partial network (CSP) filters the main layer and extracts the necessary information from it to increase accuracy and reduce restrictions. It has three significant sections. They are backbone, neck and head. The backbone is made up of CSPDarknet structure, which is a new version of darknet as shown in Figure 2.

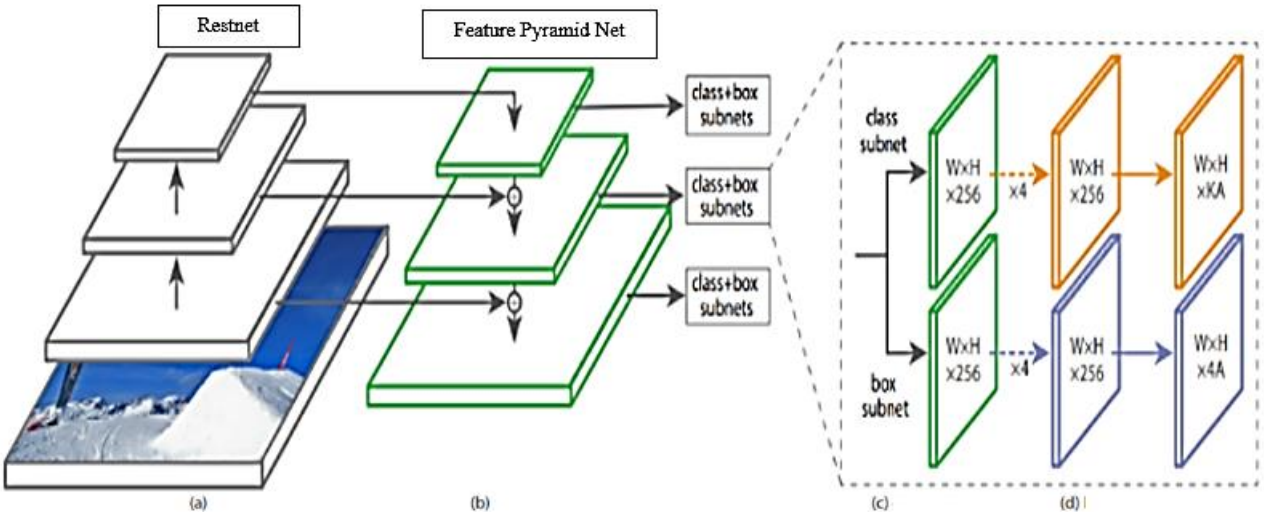

Figure 1. RetinaNet architecture (source:https://developers.arcgis.com/python/guide/images/RetinaNet.png) (a) ResNet, (b) feature pyramid net, (c) class subnet (top), (d) box subnet (bottom)

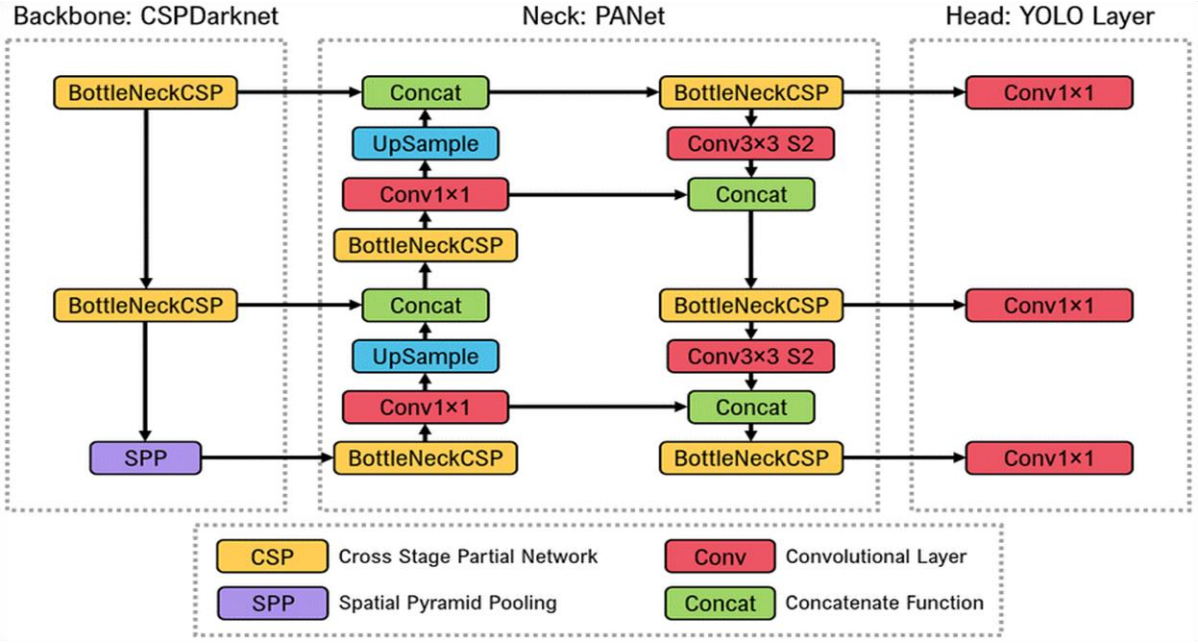

Figure 2. YOLOv5 network architecture [26] (source: DOI: 10.48550/arXiv.2205.11830).

The head finally generates the output in the YOLOv5 structure. The YOLOv5 model reduces excess data extraction and enhances generalization by utilizing several data augmentation strategies. These techniques include: mosiac growth, copy-paste growth, random affine transformation, mix-up growth, albumentations library, HSV growth and random horizontal flip. In YOLOv5, three methods are used to calculate the loss.

1. Classes loss, also called the binary entropy loss, determines the error for the categorization work
2. Object loss, additional binary entropy loss, calculates the inaccuracy for an object's occurrence in the lockup
3. Location loss, or Entire IoU (Intersection over Union) loss, finds the inaccuracy for localizing the object within the area.

The following equation represents the general loss function:

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc}$$

PANet (Path Aggregation Network) carries out a characteristic pyramid network that comprises dissimilar layers holding hierarchical layers. The characteristic pyramid network classifies the vehicles shown in the photo. The work of head is to achieve object discovery which comprises box values: width, length, object name, and object likelihood. Finally, YOLO layer is generating the output [27].

The three layers of YOLO-v5 are combined into a single layer. Spatial pyramid pooling is an aggregation layer that takes randomly sized pictures and converts them to appropriate size, to eliminate the problem of irregular sizes. The last three characteristic layers are identification segments. The neck has equally both spatial pyramid pooling area and PANet. This makes it easy to organize and add IoT devices easily [28]. YOLO version v5 architectures have several modifications depending on their sizes. They are YOLOv5n called nano, YOLOv5s called small, YOLOv5l called large and YOLOv5x called extra-large.

3.3 YOLOv6 Architecture

The YOLOv6 model, provided by Li. *et al*. [29], seamlessly represents the COCO data standard and has 295 layers as shown in Figure 3. YOLOv6 model is a single stage method for discovering objects for diverse applications. YOLOv6 performs object discovery using several weights such as YOLOv6n.pt, YOLOv6s.pt and YOLOv6t.pt.

YOLOv6 implements the structure directly in python. YOLO v6 is especially motivated for industrial use with improved performance and exactness. Similarly, like the previous YOLOv5 model, YOLOv6 has three parts. They are mainly the backbone, neck and the head of the network. All these parts have some improved techniques. The backbone is doing the primary job of separating the characteristics of the object. Subsequently, the network's neck and head receive these properties for object discovery. The RepBlock, RepConv, and CSPStackRep modules that comprise the core of EfficientRep are utilized by YOLOv6. Because of the lower accuracy of ResNets, a reparametrized backbone that modifies the network anatomy during training and inference improves accuracy. The reparametrized VGG networks are used in the YOLOv6 nano, tiny, and small model version. The YOLOv6 architecture uses reparametrized variants of the CSP backbone for the medium and large model versions.

The path aggregation network (PAN) joins the properties from different reparametrized sections of the picture called as Rep-PAN. Efficient Decoupled Head, which divides the backbone differently and does not share any attributes, is used by YOLOv6. YOLOv6 has easy device responsive structure for support and neck filters the accurate characteristics. It also processes intersection over union loss to extend correctness of object identification. The YOLO v6 model needs two loss functions. They are varifocal loss (VFL) and distribution focal loss (DFL) along with box regression loss. Extracted information from samples are stable with the aid of VFL.

DFL is mainly used to treat distorted pictures. YOLOv6 replicates over the backbone and process Rep-PAN (Re-parameterized path aggregation network) neck as shown in Figure 3.

3.4 YOLOv7 Architecture

YOLOv7 was developed by researchers Chien-Yao Wang *et al.* [30] in July 2022, considering the storage needed to store the layers and the space needed to propagate the function through the layers. Figure 4 represents YOLOv7 design. This model differs from the preceding YOLO models in that it is the fastest and most accurate model. The picture's portions are distinguished by a backbone. Subsequently, characteristics of the picture is joined in the neck segment and processed by the head to find the objects in the picture. The model absorbs effectively fast with fewer features. The final merger of the layer chosen is the 'Extended Efficient Layer Aggregation Network (E-ELAN)', which is a progressive version of the ELAN computing part.

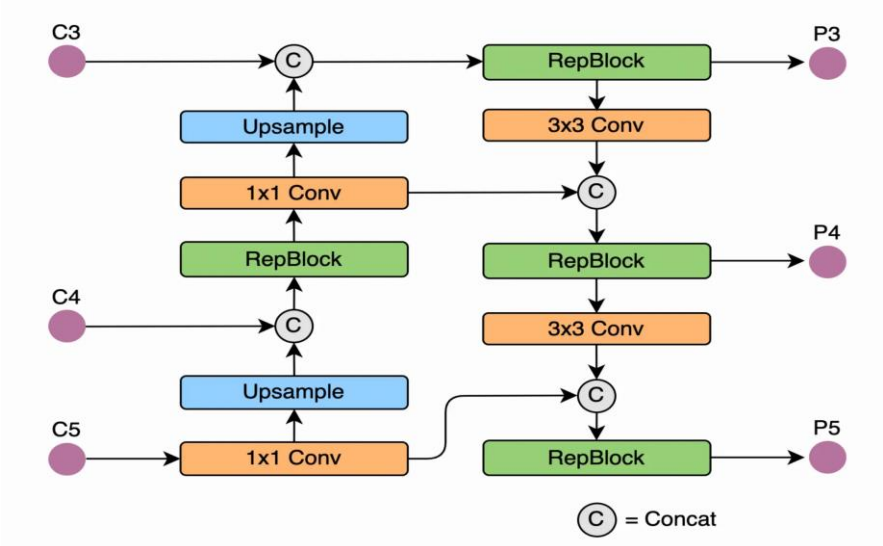

Figure 3. YOLOv6 Rep PAN Neck (source: https://blog.roboflow.com/yolov6/).

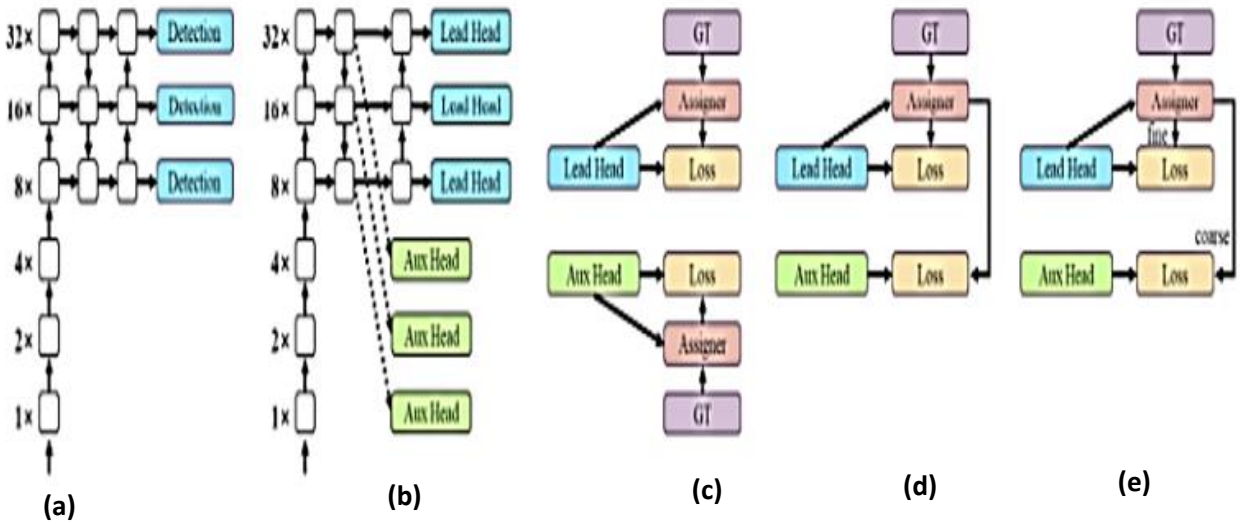

Figure 4. Architecture of YOLOv7 (source: 'https://blog.roboflow.com/yolov7-breakdown/') (a) Depiction of Normal model (b) Depiction of Model with auxiliary head (c) Depiction of Independent assigner (d) Depiction of Lead guided assigner (e) Depiction of Coarse-to-fine lead guided assigner.

YOLOv7 training has paths to see how system segments receive characteristics data. Moreover, YOLO version-7 tiny model type is used to assess a lesser group of images. Additionally, the trainable bag of freebies technique improves the model's performance without raising the cost of training. A post training approach called Re-parameterization is utilized, that helps the model perform better. Through this approach training period is extended however better inference outcomes are observed. Two main re-parametrization methods used are Model level and Module level ensemble re-parametrization methods. The auxiliary head is the head that helps in the center. The lead head does the object discovery in this model. The labels are created depending on the object discovery. Loss is computed using both the lead head and the auxiliary heads. As shown in the Fig. 4 (d) two groups of labels are created: a set of coarse labels for training the auxiliary head and a set of quality labels for training the lead head. YOLOv7 performance is considered better than all the preceding models with highest accuracy.

3.5 YOLOv8 Architecture

YOLOv8 model was created by 'Ultralytics'. The model is a leading-edge model that is carried forward from the accomplishments of earlier YOLO models while adding new features and enhancements to increase adaptability. YOLOv8 uses a single shot multi-box detector (SSD) algorithm, which discovers the bounding boxes and likelihoods of a category for an object in a single advancing pass. The YOLOv8 architecture is grounded on the ResNet-50 backbone. With the COCO dataset, the ResNet-50 backbone is then trained to discover objects in 80 dissimilar types. The SPP-YOLO (Spatial Pyramid Pooling YOLO) model architecture is another new feature of YOLOv8. It's the amalgamation of a ResNet-50 construction, Mosaic data growth, class-precise anchor boxes, and SPP-YOLO. The backbone of YOLOv8 has many convolutional layers arranged in an orderly way to extract characteristics from the picture [31]. YOLOv8 is an excellent choice for real-time object discovering applications because it is highly effective and precise. This model makes it a best executing model in the arena of object discovery.

4. Experimental Results and Comparison

Around 300 pictures have been collected from actual traffic situations from the roads of Rajkot (Gujarat, India) during the early months of 2023. The traffic flow scenes are verified for vehicle finding depending on the COCO labels being applied in all YOLO and RetinaNet models. The inference program available in all YOLO models can be used to send a set of photographs to a selected folder where the YOLO model will make its predictions, i.e., the outcomes are saved in the selected folder. While executing the inference, we can pass the source information, YOLO weights, picture size, configuration threshold value, and intersection over union threshold value and the augmented inference name. The annotation text file per picture is created in the YOLO models. Each file includes bounding box details for every vehicle discovered in the picture. The vehicles are counted and it is presented by applying the programming of python language. The confidence threshold is made about 0.25. The picture resolution of all YOLO models is processed as 640*640. The dataset consists of 300 images. The dataset strength is determined by the speed and accuracy of the object detection model. Since deep learning is the newest field of study and development, it is the method of choice chosen for object finding. The restriction in the models is that as we give different epochs value and confidence score, the result may tend to vary. The experimental results found out may not be as accurate and feasible, so there is a way for further research in traffic management. The purpose of implementing this research is to solve the problems of the urban city in controlling, managing the traffic and helping the society to have a safe environment.

4.1 RetinaNet Experimental Results

In RetinaNet, all 300 pictures are passed in batch for processing. The vehicles like motorcar, motorbike, bus, and truck are checked using the model by applying the confidence score of about 0.40 for discovering the vehicles. After the pictures were tested using RetinaNet model, the vehicles were discovered and counted as shown in traffic scenes in Figures 5(a-b). The confidence scores set for RetinaNet is given in Table 1. Vehicle finding ratios and counting by RetinaNet for scene 1 and 2 are shown in Table 2.

4.2 YOLOv5 Experimental Results

In YOLOv5 system, we executed the YOLOv5s.pt version in Tensorflow for object detection. By running this inference program, the detected objects in tensorflow platform are stored in a folder. A python file named detect.py inbuilt in YOLOv5, is then used to detect the objects identities. The YOLOv5 model is tested using PyTorch Hub inferences with an epoch of 300. Figure 6 (a-b) depicts sample scenes 3 and 4 of object detection by YOLOv5 model. Table 3 lists the configuration settings used for YOLOv5. Table 4 lists the vehicles finding ratios and counting by YOLOv5 for both scenes. As shown in Table 4, the output number of motorcars, trucks and motorbikes are displayed. Subsequently, a vehicle of little size in the photographs can be revealed when the confidence score is above 0.25. The inference is implemented and the results are transferred to a text file. After checking the file, the total count of vehicles for every picture is calculated.

Table 1. Confidence Score for Traffic Scene1 and Traffic Scene 2

Configuration done in Traffic scene1		**Configuration done in Traffic scene2**	
Confidence score	0.40	Confidence score	0.40

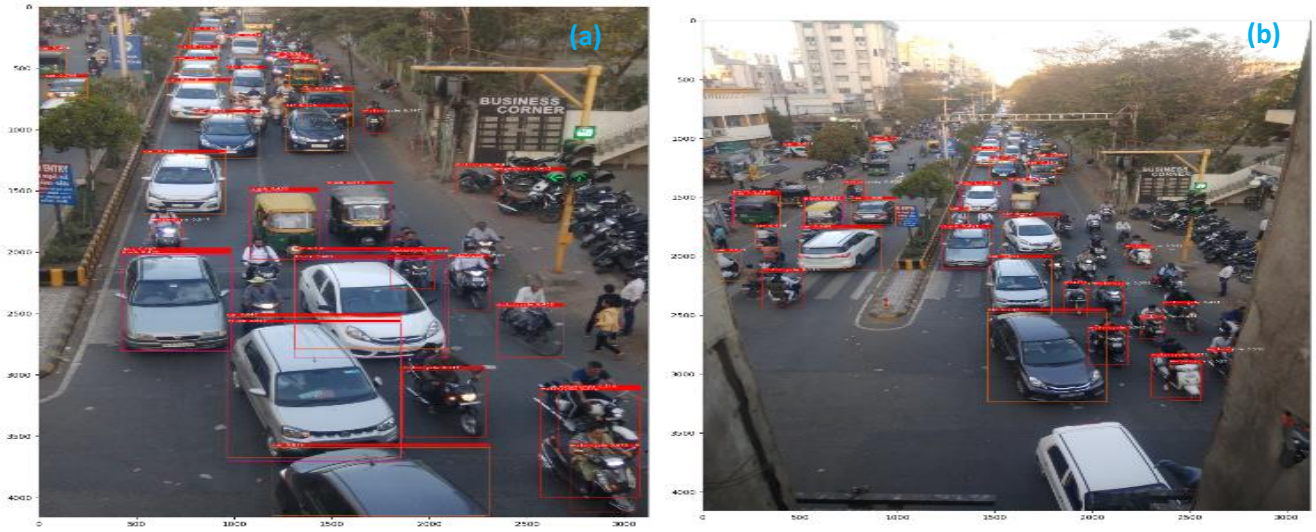

Figure 5. Vehicle finding by RetinaNet: (a) Traffic scene1 (b) Traffic scene2

Table 2. Accuracy score for Traffic Scene1 and Traffic Scene 2

Traffic scene1 results in ratios		**Traffic scene2 results in ratios**	
Vehicle Type	**Accuracy in percentage**	**Vehicle Type**	**Accuracy in percentage**
motorbike	0.91	motorcar	0.828
motorcar	0.819	motorcar	0.752
motorcar	0.794	motorcar	0.737
motorcar	0.774	motorcar	0.699
motorcar	0.704	motorcar	0.696
motorcar	0.819	motorcar	0.691
bus	0.701	motorbike	0.670
motorcar	0.696	motorcar	0.631
motorcar	0.672	motorcar	0.621
motorcar	0.668	motorcar	0.601
motorcar	0.591	truck	0.588
motorcar	0.585	motorbike	0.569
motorcar	0.574	motorbike	0.565
motorcar	0.570	motorcar	0.562
motorbike	0.565	motorbike	0.555
motorbike	0.547	motorcar	0.553
truck	0.543	motorcar	0.542
motorbike	0.542	motorcar	0.536
motorcar	0.536	motorbike	0.536
motorbike	0.517	motorbike	0.532

motorbike	0.510	motorbike	0.523
motorcar	0.502	truck	0.518
motorcar	0.499	motorcar	0.511
bus	0.497	motorbike	0.510
motorcar	0.484	motorbike	0.471
motorcar	0.479	motorcar	0.464
motorbike	0.478	motorcar	0.448
motorbike	0.473	truck	0.438
bicycle	0.462	motorcar	0.426
motorbike	0.452	truck	0.425
motorcar	0.444	truck	0.421
truck	0.443	motorbike	0.411
truck	0.435	motorbike	0.409
truck	0.435	motorcar	0.408
motorbike	0.413	truck	0.518
motorbike	0.410	motorcar	0.511
truck	0.409	motorbike	0.510
motorcar	0.404	motorbike	0.471
Total vehicle count 38		**Total vehicle count 34**	

Table 3. Configuration for Traffic scene 3 and 4

Configuration done in Traffic scene 3		**Configuration done in Traffic scene 4**	
Epoch	300	Epoch	300
Confidence score	0.25	Confidence score	0.25

Figure 6. Vehicle finding by YOLOv5 architecture: (a) Traffic scene 3 (b) Traffic scene 4.

Table 4. Accuracy achieved YOLO v5

Traffic scene3 results in ratios		Traffic scene4 results in ratios	
Vehicle Type	**Accuracy in percentage**	**Vehicle Type**	**Accuracy in percentage**
motorcar	0.270633	motorcar	0.255833
motorbike	0.294898	motorcar	0.256711
motorcar	0.302571	truck	0.276956
motorbike	0.305478	motorbike	0.283592
motorcar	0.311523	motorcar	0.285504
motorbike	0.333797	motorbike	0.290127
motorbike	0.336011	motorcar	0.291473
motorbike	0.357609	motorcar	0.308051
truck	0.37085	motorbike	0.328948
bus	0.373993	motorcar	0.339722
bus	0.378617	motorcar	0.346691
motorbike	0.384037	motorcar	0.362092
motorbike	0.436987	motorbike	0.368429
motorbike	0.441676	motorcar	0.385204
motorbike	0.496965	motorbike	0.398947
motorbike	0.507397	motorbike	0.417542
motorcar	0.509181	motorbike	0.427298
motorbike	0.533995	motorbike	0.437601
motorbike	0.541369	motorbike	0.461392
motorcar	0.557437	motorbike	0.479142
motorcar	0.616531	motorcar	0.479186
motorcar	0.636341	motorbike	0.483516
motorbike	0.667697	motorbike	0.509056
motorcar	0.725113	motorbike	0.519409
motorcar	0.777148	motorcar	0.539643
motorcar	0.818693	motorcar	0.579279
motorcar	0.824248	motorcar	0.584506
motorcar	0.828488	motorcar	0.597602
motorcar	0.831022	motorcar	0.621419
motorcar	0.86426	motorbike	0.671043
motorcar	0.87088	motorcar	0.765274
		motorcar	0.767315
		motorcar	0.775077
		motorcar	0.81319
		motorcar	0.832016
		motorcar	0.882242
		car	0.925397
Total vehicles count: 31		**Total vehicles count: 37**	

4.3 YOLOv6 Experimental Results

Present traffic scenes are tested to find vehicles with the YOLOv6 model depending on COCO dataset labels. The model is inferred and the outcomes are kept in the inference folder. For inference, we implement the inference python program by loading YOLOv6 scales from YOLO6s.pt whereas keeping a source directory containing all traffic scenes. The outcomes, including vehicle detection, can be seen in the traffic scenes in Figure 7 below. Table 5 gives the experimental configuration used for YOLOv6.

The assurance score value is specified as 0.25 and the exact search of vehicles is performed by the YOLOv6 system. Traffic scenes are also experimented with yolov6s.pt weights and yolov6n.pt weights. The detection is found more accurate in yolov6s.pt than with yolov6n.pt weights. The total count of vehicles for every picture is calculated from the text file formed later performing the finding technique is shown in Table 6. YOLOv6 is fast, but finding of vehicles is almost same as YOLOv5.

4.4 YOLOv7 Experimental Results

PyTorch is used to evaluate the YOLOv7 model's vehicle finding capabilities in real-time traffic circumstances. Experiments are conducted by running the YOLOv7 detect python file and providing weights to yolov7.pt. Table 7 shows the experimental configuration used for YOLOv7. A text file with the results as shown in Figure 8 is created, and the total count of vehicles is calculated for analyses as shown in Table 8.

At the same inference speeds, YOLOv7 approach draws boxes more precisely than the earlier versions. With the help of the algorithm, we were able to identify numerous motorcars in the traffic scene as shown in Table 8 with confidence scores of over 30% and motorbikes with confidence scores of over 29%. Compared to YOLOv6 and YOLOv5, YOLOv7 is substantially better at finding vehicles. The precision of vehicle finding is greater, and inference speed is faster. The YOLOv7 performance delivers excellent discovery precision with reduced running time.

4.5 YOLOv8 Experimental Results

The YOLOv8 model is a highly recommended solution for a diverse selection of tasks, including vehicle discovery, picture separation, and picture categorization in different areas of road. The execution of YOLOv8 model can be done through the command line mode starting with a YOLO command. Yolo takes additional arguments and can be used for several jobs and modes. We can also execute YOLOv8 model using the python programming interface giving the same command line procedure and arguments. The YOLOv8 structure now supports all YOLO architectures, not only the v8 version. Table 9 shows the configuration settings used for YOLOv8. Figure 9(a-b) depicts sample traffic scenes with object detection by YOLOv8 with a Vehicle finding ratios and counting by YOLOv8 are shown in Table 10.

5. Comparison of models

The inference generated by different models: RetinaNet, YOLOv5, YOLOv6, YOLOv7 and YOLOv8 are compared. The comparative information is specified in Table 11. Comparison of IOU and mean average precision are shown in Table 12.

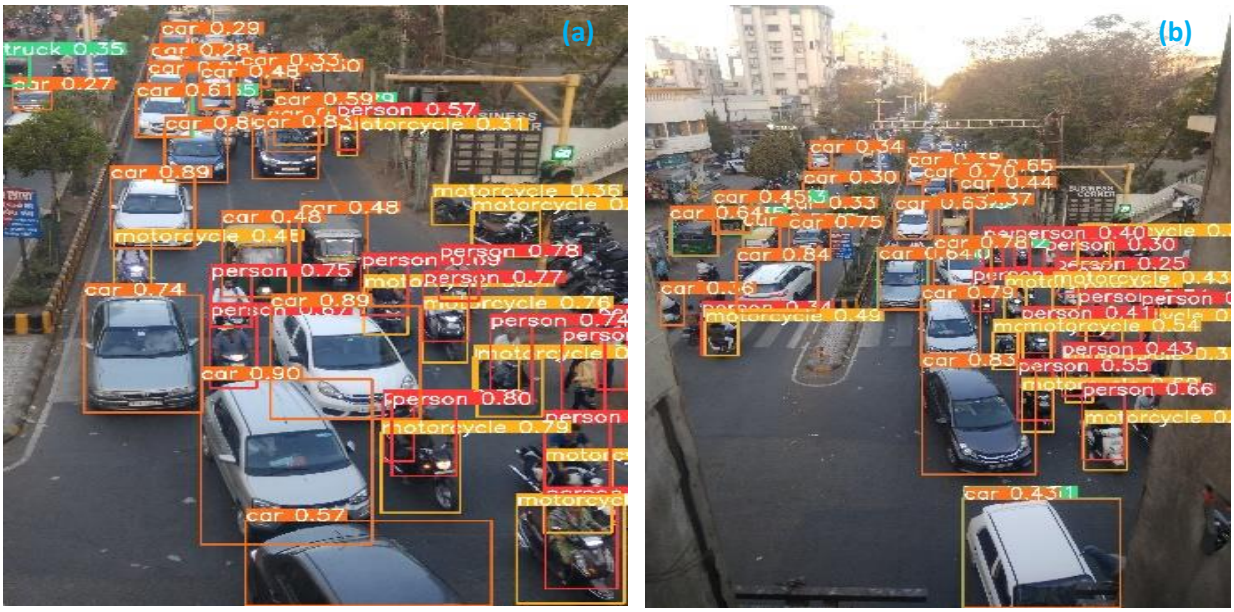

Figure 7. Vehicle finding by YOLOv6 architecture: (a) Traffic scene5 (b) Traffic scene6

Table 5. Configurations for scene 5 and 6

Configuration done in Traffic scene5		**Configuration done in Traffic scene6**	
Assurance score	0.25	Assurance score	0.25
Weights used	yolov6n.pt	Weights used	yolov6n.pt
	yolov6s.pt		yolov6s.pt

Table 6. Accuracy score for YOLO v6

Traffic scene5 results in ratios		**Traffic scene6 results in ratios**	
Vehicle Type	**Accuracy in percentage**	**Vehicle Type**	**Accuracy in percentage**
motorbike	0.260062	Motorbike	0.260151
motorcar	0.268043	truck	0.295251
motorbike	0.269405	motorcar	0.297282
motorcar	0.270306	motorbike	0.30751
motorcar	0.275	motorbike	0.308834
motorcar	0.281858	truck	0.311784
truck	0.290482	truck	0.320069
motorcar	0.294335	motorcar	0.328065
motorcar	0.295114	truck	0.332544
motorbike	0.311958	motorbike	0.334587
motorcar	0.32724	motorcar	0.338355
truck	0.348197	truck	0.347572
truck	0.353051	truck	0.352693
motorbike	0.35909	motorcar	0.355528
motorbike	0.477036	motorcar	0.366974
motorcar	0.479909	motorcar	0.380556
motorcar	0.482395	truck	0.385722
motorcar	0.482452	motorbike	0.427534
motorbike	0.496994	motorcar	0.4333
motorbike	0.535461	motorcar	0.438365
motorcar	0.574991	motorbike	0.446969
motorcar	0.593152	motorcar	0.45289
motorbike	0.594723	motorbike	0.493434
motorcar	0.607415	motorbike	0.530705
motorbike	0.658722	motorbike	0.544792
motorcar	0.740557	motorcar	0.549072
motorbike	0.75932	motorbike	0.604717
motorbike	0.786572	motorcar	0.629918
motorcar	0.796362	motorcar	0.636936

motorcar	0.834489	motorcar	0.641227
motorcar	0.886286	motorcar	0.64933
motorcar	0.891913	motorcar	0.702426
motorcar	0.901568	motorcar	0.749039
		motorcar	0.779763
		motorcar	0.788192
		motorcar	0.828632
		motorcar	0.837292
Total vehicles count: 33		Total vehicles count: 37	

Table 7. Configurations for YOLO v7

Configuration done in Traffic scene7		**Configuration done in Traffic scene8**	
Weights used	yolov7.pt	Weights used	yolov7.pt
Confidence score	0.25	Confidence score	0.25
save_txt	True	save_txt	True
save_conf	true	save_conf	true

Figure 8 Vehicle finding by YOLOv7 architecture: (a, c) Traffic scene7 (b, d) Traffic scene8

Table 8. Accuracy for YOLO v7

Traffic scene7 results in ratios		**Traffic scene8 results in ratios**	
Vehicle Type	**Accuracy in percentage**	**Vehicle Type**	**Accuracy in percentage**
truck	0.257116	motorcar	0.260167
motorbike	0.259187	motorcar	0.264858
motorbike	0.312189	motorbike	0.291762

motorbike	0.355464	motorbike	0.32433
truck	0.371609	motorcar	0.345887
motorcar	0.373668	motorbike	0.365894
motorbike	0.376711	motorcar	0.377315
motorcar	0.379007	truck	0.379879
motorcar	0.381933	motorbike	0.399485
motorbike	0.436986	truck	0.399768
motorcar	0.444589	motorcar	0.401903
motorcar	0.446888	motorbike	0.40231
motorbike	0.450354	motorcar	0.430155
motorbike	0.499502	motorbike	0.468989
truck	0.536581	motorbike	0.474374
motorbike	0.561187	truck	0.524824
motorbike	0.571602	motorbike	0.529871
motorcar	0.6025	motorbike	0.557248
motorbike	0.607168	truck	0.56113
motorcar	0.666344	motorcar	0.586487
motorcar	0.712134	motorbike	0.591815
motorbike	0.717952	motorbike	0.611843
bus	0.721377	truck	0.614766
motorbike	0.724302	motorbike	0.622468
motorbike	0.731917	motorcar	0.628231
motorbike	0.736459	motorbike	0.633681
motorbike	0.762183	motorbike	0.710865
motorcar	0.815315	motorbike	0.714016
motorcar	0.832759	motorbike	0.718434
motorcar	0.839949	motorbike	0.736138
motorcar	0.852814	motorcar	0.749721
motorcar	0.88877	motorcar	0.769283
motorcar	0.898797	motorcar	0.826532
motorcar	0.925041	motorcar	0.854255
		motorcar	0.864586
		motorcar	0.889316
		motorcar	0.891992
		motorcar	0.921667
		motorcar	0.935801
Total vehicles count: 34		Total vehicles count: 39	

Table 9. Configurations for YOLOv8

Configuration done in Traffic scene 9		**Configuration done in Traffic scene 10**	
Weights used	yolov8n.pt	Weights used	yolov8n.pt
	yolov8s.pt		yolov8n.pt
Confidence score	0.25	Confidence score	0.25
Epoch	100	Epoch	100
save_txt (Save to text file)	True	save_txt (Save to text file)	True
save (Save images with results)	True	Save (Save images with results)	True
save_conf (Save results with confidence score)	true	save_conf (Save results with confidence score)	true

Figure 9. Vehicle finding by YOLOv8 architecture: (a) Traffic scene9 (b) Traffic scene10

Table 10. Accuracy for YOLOv8

Traffic scene9 results in ratios		**Traffic scene10 results in ratios**	
Vehicle Type	**Accuracy in percentage**	**Vehicle Type**	**Accuracy in percentage**
motorcar	0.250695	motorcar	0.250475
motorbike	0.283045	motorbike	0.269626
motorcar	0.285067	motorbike	0.273694
motorcar	0.313786	motorcar	0.273944
truck	0.325756	motorcar	0.296799
truck	0.327067	motorbike	0.338501
motorbike	0.375696	motorbike	0.345737
motorbike	0.376555	motorbike	0.346801
motorbike	0.383041	truck	0.37397
motorcar	0.394837	motorbike	0.390137
motorbike	0.407953	truck	0.391953
motorcar	0.412414	motorcar	0.418386
motorbike	0.508428	motorbike	0.421259

motorbike	0.524795	motorcar	0.440974
motorbike	0.525136	motorcar	0.462227
bus	0.528271	motorbike	0.47407
motorcar	0.566357	motorbike	0.484203
motorbike	0.566725	motorbike	0.543637
motorcar	0.59695	motorcar	0.549816
motorbike	0.632423	motorcar	0.562443
bus	0.657576	motorcar	0.571549
motorbike	0.659603	motorbike	0.616666
motorbike	0.668292	motorbike	0.657768
motorcar	0.731787	motorcar	0.697213
motorcar	0.755035	motorcar	0.702297
motorcar	0.762165	motorcar	0.72752
motorcar	0.801853	motorcar	0.738035
motorcar	0.83133	motorcar	0.776989
motorcar	0.850255	motorcar	0.813323
motorcar	0.87293	motorcar	0.842514
motorcar	0.897395	motorcar	0.862867
motorcar	0.921514		
motorcar	0.922723		
Total vehicles count: 33		Total vehicles count: 38	

Table 11. Comparison of inference speed and total count accuracy in vehicle discovery

Model	**RetinaNet**		**YOLOv5**		**YOLOv6**		**YOLOv7**		**YOLOv8**	
Picture name	**Traffic scene**		**Traffic scene**		**Traffic scene**		**Traffic scene**		**Traffic scene**	
Picture Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Vehicles Total count	38	34	31	37	33	37	34	39	33	38
Inference Speed in seconds	3.4s	3.8s	0.2s	0.4s	1.9s	2.4s	2.2s	2.3s	0.2s	0.3s

Table 12. Comparison of IOU and mean average precision in vehicle discovery

Model	**RetinaNet**	**YOLOv5**	**YOLOv6**	**YOLOv7**	**YOLOv8**
Confidence Score	0.05	0.25	0.25	0.25	0.25
Intersection over Union (IoU)	0.5	0.45	0.45	0.45	0.7
Mean Average Precision (mAP)	84%	74%	74%	74%	50%

It is found from the results that RetinaNet model gives accurate results but takes more time. Also, it is found that YOLO5 results are very accurate as it can detect a high number of vehicles in very less time as compared to other models. YOLOv6 is slightly better than YOLOv5 in accuracy but speed is less than

YOLOv5 in finding vehicles. YOLOv7 is more advanced than YOLOv6 and YOLOv5. It also provides accurate results in vehicle detection and counting at significantly higher speeds. YOLOv7 model allows us to spot and count several vehicles. YOLOv8 is the newest developed version and it is quick and accurate than all other models. Therefore, depending on the above results, it can be concluded that traffic simulation can be applied to continuous traffic movements in real time with the support of various camera, sensors and photographs processing methods. This information should be used for assessing and monitoring traffic progress based on actual traffic densities. A program can be formed to implement the traffic management framework by generating complete vehicle information and categories [32].

6. Conclusion

This paper presented extensive experimental evaluation of several vehicle discovery deep learning based using a common traffic image dataset. Total five deep learning architectures such as RetinaNet, YOLOv5, YOLOv6, YOLOv7 and YOLOv8 are reviewed and compared. The base architecture of all these models is presented at first. It is proposed that by efficiently using these systems, we can reduce traffic bottleneck on our city's road. Vehicles are counted and a traffic report is prepared. It is found from the results out of all applied models YOLOv8 is the newest developed version and it is quick and accurate than all other models. Using YOLOv8, we can implement any other YOLO model as well. However, we have restrictions in the YOLO system. Little vehicles such as motorbikes, which are together, as seen in traffic scenes are not identified with YOLOv8 models, however YOLOv7 has shown better counts for detecting smaller vehicles as well. The study presented can make recommendations towards a development of a completely automated traffic management system in real time and that can suggest reroutes based on current traffic levels on the roadways. The deep learning based traffic assessment system can be integrated in varied routes suggestive applications for real time informed decisions and aid traffic management in efficient manner.

References

[1] C. Janiesch, P. Zschech, K. Heinrich, Machine learning and deep learning. Electronic Markets, Springer, 31, (2021) 685-695. https://doi.org/10.1007/s12525-021-00475-2

[2] H. Kukadiya, D. Meva, N. N. Arora, S. Srivastava, Effective Groundnut Crop Management by Early Prediction of Leaf Diseases through Convolutional Neural Networks. International Research Journal of Multidisciplinary Technovation, 6(1), (2023) 17–31. https://doi.org/10.54392/irjmt2412

[3] C. Zhao, R. W. Liu, J. Qu, R. Gao, Deep learning-based object detection in maritime unmanned aerial vehicle imagery: Review and experimental comparisons. Engineering Applications of Artificial Intelligence, 128, (2024) 107513. https://doi.org/10.1016/j.engappai.2023.107513

[4] G. Cheng, X. Xie, J. Han, L. Guo, G.S. Xia, Remote sensing image scene classification meets deep learning: Challenges, methods, benchmarks, and opportunities. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 13, (2020) 3735-3756. https://doi.org/10.1109/JSTARS.2020.3005403

[5] C. Li, X. Li, M. Chen, X, Sun, (2023) Deep Learning and Image Recognition. In 2023 IEEE 6th International Conference on Electronic Information and Communication Technology, IEEE, China. https://doi.org/10.1109/ICEICT57916.2023.10245041

[6] S.S.A. Zaidi, M.S. Ansari, A. Aslam, N. Kanwal, M. Asghar, B. Lee, A survey of modern deep learning based object detection models. Digital Signal Processing, 126, (2022) 103514. https://doi.org/10.1016/j.dsp.2022.103514

[7] S.M. Nagarajan, G.G. Deverajan, A.K. Bashir, R.P. Mahapatra, M.S. Al-Numay, IADF-CPS: Intelligent Anomaly Detection Framework towards Cyber Physical Systems. Computer Communications, 188, (2022) 81-89. https://doi.org/10.1016/j.comcom.2022.02.022

[8] M.A. Amanullah, R.A.A. Habeeb, F.H. Nasaruddin, A. Gani, E. Ahmed, A.S.M. Nainar, N.M. Akim, M. Imran, Deep learning and big data technologies for IoT security. Computer Communications, 151, (2020) 495-517. https://doi.org/10.1016/j.comcom.2020.01.016

[9] M. Karuppiah, T.V. Ramana, R. Mohanty, G.G. Devarajan, S.M. Nagarajan, UIoTN-PMSE: Ubiquitous IoT network-based predictive modeling in smart environment. International Journal of Communication Systems, (2023) e5661. https://doi.org/10.1002/dac.5661

[10] G.G. Devarajan, M. Thirunnavukkarasan, S.I. Amanullah, T. Vignesh, A. Sivaraman, An integrated security approach for vehicular networks in smart cities. Transactions on Emerging Telecommunications Technologies, 34(11), (2023) e4757. https://doi.org/10.1002/ett.4757

[11] M. Abbasi, A. Shahraki, A. Taherkordi, Deep learning for network traffic monitoring and analysis (NTMA): A survey. Computer Communications, 170, (2021) 19-41. https://doi.org/10.1016/j.comcom.2021.01.021

[12] J. Guo, H. He, T. He, L. Lausen, M. Li, H. Lin, X. Shi, C. Wang, J. Xie, S. Zha, A. Zhang, H. Zhang, Z. Zhang, Z. Zhang, S. Zheng, Y. Zhu, Gluoncv and gluonnlp: Deep learning in computer vision

and natural language processing, The Journal of Machine Learning Research, 21(1), (2020) 845-851.

[13] C. Cai, M. Wei, Adaptive urban traffic signal control based on enhanced deep reinforcement learning. Scientific Reports, 14, (2024) 14116. https://doi.org/10.1038/s41598-024-64885-w

[14] B. Lin, Y. Guo, H. Luo, M. Ding, TITE: A transformer-based deep reinforcement learning approach for traffic engineering in hybrid SDN with dynamic traffic, Future Generation Computer Systems, 161, (2024) 95-105. https://doi.org/10.1016/j.future.2024.07.006

[15] A. John, D. Meva, Comparative Study of Various Algorithms for Vehicle Detection and Counting in Traffic, Communications in Computer and Information Science. Advancements in Smart Computing and Information Security (ASCIS 2022), 1760, (2022) 271–286. https://doi.org/10.1007/978-3-031-23095-0_20

[16] C.J. Lin, J.Y. Jhang, Intelligent traffic-monitoring system based on YOLO and convolutional fuzzy neural networks. IEEE Access, 10, (2022) 14120-14133. https://doi.org/10.1109/ACCESS.2022.3147866

[17] Y.Q. Huang, J.C. Zheng, S.D. Sun, C.F. Yang, J. Liu, Optimized YOLOv3 algorithm and its application in traffic flow detections. Applied Sciences, 10(9), (2020) 3079. https://doi.org/10.3390/app10093079

[18] R. Rahman, Z.B. Azad, B.M. Hasan, Densely-populated traffic detection using YOLOv5 and non-maximum suppression ensembling. Proceedings of the International Conference on Big Data, IoT, and Machine Learning, 95, (2021) 567–578. https://doi.org/10.1007/978-981-16-6636-0_43

[19] L. Zhang, H. Wang, X. Wang, S. Chen, H. Wang, K. Zheng, H. wang, Vehicle Object Detection Based on Improved RetinaNet. Journal of Physics: Conference Series, 1757(1), (2021) 012070. https://doi.org/10.1088/1742-6596/1757/1/012070

[20] A. Abdullah, J. Oothariasamy, Vehicle counting using deep learning models: a comparative study. International Journal of Advanced Computer Science and Applications, 11(7), (2020) 697-703. https://doi.org/10.14569/IJACSA.2020.0110784

[21] A. Saadeldin, M.M. Rashid, Video-based vehicle counting and analysis using YOLOv5 and DeepSORT with deployment on Jetson Nano. AJoEEE, 2(2), (2022) 11-20. https://doi.org/10.69955/ajoeee.2022.v2i2.34

[22] Y. Chen, Z. Li, An Effective Approach of Vehicle Detection Using Deep Learning. Computational Intelligence and Neuroscience, (2022) 1-9. https://doi.org/10.1155/2022/2019257

[23] Y. Zhang, Z. Guo, J. Wu, Y. Tian, H. Tang, X. Guo, Real-Time Vehicle Detection Based on Improved YOLO v5. Sustainability, 14(19), (2022) 12274. https://doi.org/10.3390/su141912274

[24] C.J. Lin, S.Y. Jeng, H.W. Lioa, A real-time vehicle counting, speed estimation, and classification system based on virtual detection zone and YOLO. Mathematical Problems in Engineering, 2021, (2021) 1577614. https://doi.org/10.1155/2021/1577614

[25] A. Bochkovskiy, C.Y. Wang, H.Y. M. Liao, (2020) YOLOv4: Optimal Speed and Accuracy of Object Detection. ArXiv.

[26] I. Katsamenis, E.E. Karolou, A. Davradou, E. Protopapadakis, A. Doulamis, N. Doulamis, D. Kalogeras, TraCon: A novel dataset for real-time traffic cones detection using deep learning. In Novel & Intelligent Digital Systems Conferences, 556, (2022) 382-391. https://doi.org/10.1007/978-3-031-17601-2_37

[27] U. Nepal, H. Eslamiat, Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs. Sensors, 22(2), (2022) 464. https://doi.org/10.3390/s22020464

[28] S.M. Ali, Comparative analysis of YOLOv3, YOLOv4 and YOLOv5 for sign language detection. International Journal of Advance Research and Innovative Ideas in Education, 7(4), (2021) 2393-2398.

[29] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, Y. Li, B. Zhang, Y. Liang, L. Zhou, X. Xu, X. Chu, X. Wei, X. Wei , (2022) YOLOv6: A single-stage object detection framework for industrial applications. ArXiv. https://doi.org/10.48550/arXiv.2209.02976

[30] C.Y. Wang, A. Bochkovskiy, H.Y.M. Liao, (2023) YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Canada. https://doi.org/10.1109/CVPR52729.2023.00721

[31] M. Sohan, T.S. Ram, V. Ch, A Review on YOLOv8 and Its Advancements. Data Intelligence and Cognitive Informatics, (2024) 529-545. https://doi.org/10.1007/978-981-99-7962-2_39

[32] N.N. Hasibuan, M. Zarlis, S. Efendi, Detection and tracking different type of cars with YOLO model combination and deep sort algorithm based on computer vision of traffic controlling. Sinkron: jurnal dan penelitian teknik informatika, 5(2B), (2021) 210-220.

Authors Contribution Statement

Anand John: Conceptualization, Investigation, methodology, data collection, Paper formulation and Writing original draft. Divyakant Thakarshibhai Meva:

Conceptualization, Supervision, Review, Paper formulation and Editing. Nidhi Arora: -Conceptualization, Supervision, Review, Paper formulation and Editing. All the authors read and approve the final version of the manuscript

Acknowledgement
Anand John thanks the Almighty God for everything. The authors appreciate the continuous support of Marwadi University, Gujarat, India

Funding
The authors declare that no funds, grants or any other support were received during the preparation of this manuscript.

Conflict of Interest
The authors declare that there are no conflicts of interest regarding the publication of this manuscript.

Data Availability
Data of traffic images will be made available upon request

Has this article screened for similarity?

Yes

About the License