Multimedia Tools and Applications (2024) 83:6755–6781
https://doi.org/10.1007/s11042-023-15365-2

IRUVD: a new still-image based dataset for automatic vehicle detection

Asfak Ali[1] · Ram Sarkar[2] · Debesh Kumar Das[2]

Received: 27 August 2021 / Revised: 11 April 2023 / Accepted: 15 April 2023 /
Published online: 17 June 2023

Abstract
One of the difficult tasks in the field of computer vision is the classification and detection of vehicles. Researchers from all over the world are working to create autonomous vehicle detection (AVD) systems due to their numerous practical applications, including highway management and surveillance systems. Deep learning techniques, which require a lot of data for proper model training, are the current AVD trend. However, a number of vehicles are discovered in India, the second-largest nation in terms of population, that are not included in the vehicle detection datasets that are currently in use. Furthermore, India's overcrowding makes traffic management difficult and unusual. In this research, we present a dataset for still-image-based vehicle detection that includes one class of pedestrians and 13 different types of vehicles that are seen on Indian urban and rural roads. Initially, we provide baseline results using some state-of-the-art deep learning models on this dataset. To improve the accuracy further, we present an ensemble-based object detection and classification model. The dataset consists of 4K images and 14.3K bounding boxes of various vehicles; that is, researchers are provided with appropriately annotated rectangular boxes for use with these vehicles in the future. A 16-megapixel Sony IMX519 high-resolution camera was used to take all images while travelling throughout West Bengal, an Indian state on the eastern side. Dataset can be found at: https://github.com/IRUVD/IRUVD.git.

Keywords Vehicle detection · Vehicle classification · Deep learning · Traffic management · Dataset

✉ Asfak Ali
asfakali.etce@gmail.com

Ram Sarkar
ramjucse@gmail.com

Debesh Kumar Das
debeshd@hotmail.com

[1] Department of Electronics and Tele-communication Engineering, 188, Raja S.C. Mallick Road, Kolkata, 700032, West Bengal, India

[2] Department of Computer Science Engineering, 188, Raja S.C. Mallick Road, Kolkata, 700032, West Bengal, India

1 Introduction

Vehicles play a crucial role in our daily lives. The most important part of the transportation department is the traffic management system. The initial traffic control system was created in the 1910s. Through the use of computer vision techniques, traffic management systems have been evolved over time to become more intelligent and efficient. In addition to the traffic management system, a lot of data is required for other systems as well including surveillance, pollution control, autonomous vehicle detection (AVD), and vision-based vehicle parking management. Most importantly, the system needs to be reliable and accurate in order to be used in real-world AVD applications. Many datasets [16, 17] have been created in recent years for a variety of object detection purposes including vehicle detection. Autonomous vehicles locates obstacles using camera captures images from the vehicle. The range of the sensor and the size of the target must be considered when detecting vehicles [26]. It should be noted that the complexity of newer make and model vehicles' shapes, sizes, colors, and textures is making this research field more challenging [11].

The pose and viewing angle make it difficult for a model to identify the precise object class. Additionally, it is challenging to create a generalized AVD dataset due to the numerous country or region-specific issues. The most widely used datasets like Pascal VOC2007 [10] and KITTI [13], only include five and seven vehicle classes that are frequently seen on urban roads, respectively. Only a few datasets are available that were created with Indian traffic conditions in mind. The most popular Indian vehicle dataset, known as IDD [38], is used for segmentation tasks. Images were taken in Bangalore and Hyderabad, two popular Indian cities. There are a total of 34 classes in this dataset, including 8 classes for vehicles. Other classes include the sky, roadside objects (such as walls, fences, billboards, etc.), distant objects (such as buildings and bridges), living things (such as animals and people), drivable and non-drivable objects (such as sidewalks and non-drivable fallbacks). Another Indian vehicle detection dataset is NITCAD [28], which includes 7 different vehicle classes. Images were collected throughout Kerala, a southern state of India, for this dataset. Despite the fact that auto-rickshaws are widely used in both urban and rural areas of India, NITCAD and IDD only included them in their datasets because they are frequently seen in urban areas.

However, more vehicles, including totos, cycle-rickshaws, and motor-rickshaws, are commonly found in rural areas, and those are not incorporated in any of the aforementioned datasets. Due to inadequate traffic control systems, auto-rickshaws and motorcycles frequently break traffic laws in rural areas. In congested urban areas, crosswalks are not always used by pedestrians. These elements contribute to India's extremely difficult traffic management systems. The fact that the same kind of vehicle is used for multiple purposes in India presents another challenge for vehicle detection, and it confuses the computer vision based learning models. A motor-rickshaw, for instance, can carry both passengers and goods. This study offers a new dataset for AVD to address those problems, and an ensemble of deep-learning models is used to provide baseline results. Vehicle detection using still images is a challenging task due to various factors, such as vehicle orientation, lighting conditions, and scale variations. In the Indian context, there are several additional challenges to consider, such as non-standard vehicles, crowded scenes, and diverse driving behaviors. This article explores the challenges in vehicle detection in the Indian context and proposes solutions to address them. The first step in developing an effective vehicle detection system is to collect a standard dataset. The dataset should cover a range of scenarios, including different lighting conditions, backgrounds, and vehicle types and orientations. In the Indian context,

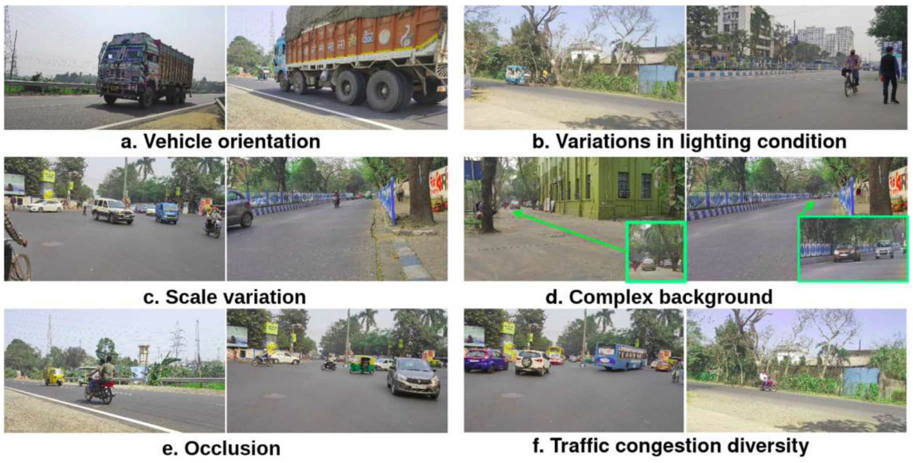

Fig. 1 Sample images taken from the dataset developed in the present work

the dataset should also include some uncommon vehicles (in other countries) such as auto-rickshaws and cycle-rickshaws. Figure 1 shows an example of a dataset that considers these factors.

One of the significant challenges in vehicle detection is the orientation of vehicles. Vehicles can have different orientations, such as front-facing, side-facing, or rear-facing, which can make it difficult to detect them using traditional methods. Additionally, variations in lighting conditions can affect the accuracy of detection. Vehicles can appear at different scales in images or videos, making it challenging to detect them using existing datasets. Furthermore, vehicles may blend into complex backgrounds, such as trees, buildings, and other vehicles, which can make it difficult to distinguish them from the surroundings. In the Indian context, traffic congestion is another significant issue. The high volume of traffic, particularly in urban areas, can make it problematic to detect vehicles accurately in crowded scenes. Moreover, in rural areas, we can observe a wide range of sometimes erratic driving behaviours, which can further complicate the detection task. Additionally, India has a diverse range of uncommon vehicles, such as auto-rickshaws and cycle-rickshaws, which can be difficult to detect if a model is trained on existing datasets.

Contributions With the aforementioned information in mind, in this paper, we have developed the IRUVD: Indian Rural and Urban Vehicle Detection, a new still-image-based dataset for AVD. Specific contrubutions of this paper are as follows:

– A new vehicle detection dataset that includes 13 vehicle classes and 1 pedestrian object has been introduced. Toto, cycle-rickshaw, and motor-rickshaw classes of vehicles that are frequently seen on Indian roads have been considered.
– To make the dataset as realistic as possible, we have taken into account both urban and rural areas of India during data collection. This aids in capturing a variety of traffic scenarios, including both low-congested rural areas without a traffic system and highly congested urban areas with a well-maintained traffic system.
– There are several challenges that are considered while preparing the current dataset, such as vehicle orientation, variations in lighting conditions, scale variation of the objects, complex backgrounds, occlusions, and different traffic diversities.

– This dataset contains 14343 properly annotated bounding boxes for 4000 labeled images. Figure 1 displays some examples. The images were captured from various locations in West Bengal, a state in eastern India. The images were taken during the day, and the objects were in various poses. The resolution of each image is 1920 × 1080.
– Utilizing the most up-to-date deep learning-based object detection models, we have benchmarked the IRUVD dataset using the You Only Look Once version 3 (YOLOv3), YOLO version 4 (YOLOv4), Scaled YOLO version 4 (Scaled-YOLOv3), and YOLO version 5 (YOLOv5) (YOLOv5). We have used a variety of object detection metrics, including recall, precision, F1-score, mean average precision (mAP) at intersection over union (IOU) threshold 0.5 (mAP@0.5), mAP score at IOU 0.75 (mAP@0.75), mAP scores at IOU 0.95 in steps of 0.05 (mAP@0.5:0.05:0.95), and mAP scores at IOU 0.95 (mAP@0.75:0.05:0.95).
• On the IRUVD dataset, we have proposed both weighted and non-weighted ensemble approaches to establish the baseline results. In this case, we have observed that the non-weighted ensemble approach performs better than the weighted ensemble approach.

The rest of the paper is organized as follows. A literature review is given in Section 2. The developed dataset is presented in Section 3. In Section 4, we describe the procedure for benchmarking the dataset. To further improve detection results, we introduce an ensemble technique in Section 5. In Section 6, we tested the proposed ensemble technique on some additional datasets. Finally, in Section 7, we conclude our paper.

2 Literature survey

Researchers have created various datasets to address a variety of difficult problems in the field of computer vision. The most popular datasets for image classification, localization, and segmentation are ImageNet [6], Microsoft COCO [22], ADE20K [48], and Pascal VOC. But only Pascal VOC2007 can be used for vehicle detection tasks. The two categories of existing vehicle detection datasets are segmentation-based datasets and localization-based datasets, which are briefly discussed below.

2.1 Segmentation-based detection dataset

Segmentation implies the finding an exact outline around the object in an image. The most widely used dataset, Pascal VOC2007, has 20 main classes, but only 5 of them can be applied for developing autonomous vehicle and traffic management systems. The most well-known datasets for segmentation-based vehicle detection are Cityscapes [5], Mapillary Vistas [29], and CBCL StreetScenes [2]. The 3.5k images in CBCL StreetScenes, which have 9 classes including cars, pedestrians, bicycles, buildings, trees, sky, roads, sidewalks, and stores, were collected from urban streets in Boston, Massachusetts, in the United States. Another dataset for vehicle detection with 25k images is Vistas. With 11 different vehicle types and a total of 37 classes, the images of this dataset were collected from urban streets across 6 continents with the goal of diversifying the detection models. A segmentation-based dataset that took into account both urban and rural areas of the USA is the Berkeley Deep Drive Video dataset. The videos used to create this dataset totaled 10,000 hours. The largest vehicle dataset, BDD100K [44], was collected from New York City. Each of the 100,000 video sequences in this dataset has a duration of 40 seconds and ten different object classes. The primary driving force behind the creation of this dataset was the desire to experience various time and weather conditions, such as sunny, cloudy, and rainy. The Cityscapes segmentation-based vehicle detection dataset, which was created taking Indian traffic conditions into consideration, is the one that most closely resembles IDD. Only 8 of the 45k images' 34 object classes are clearly identifiable as vehicle classes. However, some vehicles that are frequently seen on Indian rural roads are not taken into account in this dataset.

2.2 Localization-based detection dataset

Localization is a method for determining an object's precise location, which is typically indicated by a rectangular box. For the sole purpose of detecting pedestrians, datasets like INRIA [7], Daimler Pedestrian [27], TudBrussels [43], and Citypersons dataset [45] were created. However, most objects encountered on the road must be able to be detected by AVD systems. A 3D car classification dataset with 207 car categories was introduced by Krause et al. [20]. The most well-known vehicle detection dataset, known as KITTI, was developed using images from Karlsruhe, Germany, and eight different vehicle classes. The KITTI dataset has 200k 3D bounding boxes and 15k images with 8 different object classes. The most recent Iranian vehicle detection dataset, LRVD [19], contains 110k images and 5 classes. BIT dataset [8] is another vehicle classification dataset that consists of 6 classes - bus, SUV, minivan, truck, microbus, and Sedan. It has more than 9.8k images which were taken day and night times from a camera installed on the highway. The limitation of this dataset is that it only contains a front view of every vehicle which is not preferable in a real-life scenario. NITCAD is a stereo vision-based autonomous navigation dataset, which was collected from Kerala, India. It has 7.5k distorted images. The main motive to develop this dataset was to provide more information about Indian roads. However, it only mentions one new class than other datasets, i.e., auto-rickshaw. Other vehicles like the toto, cycle-rickshaw, and motor-rickshaw seen on Indian roads were not taken into account when these datasets were made. In most cases, the object detection model predicts the presence of multiple objects within a single bounding box. The most accurate bounding boxes can be determined using either Non-maximum Suppression (NMS) or Soft Non-maximum Suppression (Soft-NMS).

2.2.1 Non-maximum suppression

Every detection model calculates the object's location using a bounding box or anchor box, the box's class, and the prediction percentage's level of confidence. A filtering technique called the NMS is used to get rid of overlapping bounding boxes. IOU is a metric that is typically calculated for each object class to determine the maximum amount of box overlap. The highest IOU box is the only one left for the final prediction. However, this method eliminates partially obscured objects of the same class, which is undesirable. Hence, this type of technique is used in the training process of a detection model.

2.2.2 Soft non-maximum suppression

To predict the most ideal bounding box from multiple bounding boxes, Bodla et al. [4] developed soft-NMS. In contrast to NMS, soft-NMS assigns the score based on the IOU value. In this method, a very low confidence score is given if the IOU value is high, which raises the detection model's training accuracy. Soft-NMS techniques are not very good at ensembling, though.

To develop an intelligent traffic control system in the Indian context, the above dataset may not be useful because of the lack of information about the vehicles like toto, cycle-rickshaw, motor-rickshaw, and tempo which are frequently seen on Indian roads. To this end, we have created a dataset with 14 classes that include a few special vehicle classes that, to the best of our knowledge, are not included in any other datasets. Since both urban and rural roads were taken into consideration, this dataset effectively captures both structured and unstructured traffic scenarios. This is a crucial component of developing nations like India's traffic management system. Table 1 compares the widely used datasets with the one we have created. YOLOv3, YOLOv4, Scaled-YOLOv4, and YOLOv5 are the four most recent deep learning-based object detection models used to benchmark the results on this dataset. Additionally, weighted and non-weighted ensembles of these deep learning models are employed to boost the detection accuracy. The ensemble technique, which is primarily used in the machine learning field, combines the decisions of multiple predictive models to make the final prediction that is more accurate than those made by the base models. It has the following main advantages,

- One of the main benefits of using the ensemble technique is that it improves the performance of the average accuracy of any present member.
- It reduces the bias-variance trade-off of the contributing members.
- It improves the robustness of the models.

3 Developed object detection dataset

Data collection and annotation process, quality and statistics of the data, and comparison with other vehicle detection datasets are presented in this section.

3.1 Data collection

Due to the lack of structure in the traffic management system, 11% of all fatalities worldwide occur in India [15]. In order to create more reliable systems that can be applied in the real world, we have created a still-image-based AVD dataset that takes into account the unique characteristics of Indian roads and vehicles. We have gathered information from various locations and at various times of the day to adequately represent the variety of traffic conditions. The traffic control system is more organized and effectively managed in urban areas. However, in rural areas, some vehicles and pedestrians disobey traffic laws, which leads to a high number of traffic accidents. When gathering the data, different perspectives, including the front, side, and back views of the same object, were taken into account. As the images were captured from different angles, the size of the objects varies a lot. We made use of a 16MP Sony IMX519 high-resolution camera to take all of these pictures. To create the database, we watched 1080p footage at 60 frames per second for more than 5 hours (during various times). Some samples images are already shown in Fig. 1.

3.2 Annotation process

A non-iconic image is one that contains multiple objects, which makes it more difficult for researchers to identify objects accurately in such images. To accurately assess the performance of any existing or newly developed methods, the data annotations must be flawless. The majority of the dataset's images are not iconic in any way. Although the annotation

Table 1 Comparison of existing vehicle detection datasets with the IRUVD

Dataset	Year	Avg. image resolution	Train	Location	Type	No. classes
CBCL StreetScenes [2]	2007	1280×960	3.5k	Boston, MA, USA	Segmentation	9
KITTI [13]	2012	1245×375	7.5k/-/7.5k	Karlsruhe, Germany	Localization	8
Yu P et al. [30]	2012	1600×1264	4.9k	-	Localization	5
BDD100K [44]	2017	1280×720	84M/36M	NY, SF, California	Localization	10
Mapillary Vistas [29]	2017	1920×1080	18k/2k/5k	6 continents	Segmentation	37
IDD [38]	2019	1920×1080	31k/10k/4k	Hyderabad, Bangalore, India	Segmentation	34
Udacity [37]	2018	1920×1080	15k	-	Localization	11
NITCAD [28]	2019	1172×544	4.8k/2.7k	Kerala	Localization	7
Otonomarac [32]	2023	1920×1080	7.6k	-	Localization	8
JUVDsi v1 [1]	2023	1920×1080	3k	West Bengal, India	Localization	9
IRUVD (our)	2023	1920×1080	4k	West Bengal, India	Localization	14

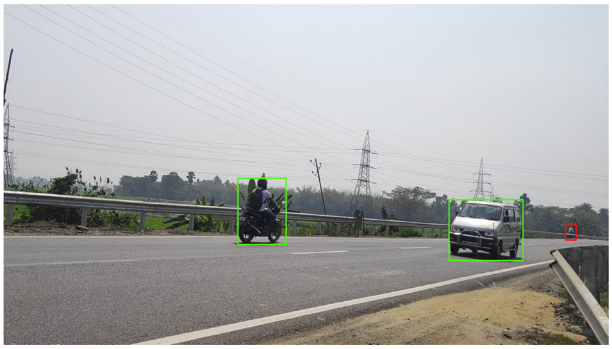

Fig. 2 An example of a difficult annotation problem. The green box shows a straightforward annotation and the red box indicates a difficult annotation

process may be prone to errors, we must make sure that they are kept to a minimum. We struggled to obtain the correct bounding boxes for each object when it was obscured by another. Objects may be regarded as noise because of their small size and light particle illumination. Figure 2 displays a challenging annotation example. Because the object in this example is too small, we have not considered it to be an object. Using the open-source program LabelIMG [36], all of the images have annotations.

3.3 Statistics of the dataset

Because they are the most varied and common on Indian roadways, we selected 13 vehicle categories and one class for pedestrians from the photographs we collected. They are *toto, bike, cyclist, auto-rickshaw, motor-rickshaw, van, tempo, car, bus, taxi, truck, jeep, cycle-rickshaw*, and *pedestrian*. A sample of each category found in the developed dataset is shown in Fig. 3. We took into account every vehicle that was included in datasets like KITTI and NITCAD. Additionally, six new vehicle classes have been added: tempo, taxi, motor-rickshaw, jeep, toto, and cycle-rickshaw. We have annotated 4000 images using the aforementioned process, and a total of 14343 bounding boxes have been labeled manually. Each image in our dataset has an average of 3.58 boxes. Figure 4 shows the frequency of each object. From this figure, it is clear that there are a maximum number of pedestrians and a minimum number of cycle-rickshaws in the dataset.

3.4 Quality of annotation

Thanks to the open-source annotation software LabelImg [36], which made it easier to prepare most accurate annotations. We have annotated the images using this software in accordance with the YOLO format. As seen in Fig. 2, typical situations like occlusion and the small size of the objects, among others, lead to errors in the annotation process. To lessen the uncertainty of the bounding boxes and class labels, we have carefully examined the results.

3.5 Comparison with other vehicle detection datasets

For AVD, a number of datasets have been made available, including CBCL StreetScenes, KITTI, Dataset by Yu P. et al., BDD100k, Mapillary Vistas, IDD, NITCAD, and others.

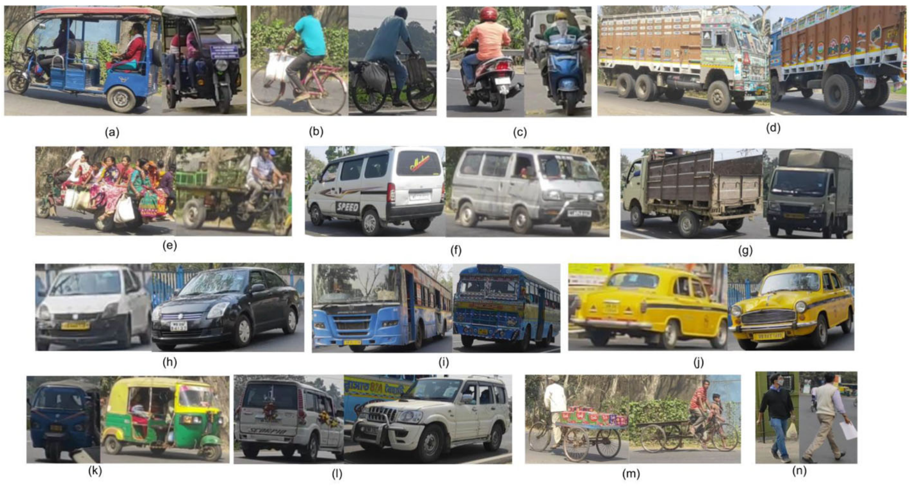

Fig. 3 Examples of different classes present in IRUVD dataset. (a) Toto (b) Cyclist (c) Bike (d) Truck (e) Motor-rickshaw (f) Van (g) Tempo (h) Car (i) Bus (j) Taxi (k) Auto-rickshaw (l) Jeep (m) Cycle-rickshaw (n) Pedestrian

There are mainly two types of datasets: segmentation based datasets and localization based datasets. The dataset developed under the current work can be applied to localization. Our dataset is more comparable to the NITCAD dataset because we have concentrated on Indian vehicle detection and classification. As seen in Table 1, our dataset has 14 classes while NITCAD has only 7 classes. The current dataset has less object classes than datasets like IDD and Mapillary Vistas, but it has more vehicles because it adds new classes like tempo, taxi, motor-rickshaw, jeep, toto, and cycle-rickshaw.

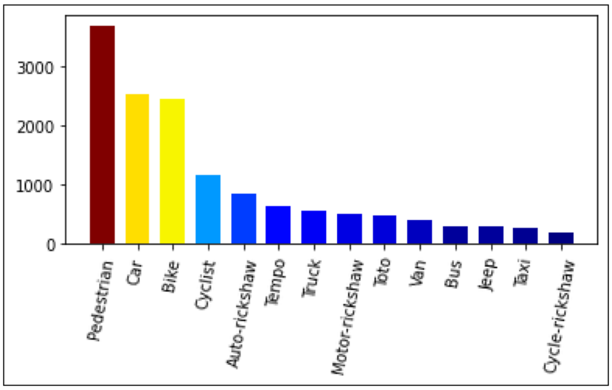

Fig. 4 Distribution of different object classes found in the IRUVD dataset. Y-axis denotes the number of occurrences of each class

4 Dataset benchmarking

Results of experimentation carried out under the current work are reported in this section. For the performance evaluation, a total of 11 methods have been applied on the IRUVD dataset.

4.1 Deep learning models used to benchmark

Several object detection models, including YOLOv3 [31], YOLOv4 [3], Scaled-YOLOv4 [40], YOLOv5 [18], YOLOX [12], YOLOR [41], YOLOv6 [21] and YOLOv7 [42] etc. have been introduced in recent years. In the current work, we have considered four state-of-the-art object detection models: YOLOv3, YOLOv4, Scaled-YOLOv4 and YOLOv5. The four models are discussed below.

4.1.1 YOLOv3

Joseph Redmon and Ali Farhadi's [31] YOLOv3 algorithm is the one of the most widely-used object detection techniques in the field of computer vision. As shown in Fig. 5, the detection process is divided into three steps: feature extraction, bounding box prediction, and class prediction. The Darknet-53 method, used by YOLOv3, for feature extraction consists of 53 convolutional layers. In three different levels, YOLOv3 anticipates the boxes to determine the precise size of the object, or prior. K-means clustering is employed to estimate the box prior. Softmax does not discard overlapping boxes, which is useful for detection in more complicated domains like AVD, so it was used in YOLOv3. Figure 5 displays a typical YOLOv3 architecture.

4.1.2 YOLOv4

Bochkovskiy et al. [3] introduced YOLOv4 as a successor of YOLOv3 to better speed and accuracy in terms of mAP [@0.5:0.05:0.95] and mAP [@0.5]. YOLOv4 consists of four sub-blocks, namely, Backbone, Neck, Dense Prediction block, and Sparse Prediction block as shown in Fig. 6. A typical object detection model takes images as input and finds the features through the convolution layer. For this purpose authors of YOLOv4 used SpineNet [9], CSPResNext50 [39], CSPDarknet53 [39], EfficientNet-B3 [34], VGG16

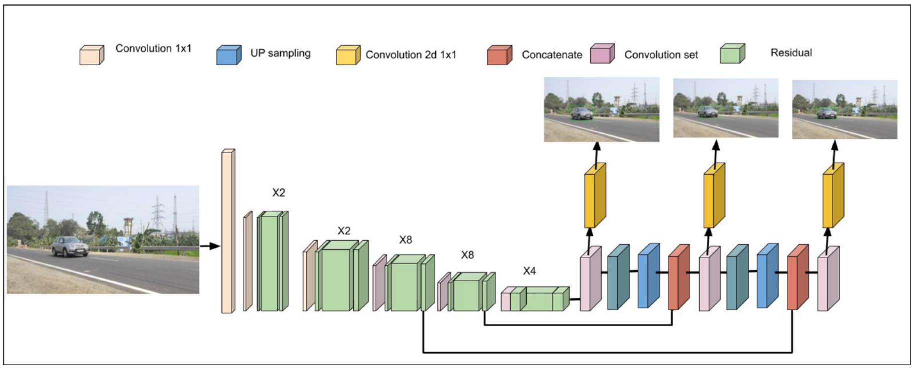

Fig. 5 Schematic diagram of the YOLOv3 object detection model with Darknet-53 backbone

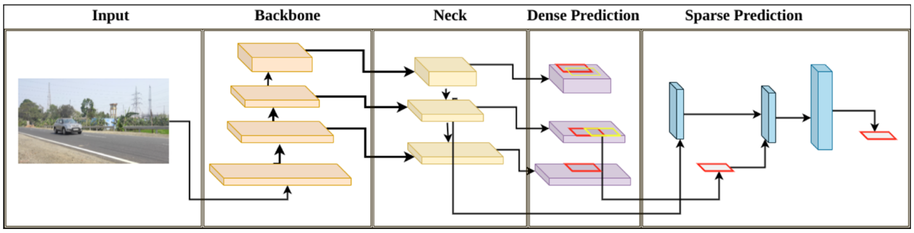

Fig. 6 The architecture of YOLOv4. It has four parts—Backbone, Neck, Dense Prediction, and Sparse Prediction

[33] Backbone for the feature extractor. In this paper, we have used pre-trained CSPDarknet53 on ImageNet as the Backbone. Extracted feature maps are mixed in the Neck region to find more generalized characteristics of the objects. YOLOv4 was examined in a few Neck configurations like Feature Pyramid Network (FPN) [23], Path Aggregation Network (PANet) [24], Neural Architecture Search Feature Pyramid Network (NAS-FPN) [14], Bi-directional Feature Pyramid Network (BiFPN) [35], Adaptively Spatial Feature Fusion (ASFF) [25], Scaled-wise Feature Aggregation Module (SFAM) [47]. PANet is the most commonly used neck structure in YOLOv4. Generally, head sections refer to the Dense Prediction block and the Sparse Prediction block. The Sparse Prediction block is used for two-stage detection, while the Dense Prediction block is used for one-stage detection. The location and class are both estimated during one stage of the detection process. Two-stage detection separates each object's locations and classes. The YOLO head has been used in this essay. The terms "Bag of freebies" and "Bag of specials" are two new techniques introduced in YOLOv4 that improve the model and boost performance. The authors noted that Rectified Linear Unit (ReLU) activation functions did not effectively optimize the features in YOLOv4. The Mish activation function was consequently used for improved performance. The YOLOv4 architecture is demonstrated in Fig. 6.

4.1.3 Scaled-YOLOv4

Bochkovskiy et al. [40] suggested the Scaled-YOLOv4 to detect both small and large objects using the same model. Scaled-YOLOv4 includes CSPDarknet53, CSPUp sampling block, and CSPDown sampling block, as demonstrated in Fig. 7. In addition to the detection bandwidth, both large and small models can use up and down-sampling without sacrificing speed or accuracy. This approach led to the development of two models, tiny-YOLOv4 and large-YOLOv4, which offered cutting-edge outcomes for both small and large object detection models in terms of mAP scores. Scaled-YOLOv4 has a backbone in the form of CSPDarknet53. To lessen the computation at the model's neck, the PAN architecture on Scaled-YOLOv4 employs the CSP-ize technique. By using this technique, the computation is slashed by about 40%. The neck also makes use of CSPSSP. It is clear that YOLOv4's training processes heavily rely on data augmentation. In the scaled YOLOv4, the model is only fine-tuned by adding training data after the training is complete. Additionally, it aids in accelerating the training process.

4.1.4 YOLOv5

Glenn Jocher et al. [18] came up with the idea for YOLOv5. It has three main sub-blocks, including Backbone, PANet Head, and Output block, like other YOLO models, as shown

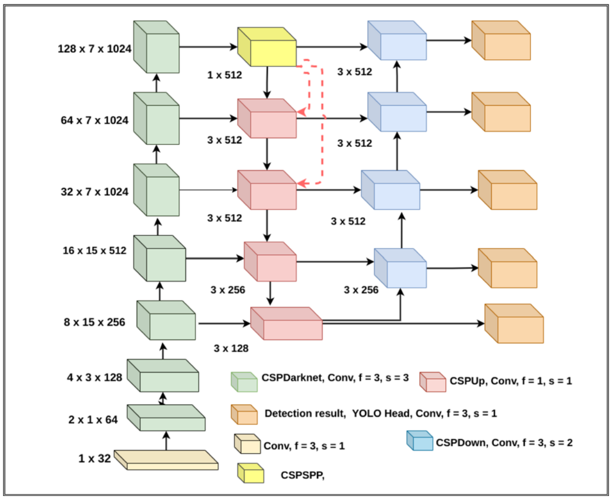

Fig. 7 Schematic diagram of the Scaled-YOLOv4 object detection model. Red lines represent the CSPUP block are replaced by the CSPSPP block for scaling

in Fig. 8. BottleNeckCSP is used to organize YOLOv5's backbone into Darknet53, also known as CSPDarknet53. For feature extraction and feature dimension reduction, a large-scale backbone like CSPDarknet53 works best. This improves the model's detection speed and accuracy. A PANet is present in the model's second component. The model's information flow from the backbone to the head is improved by PANet neck. A bottom-up method is used to pass information through the feature pyramid, allowing the model to recognize more low-level features. The model propagates the features to other layers with the aid of skip connections. Last but not least, the YOLO head is used by the YOLOv5's head or output section. Similar to YOLOv3, YOLOv5 forecasts the output in three distinct scales, namely $(72 \times 72, 36 \times 36, \textit{and } 18 \times 18)$, enabling the model to recognize objects of various sizes. Four distinct models based on trainable parameters are available in YOLOv5.

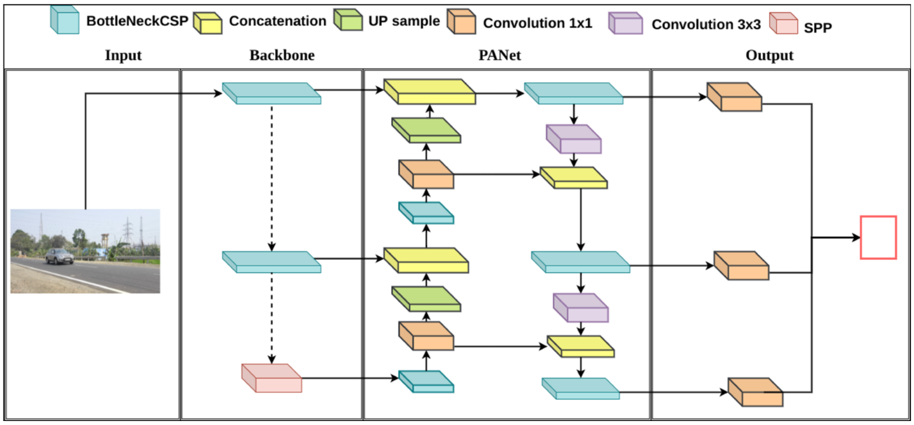

Fig. 8 An illustration of YOLOv5s architecture, with CSP Backbone, PANet Neck, and YOLO head

- YOLOv5s (small)
- YOLOv5m (medium)
- YOLOv5l (large)
- YOLOv5x (extra-large)

In this paper, we have used YOLOv5s to benchmark the results on the IRUVD dataset.

4.2 Object detection benchmarks

As mentioned earlier, we have reported object detection benchmarks using state-of-the-art deep learning models. Class-wise mAP scores from various detection models at an IOU threshold of 0.5 to 0.95 in steps of 0.05 are displayed in Table 2. For the evaluation of each model, we have used a 3-fold cross-validation scheme in order to get more accurate results. This table clearly shows that the detection models classify toto, truck, bus, and motor-rickshaw with greater accuracy. However, we have found that the accuracy of object detection is lower for cars, taxis, bicycles, and pedestrians. Since the shapes of a car and a taxi are almost identical but their colors differ, the object detection models occasionally fail to classify them properly. In Tables 3 and 4 we have shown the class-specific mAP at IOUs of 0.5 and 0.75. The average precision, recall, F1-score, and mAP score at IOU thresholds of 0.5, 0.75, and 0.5 to 0.95 in steps of 0.05 are displayed in Table 5. YOLOv5s yields the highest score for precision. In contrast, Scaled-YOLOv4 yields the highest rating for recall. The Scaled-YOLOv4 model yields the highest overall ratings.

4.3 Confusion matrix and miss rate

The confusion matrix for the YOLOv5s model on the current AVD dataset is shown in Fig. 9. As can be seen, every object has been detected almost perfectly. However, because to

Table 2 Performance comparison on the IRUVD dataset, class-wise mAP scores at IOU threshold ranging from 0.5 to 0.95 in steps of 0.05(mAP@[0.5:0.05:0.95])

Model / Class	YOLOv3	YOLOv4	Scaled-YOLOv4	YOLOv5s
Auto-rickshaw	0.618	0.686	0.752	0.74
Bike	0.672	0.733	0.794	0.774
Bus	0.720	0.787	0.887	0.82
Car	0.657	0.719	0.788	0.76
Cycle-rickshaw	0.714	0.827	0.894	0.78
Cyclist	0.584	0.658	0.705	0.69
Motor-rickshaw	0.750	0.860	0.912	0.83
Pedestrian	0.534	0.634	0.671	0.63
Taxi	0.589	0.664	0.744	0.71
Tempo	0.704	0.780	0.814	0.77
Toto	0.797	0.865	0.907	0.88
Truck	0.79	0.864	0.937	0.87
Van	0.760	0.858	0.8819	0.839
Jeep	0.719	0.779	0.850	0.80

Table 3 Performance comparison on the IRUVD dataset, class-wise mAP scores at IOU threshold 0.5(mAP@0.5)

Class \ Model	YOLOv3	YOLOv4	Scaled-YOLOv4	YOLOv5s
Auto-rickshaw	0.898	0.852	0.910	0.916
Bike	0.939	0.918	0.958	0.954
Bus	0.942	0.931	0.973	0.953
Car	0.894	0.863	0.913	0.917
Cycle-rickshaw	0.930	0.944	0.958	0.902
Cyclist	0.859	0.837	0.852	0.878
Motor-rickshaw	0.971	0.971	0.980	0.961
Pedestrian	0.852	0.859	0.893	0.887
Taxi	0.804	0.841	0.883	0.898
Tempo	0.90	0.919	0.918	0.895
Toto	0.991	0.959	0.975	0.974
Truck	0.992	0.977	0.988	0.975
Van	0.932	0.931	0.920	0.908
Jeep	0.930	0.891	0.923	0.907

their similar shapes, as already indicated, taxis are sometimes mistaken as cars. Following are a few erroneous classifications between some classes:

- Car and Taxi
- Cyclist and Pedestrian

Table 4 Performance comparison on the IRUVD dataset, class-wise mAP scores at IOU threshold 0.75(mAP@0.75)

Class \ Model	YOLOv3	YOLOv4	Scaled-YOLOv4	YOLOv5s
Auto-rickshaw	0.710	0.805	0.833	0.850
Bike	0.804	0.864	0.888	0.903
Bus	0.912	0.902	0.958	0.917
Car	0.789	0.820	0.869	0.841
Cycle-rickshaw	0.888	0.944	0.958	0.902
Cyclist	0.683	0.782	0.786	0.817
Motor-rickshaw	0.903	0.971	0.963	0.934
Pedestrian	0.603	0.766	0.793	0.769
Taxi	0.713	0.767	0.822	0.796
Tempo	0.859	0.881	0.884	0.870
Toto	0.935	0.951	0.967	0.974
Truck	0.981	0.972	0.983	0.934
Van	0.932	0.931	0.920	0.896
Jeep	0.931	0.882	0.907	0.889

Table 5 Performance comparison on the IRUVD dataset using different state-of-the-art object detection models

Method	Precision	Recall	F1 score	mAP @0.5	mAP @0.75	mAP @0.5:0.05:0.95
YOLOv3 [31]	0.913	0.92	0.916	0.917	0.827	0.687
YOLOv4 [3]	0.90	0.937	0.917	0.944	0.874	0.765
Scaled-YOLOv4 [40]	0.786	0.962	0.812	0.932	0.895	0.814
YOLOv5s [18]	0.946	0.919	0.931	0.924	0.878	0.782

– Toto and Auto-rickshaw
– Cycle-rickshaw and Cyclist
– Cyclist and Motor-Bike

We have shown the total number of the true positive and false positive objects detected by each model. We have presented class-wise log average miss rate for each model to see how the model works. Figure 10(a) shows the total number of true positive and false positive objects detected by the YOLOv3 model for each class, whereas (b) shows the log average miss rate of the YOLOv3 model for each class. Figures 11, 12 and 13 show the similar results produced by Scaled-YOLOv4, YOLOv5 and YOLOv4 models, respectively.

4.4 Precision vs. recall curve

Precision measures the percentage of true positives, whereas recall measures the percentage of false negatives. Low false negative rates indicate a high recall value, and high true positive rates indicate a high precision value. A perfect model needs to be highly accurate and highly reliable. The precision-recall trade-off, on the other hand, states that as precision increases, recall decreases, and vice versa. For each of the 14 classes, the precision vs. recall curve is displayed in Fig. 14. This graph shows that we have achieved encouraging results for the bus, cycle-rickshaw, toto, van, and motor-rickshaw. We have received subpar results for some classes, including auto-rickshaw, pedestrian, taxi, cyclist and jeep, and this indicates a precision-recall trade-off.

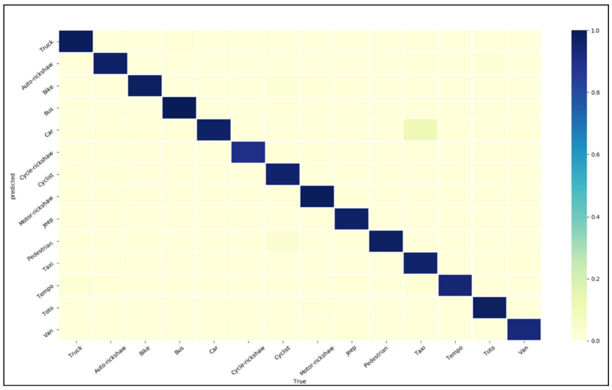

Fig. 9 Confusion matrix on the current AVD dataset using YOLOv5s

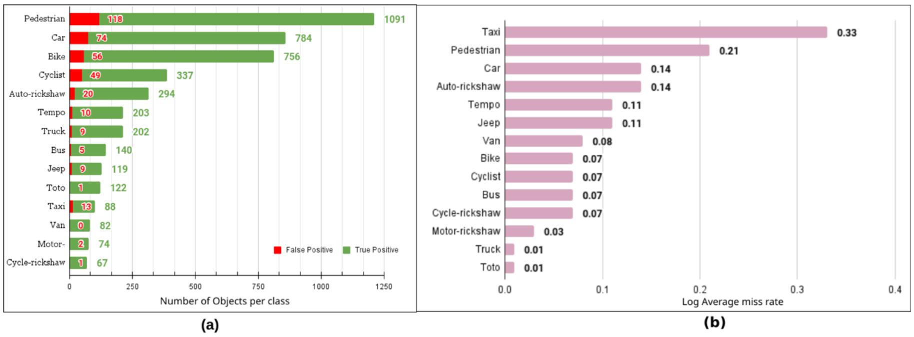

Fig. 10 Vehicle detection results produced by YOLOv3: (a) Number of false positives and true positives for each class, and (b) Log Average miss rate of each class

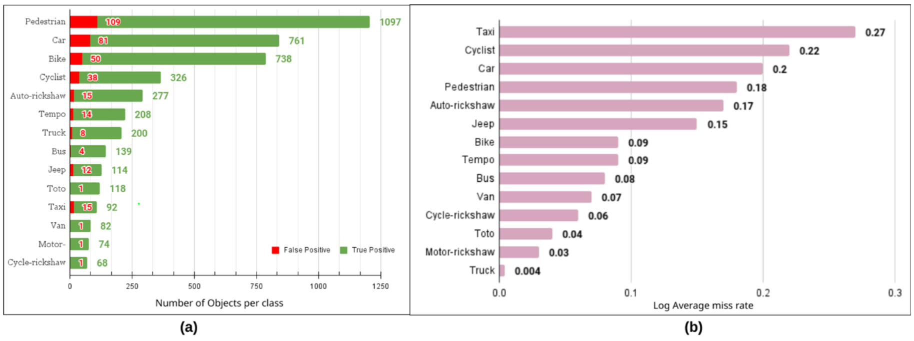

Fig. 11 Vehicle detection results produced by YOLOv4: (a) Number of false positives and true positives for each class, and (b) Log Average miss rate of each class

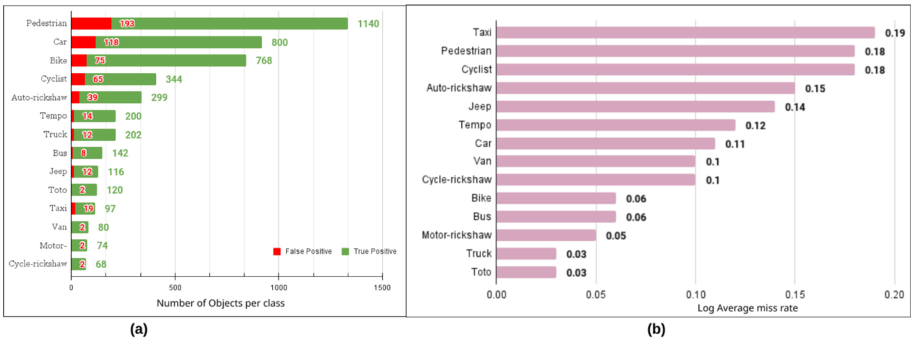

Fig. 12 Vehicle detection results produced by Scaled-YOLOv4: (a) Number of false positives and true positives for each class, and (b) Log Average miss rate of each class

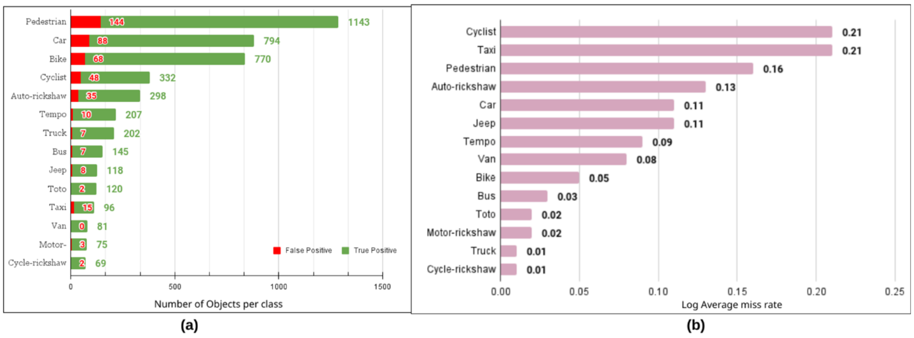

Fig. 13 Vehicle detection results produced by YOLOv5: (a) Number of false positives and true positives for each class, and (b) Log Average miss rate of each class

5 Ensemble techniques

The proposed ensemble method, which is actually a box estimation technique used to predict the precise bounding box from the N number of bounding boxes provided by the same number of detection algorithms, has been discussed in this section. Even though a single object detection model locates objects fairly accurately, it occasionally classifies the background as an object or becomes confused with objects that have a similar shape. We have suggested an ensemble technique, which is described below, to address this problem.

- Consider the use of N detection models to create the aforementioned ensemble. At a specific IOU threshold T, each model provides a prediction score for the detected object. In our situation, we have established the limit $T >= 0.5$. The box that the N^{th} model returns is designated as B_N.
- Each box includes the object's height and width as well as the class prediction, prediction confidence, and center coordinates (x, y). The algorithms count how many models classified an object as belonging to the same class for a given object out of N predictions for that class. The class that the majority of models select is the actual class of the object. Now, the true class is estimated using the prediction confidence scores if the same number of models correctly predict an object that belongs to multiple classes. The model with the highest confidence score is considered to have provided the final class for an object. For instance, if four models are used for assembly and three of them identify an object as a car and one as a taxi, our suggested method will label the object as a car. The method considers an object to be a car if two models predict it to be a car and the other two models predict it to be a taxi, but the first two models have the highest confidence scores. False predictions can be decreased by using this method.
- Our method does not remove overlapping anchor boxes like NMS and Soft-NMS. However, it estimates new anchor box from N number of boxes obtained from different detection models. We propose a technique to estimate the box using a weighted method or a non-weighted method. Let us consider that N number of detection models gives N number of predictions for a particular object as $[C_1, S_1, x_1, y_1, w_1, h_1]$, ..., $[C_N, S_N, x_N, y_N, w_N, h_N]$, where C_N = class predicted by N^{th} model, S_N = confidence score given by N^{th} model, x_N = x coordinate given by N^{th} model, y_N = y coordinate given by N^{th} model, w_N = width given by N^{th} model, h_N = height given by N^{th} model. Then parameters are estimated using (1)—(7). The

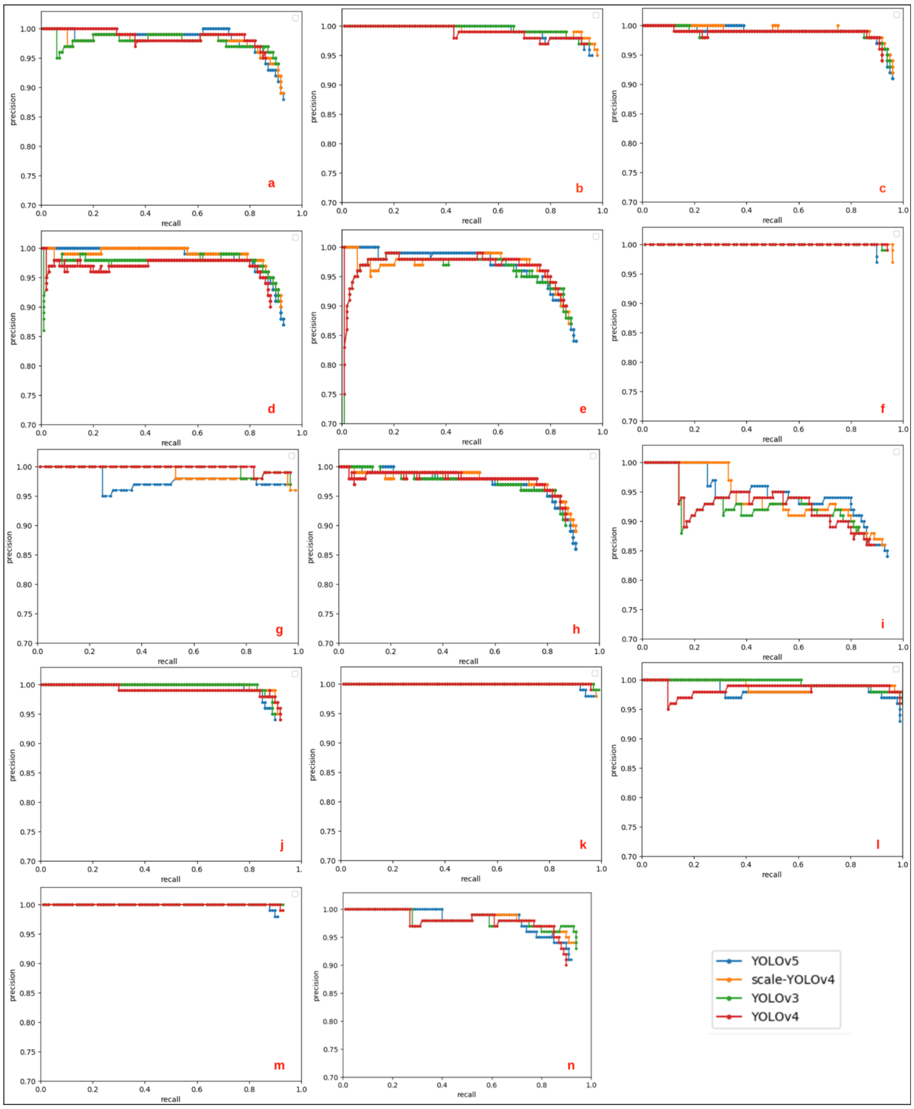

Fig. 14 Precision vs. recall plots of 14 object classes using YOLOv3, YOLOv4, Scaled-YOLOv4 and YOLOv5 object detection models. (a) Auto-rickshaw, (b) Bus, (c) Bike, (d) Car, (e) Cyclist, (f) Cycle-rickshaw, (g) Motor-rickshaw, (h) Pedestrian, (i) Taxi, (j) Tempo, (k) Toto, (l) Truck, (m) Van, and (n) Jeep

weighted method takes P_1, P_2, ..., P_N as inputs, where P are the parameter which we want to estimate, i.e., x coordinate, y coordinate, width, height, etc., and N is the number of models used in the ensemble technique. On the other hand, P_1, P_2, ..., P_N , W_1, W_2, ..., W_N are used as inputs for the non-weighted method, where P is x coordinate, y coordinate, width, height, and W_N is the confidence score of N^{th} model.

We have considered the confidence score as the weight in weighted methods. To estimate the parameters, we have used seven different popular functions that are described below:

- **Non-weighted methods:**
  1. **Mean:**
$$f(P_1, P_2, ..., P_N) = \frac{\sum_{i=1}^{N} P_i}{N} \tag{1}$$
  2. **Harmonic Mean (HM):**
$$f(P_1, P_2, ..., P_N) = \frac{N}{\sum_{i=1}^{N} \frac{1}{P_i}} \tag{2}$$
  3. **Contraharmonic Mean (CM):**
$$f(P_1, P_2, ..., P_N) = \frac{\sum_{i=1}^{N} {P_N}^2}{\sum_{i=1}^{N} P_N} \tag{3}$$
  4. **Root Mean Square (RMS):**
$$f(P_1, P_2, ..., P_N) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} {P_N}^2} \tag{4}$$
- **Weighted methods:**
  1. **Weighted Mean (WM):**
$$f(P_1, .., P_N, W_1, .., W_N) = \frac{\sum_{i=1}^{N} W_i P_i}{\sum_{i=1}^{N} W_i} \tag{5}$$
  2. **Weighted Harmonic Mean (WHM):**
$$f(P_1, .., P_N, W_1, .., W_N) = \frac{\sum_{i=1}^{N} W_i}{\sum_{i=1}^{N} \frac{W_i}{P_i}} \tag{6}$$
  3. **Weighted Geometric Mean (WGM):**
$$f(P_1, .., P_N, W_1, .., W_N) = \frac{\sum_{i=1}^{N} {P_N}^2}{\sum_{i=1}^{N} P_N} \tag{7}$$

Using (1)–(7), all the said parameters are estimated for the newly obtained bounding boxes. An illustration of our ensemble architecture is shown in Fig. 15. Tables 6 and 7 show results of an ensemble of different models using various methods in terms of mAP at IOU threshold 0.5, mAP at IOU threshold 0.75, mAP at IOU threshold ranging from 0.5 to 0.95 in steps of 0.05. We have combined two and more models to form different ensembles. From Table 6, it can be seen that the combination of YOLOv3 and YOLOv4 using the Mean method achieves 1% improvement on mAP@[0.5:0.05:0.95] than YOLOv4 and 8.8% than YOLOv3. Popular techniques like NMS and soft-NMS achieve only 3.8% more than YOLOv3, but 5% less than YOLOv4. Ensemble of YOLOv4 and Scaled-YOLOv4 using the Mean method gives an 83.6% mAP@[0.5:0.05:0.95] which is 7.1% more than YOLOv4 and 2.2% more than Scaled-YOLOv5. Other methods give similar performance, however, NMS and soft-NMS give less mAP than Scaled-YOLOv4 because of the unnecessary elimination of overlapping boxes. Fusion of YOLOv4 and YOLOv5 provides an 82.3% of mAP@[0.5:0.05:0.95], which is 1.5% higher than Scaled-YOLOv4 and 4.7% higher than

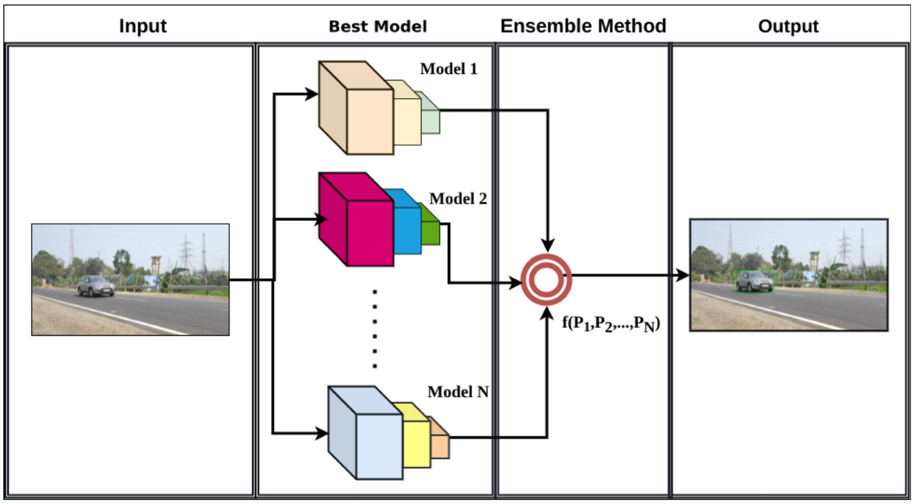

Fig. 15 An illustration of the ensemble method used in the present work which considers N number of base models

YOLOv5s. When we have used an ensemble of YOLOv5 and Scaled-YOLOv4 models, NMS and S-NMS display better results than other methods, which is a 5% improvement than YOLOv5 and 1.9% improvement than the Scaled-YOLOv4. Not only two models, but more than two models are also used to form the ensemble, as shown in Table 7. Combinations of YOLOv3, YOLOv4, Scaled-YOLOv4 and YOLOv3, YOLOv4, YOLOv5 provide better results than a single model but do not provide better results than the combination of two models. However, an ensemble of YOLOv4, Scaled-YOLOv4, and YOLOv5 models using the Mean method shows higher performance than all other models. An illustration of a bar plot of mAP@[0.5:0.05:0.95] by the ensemble of different models using different methods for each class is shown in Fig. 16. A number of false positives, true positives and log average miss rate of each class given by the ensemble of YOLOv4, Scaled-YOLOv4, and YOLOv5 using the Mean method are shown in Fig. 17. In Fig. 18, a comparison of class-wise mAP at different thresholds from 0.5 to 0.95 is presented. We have used YOLOv3, YOLOv4, Scaled-YOLOv4, and YOLOv5, an ensemble of YOLOv4, Scaled-YOLOv4 using Mean method and ensemble of YOLOv4, Scaled-YOLOv4, YOLOv5 using Mean method. From Fig. 18 it can be clearly seen that YOLOv3 has the worst performance, whether as the ensemble of YOLOv4, Scaled-YOLOv4, YOLOv5 using the Mean method outperforms other models.

6 Experimentation on other datasets

In this paper, we have presented a dataset for vehicle detection on Indian roads and benchmarked the results using four state-of-the-art deep learning-based detection models. We have also proposed an ensemble technique for the improvement of the benchmark results on the developed IRUVD dataset. For better understanding of how a model trained on existing datasets fails to detect a vehicle in the complex situation on Indian roads, we have tested the

Table 6 Performance comparison of the ensemble of two models using different methods on the IRUVD dataset

Model	Method	mAP @0.5	mAP @0.75	mAP @0.5:0.05:0.95
YOLOv3 + YOLOv4	Mean	0.940	0.89	0.775
	WGM	0.940	0.891	0.770
	WM	0.940	0.897	0.775
	NMS	0.942	0.875	0.721
	Soft-NMS	0.944	0.881	0.725
YOLOv4 + Scaled-YOLOv4	Mean	0.943	**0.914**	**0.836**
	WGM	0.944	0.911	0.831
	RMS	0.944	0.913	**0.835**
	HM	0.938	0.904	0.822
	WHM	0.941	0.908	0.828
	WM	0.944	0.914	**0.834**
	NMS	0.944	0.906	0.794
	Soft-NMS	**0.946**	0.912	0.797
YOLOv4 + YOLOv5	Mean	**0.947**	**0.916**	0.823
	WGM	**0.945**	0.912	0.821
	RMS	**0.947**	**0.915**	0.822
	HM	**0.945**	0.910	0.820
	WHM	**0.946**	0.912	0.828
	WM	**0.947**	**0.915**	0.824
	CM	0.930	0.898	0.806
	NMS	0.944	0.905	0.793
	Soft-NMS	**0.947**	0.909	0.796
YOLOv5 + Scaled-YOLOv4	Mean	0.933	0.910	0.829
	WGM	0.944	0.906	0.822
	RMS	0.944	0.909	0.828
	HM	0.943	0.905	0.818
	WHM	0.944	0.908	0.824
	WM	0.944	0.910	0.827
	NMS	0.942	0.907	0.832
	Soft-NMS	0.944	0.907	0.833

The 1^{st}, 2^{nd}, and 3^{rd} score results are indicated in red, blue, and green fonts respectively

YOLOv5 model trained on two datasets, namely Udacity Self-Driving-Car [37] and Otonomarc [32]. Comparative results on the IRUVD dataset are shown in Table 8. In terms of precision, recall, F1-score, and mAP@0.5, the model trained on the proposed dataset outperforms the models trained on the Udacity Self-Driving-Car and Otonomarc datasets. We have analysed predictions of the YOLOv5 model trained on the Udacity Self-Driving-Car, Otonomarc, and IRUVD datasets, as shown in Fig. 19. From Fig. 19, it is clear that the model trained on IRUVD gives the best result, whereas the model trained on the Udacity Self-Driving-Car model shows the worst performance.

Table 7 Performance comparison of the ensemble of three models using different methods on the IRUVD dataset

Model	Method	mAP @0.5	mAP @0.75	mAP @0.5:0.05:0.95
YOLOv3 + YOLOv4 + Scaled-YOLOv4	Mean	0.942	0.907	0.815
	WGM	0.943	0.903	0.809
	RMS	0.942	0.906	0.810
	HM	0.940	0.895	0.803
	WHM	0.940	0.898	0.805
	WM	0.942	0.907	0.812
	NMS	**0.946**	0.877	0.724
	Soft-NMS	**0.948**	0.885	0.728
YOLOv3 + YOLOv4 + YOLOv5	Mean	0.938	0.902	0.803
	WGM	0.937	0.898	0.797
	RMS	0.938	0.901	0.799
	HM	0.936	0.893	0.793
	WHM	0.936	0.897	0.796
	WM	0.938	0.902	0.801
	NMS	**0.946**	0.877	0.724
	Soft-NMS	**0.948**	0.882	0.728
YOLOv4 + Scaled-YOLOv4 + YOLOv5	Mean	**0.946**	0.920	**0.841**
	WGM	**0.946**	**0.917**	**0.835**
	RMS	**0.947**	**0.920**	**0.838**
	HM	**0.946**	0.915	0.829
	WHM	**0.946**	**0.918**	**0.835**
	WM	**0.946**	**0.920**	**0.838**
	CM	0.932	0.902	0.823
	NMS	**0.947**	0.908	0.796
	Soft-NMS	0.950	0.913	0.799

The 1st, 2nd, and 3rd score results are indicated in red, blue, and green fonts respectively

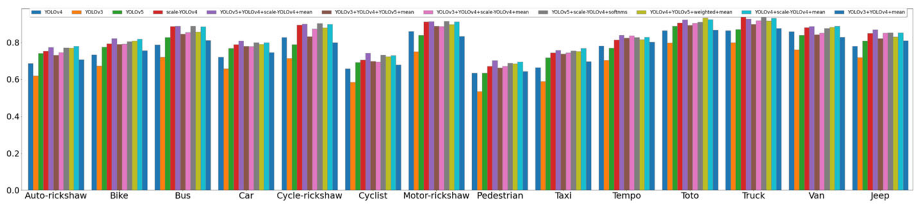

Fig. 16 A comparison of class-wise mAP@[0.5:0.05:0.95] of detection results using YOLOv3, YOLOv4, Scaled-YOLOv4, YOLOv5 and top seven the ensemble models in terms of mAP@[0.5:0.05:0.95] shown in Tables 6 and 7 on the IRUVD dataset

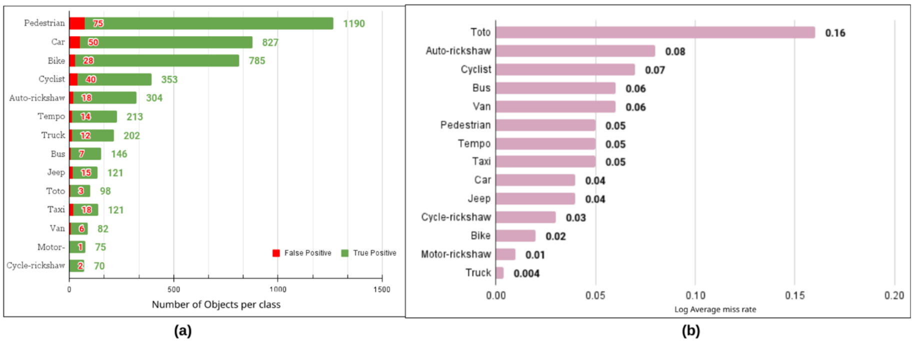

Fig. 17 Detection results produced by the ensemble of YOLOv4, Scaled-YOLOv4, and YOLOv5 using the Mean method: (a) Number of false positives and true positives of each class, and (b) Log average miss rate of each class

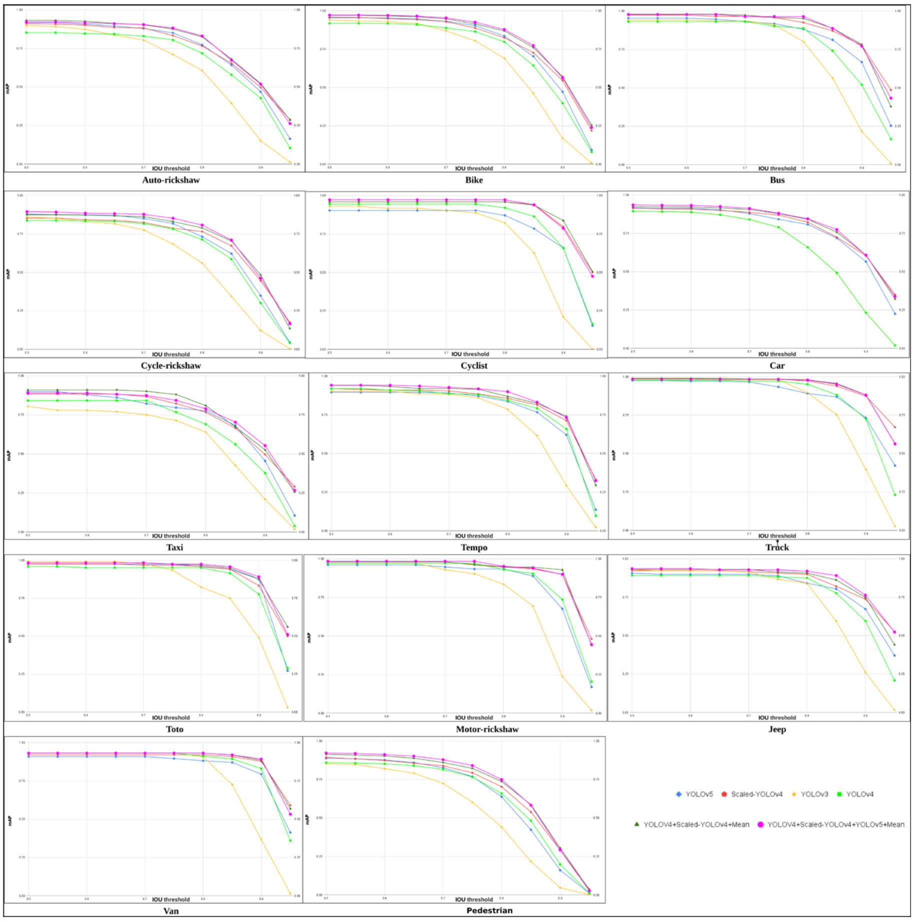

Fig. 18 mAP score plots given by YOLOv3, YOLOv4, Scaled-YOLOv4, YOLOv5 and the ensemble of YOLOv4 and Scaled-YOLOv4 using the Mean method, and the ensemble of YOLOv4, Scaled-YOLOv4 and YOLOv5 using the Mean method for each class

Table 8 Performance comparison of YOLOv5 model trained on IRUVD dataset (our) and separately tested on Udacity Self-Driving-Car, Otonomarc and IRUVD (our) datasets

Dataset	Precision	Recall	F1 score	mAP @0.5
Otonomarac	0.471	0.469	0.470	0.432
Udacity Self-Driving-Car	0.411	0.283	0.335	0.327
IRUVD (Our)	**0.946**	**0.919**	**0.931**	**0.924**

The best scores are indicated in bold fonts

We have also evaluated our proposed ensemble of YOLOv4 and YOLOv5 models on Udacity Self-Driving-Car and Otonomarc datasets, and we have obtained 31.05% and 13.29% mAP@0.5 improvements, respectively. Results of this testing are shown in Table 9. From the table, it is observed that in terms of mAP@0.5:0.05:0.95, the proposed ensemble method outperforms the base models, thereby ensuring the effectiveness of the ensemble method for the problem under consideration.

7 Conclusion and future scope

Autonomous vehicles, intelligent traffic management systems, and other technologies have largely taken over our daily lives. Traffic management is becoming more and more challenging as vehicles proliferate, especially in crowded nations like India. Nowadays, deep learning models are primarily used by researchers to build an effective AVD system, which requires a large amount of data. Several datasets that are freely available for this purpose have been found in the literature. However, the majority of them only take into account the traffic patterns and vehicles that are frequently seen on urban roads, making them

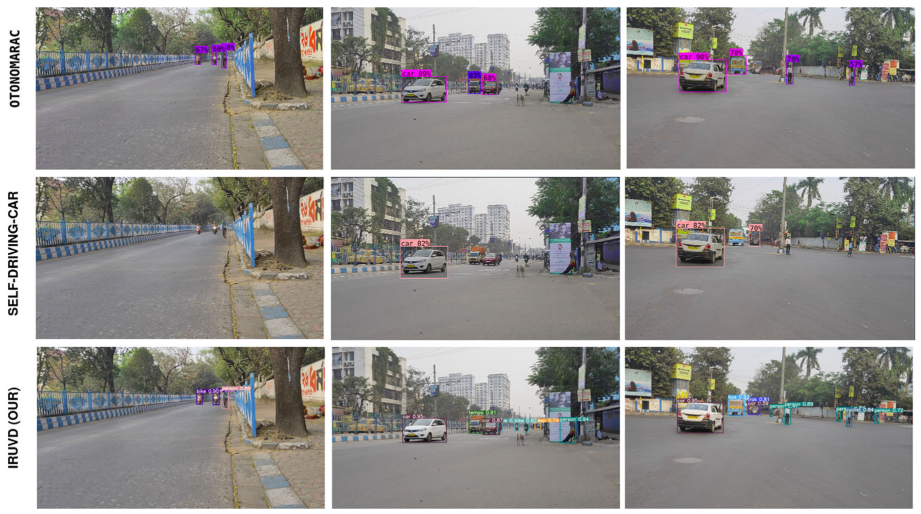

Fig. 19 Qualitative results of YOLOv5 model trained on Udacity Self-Driving-Car, Otonomarc and IRUVD (our) datasets

Table 9 Performance comparison of the ensemble of two models using different methods on Otonomarc and Udacity Self-Driving-Car datasets

Dataset		Otonomarc		Udacity Self-Driving-Car	
Model	Method	mAP @0.5	mAP @0.5:0.05:0.95	mAP @0.5	mAP @0.5:0.05:0.95
YOLOv4	–	0.395	0.268	0.602	0.532
YOLOv5	–	0.412	0.312	0.621	0.558
YOLOv4 + YOLOv5	Mean	**0.443**	**0.332**	0.629	0.561
	WGM	0.415	**0.332**	**0.638**	**0.567**
	RMS	0.419	0.324	0.627	0.553
	HM	0.422	**0.328**	0.623	0.553
	WHM	0.422	**0.332**	**0.641**	0.562
	WM	**0.441**	**0.327**	**0.633**	**0.573**
	CM	**0.425**	**0.332**	**0.633**	**0.572**
	NMS	0.415	0.317	0.623	0.562
	Soft-NMS	**0.443**	**0.327**	**0.641**	**0.573**

The 1st, 2nd, and 3rd score results are indicated in red, blue, and green fonts, respectively

less useful for creating a comprehensive traffic management system. To this end, in this paper, we have developed a 14-class IRUVD dataset. Many vehicle classes that are frequently seen in rural areas were not taken into account in past datasets. New vehicle classes like the toto, motor-rickshaw, tempo, taxi, and cycle-rickshaw have been included in the dataset. With 14343 popper annotations, we have 4000 high-quality images to offer. Four state-of-the-art deep learning-based object detection models, namely YOLOv3, YOLOv4, Scaled-YOLOv4, and YOLOv5, have been used to benchmark the results on the said dataset. Additionally, we have suggested weighted and non-weighted ensemble techniques that improve mAP@[0.5:0.05:0.95] by 1.9%. Despite our best efforts, some classes like cycle-rickshaw, jeep, taxi, bus, etc. have a very small number of samples. This may result in overfitting of deep learning models. Hence to address this, we may employ several types of data augmentation techniques in future. However, in our research, we have observed that current models operate efficiently without applying any data augmentation techniques because the YOLO models use the focal loss for training, which can deal with imbalance data. [46]. On the other hand ensemble approaches surpass existing models in terms of the performance metrics under consideration. Another gap in our dataset is that we have not consider varied weather conditions such as wet, hazy, or overcast days, as well as the time of day such as evening or night. We would like to resolve these constraints in our future attempts. Our dataset can be expanded by collecting data from new types such as animals, signboards, and so on. It is necessary to conduct research on the development of unique architectures capable of detecting and classifying in a wide range of settings, including numerous edge cases. Our another future plan is to build a video dataset in Indian context for AVD purposes.

Acknowledgements A part of this work is funded by DST (ICPS) CPS-Individual/2018/403(G).

Data Availability The developed dataset can be found at the GitHub repository: https://github.com/IRUVD/IRUVD.git.

Declarations

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

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

References

1. Bhattacharyya A, Bhattacharya A, Maity S, Singh P, Sarkar R (2023) Juvdsi v1: developing and benchmarking a new still image database in Indian scenario for automatic vehicle detection. Multimed Tools Appl 1–33
2. Bileschi SM, Wolf L (2006) Cbcl streetscenes. Technical report
3. Bochkovskiy A, Wang C-Y, Liao H-YM (2020) Yolov4: optimal speed and accuracy of object detection. arXiv:2004.10934
4. Bodla N, Singh B, Chellappa R, Davis LS (2017) Improving object detection with one line of code. arXiv:1704.04503
5. Cordts M, Omran M, Ramos S, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The cityscapes dataset for semantic urban scene understanding. arXiv:1604.01685
6. Deng J, Dong W, Socher R, Li L-J, Li K, Li F-F (2009) Imagenet: a large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, pp 248–255
7. Dollár P, Wojek C, Schiele B, Perona P (2009) Pedestrian detection: a benchmark. In: 2009 IEEE conference on computer vision and pattern recognition. IEEE, pp 304–311
8. Dong Z, Wu Y, Pei M, Jia Y (2015) Vehicle type classification using a semisupervised convolutional neural network. IEEE Trans Intell Transp Syst 16(4):2247–2256
9. Du X, Lin T-Y, Jin P, Ghiasi G, Tan M, Cui Y, Le QV, Song X (2019) Spinenet: learning scale-permuted backbone for recognition and localization. arXiv:1912.05027
10. Everingham M, Van Gool L, Williams CKI, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. Int J Comput Vis 88(2):303–338
11. Ershad SF (2013) Developing feature representation and respected innovative database collecting algorithm for texture analysis 11
12. Ge Z, Liu S, Wang F, Li Z, Sun J (2021) Yolox: exceeding yolo series in 2021. arXiv:2107.08430
13. Geiger A, Lenz P, Urtasun R (2012) Are we ready for autonomous driving? The kitti vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition. IEEE, pp 3354–3361
14. Ghiasi G, Lin T-Y, Pang R, Le QV (2019) NAS-FPN: learning scalable feature pyramid architecture for object detection. arXiv:1904.07392
15. India tops the world with 11% of global death in road accidents: World bank report. shorturl.at/goEXZ, February 2021
16. Jian M, Qi Q, Dong J, Yin Y, Lam K-M (2018) Integrating qdwd with pattern distinctness and local contrast for underwater saliency detection. J Vis Commun Image Represent 53:31–41
17. Jian M, Qi Q, Yu H, Dong J, Cui C, Nie X, Zhang H, Yin Y, Lam K-M (2019) The extended marine underwater environment database and baseline evaluations. Appl Soft Comput 80:425–437
18. Jocher G, Stoken A, Borovec J, NanoCode012, Chaurasia A, TaoXie, Liu C., Abhiram V, Laughing, tkianai, yxNONG, Hogan A, Mammana L, AlexWang1900, Hajek J, Diaconu L, Marc Y, Kwon O, Wanghaoyang0106, Defretin Y, Lohia A, ml5ah, Milanko B, Fineran B, Khromov D, Ding Y, Doug D, Ingham F (2021) ultralytics/yolov5: v6.0 - YOLOv5n ‘Nano’ models, Roboflow integration, TensorFlow export, OpenCV DNN support
19. Khosravi H, Gholamalinejad H (2020) Irvd: a large-scale dataset for classification of iranian vehicles in urban streets 06
20. Krause J, Stark M, Deng J, Li F-F (2013) 3d object representations for fine-grained categorization. In: 2013 IEEE International conference on computer vision workshops, pp 554–561
21. Li C, Li L, Jiang H, Weng K, Geng Y, Li L, Ke Z, Li Q, Cheng M, Nie W et al (2022) Yolov6: a single-stage object detection framework for industrial applications. arXiv:2209.02976
22. Lin T-Y, Maire M, Belongie SJ, Bourdev LD, Girshick RB, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft COCO: common objects in context. arXiv:1405.0312
23. Lin T-Yi, Dollár P, Girshick RB, He K, Hariharan B, Belongie SJ (2016) Feature pyramid networks for object detection. arXiv:1612.03144
24. Liu S, Lu Q, Qin H, Shi J, Jia J (2018) Path aggregation network for instance segmentation. In: 2018 IEEE/CVF conference on computer vision and pattern recognition, pp 8759–8768

25. Liu S, Di H, Wang Y (2019) Learning spatial fusion for single-shot object detection. arXiv:1911.09516
26. Maity S, Bhattacharyya A, Singh P, Kumar M, Sarkar R (2022) Last decade in vehicle detection and classification: a comprehensive survey. Arch Comput Methods Eng 1–38
27. Munder S, Gavrila DM (2006) An experimental study on pedestrian classification. IEEE Trans Pattern Anal Mach Intell 28(11):1863–1868
28. Namburi S, Joseph A, Umamaheswaran S, Priyanka Ch, Malavika M, Sankaran P (2020) Nitcad—developing an object detection, classification and stereo vision dataset for autonomous navigation in indian roads. Procedia Comput Sci 171:207–216 (01)
29. Neuhold G, Ollmann T, Bulò SR, Kontschieder P (2017) The mapillary vistas dataset for semantic understanding of street scenes. In: 2017 IEEE International conference on computer vision (ICCV), pp 5000–5009
30. Peng Y, Jin JS, Luo S, Min X, Cui Y (2012) Vehicle type classification using pca with self-clustering. In: 2012 IEEE International conference on multimedia and expo workshops. IEEE, pp 384–389
31. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv:1804.02767
32. Sener E, Sebatli-Saglam A, Cavdur F (2021) Otonom-paylaşımlı araç yönetim sistemi. J Polytechnic
33. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition
34. Tan M, Le QV (2019) Efficientnet: rethinking model scaling for convolutional neural networks. arXiv:1905.11946
35. Tan M, Pang R, Le QV (2019) Efficientdet: scalable and efficient object detection. arXiv:1911.09070
36. Tzutalin (2015) Labelimg. git code. https://github.com/tzutalin/labelImg
37. Udacity self driving car (2018). https://github.com/udacity/self-driving-car
38. Varma G, Subramanian A, Namboodiri AM, Chandraker M, Jawahar CV (2018) IDD: a dataset for exploring problems of autonomous navigation in unconstrained environments. arXiv:1811.10200
39. Wang C-Y, Liao H-YM, Yeh I-H, Wu Y-H, Chen P-Y, Hsieh J-W (2019) Cspnet: a new backbone that can enhance learning capability of CNN. arXiv:1911.11929
40. Wang C-Y, Bochkovskiy A, Liao H-Y (2020) Scaled-yolov4: scaling cross stage partial network. arXiv:2011.08036
41. Wang C-Y, Yeh I-H, Liao H-YM (2021) You only learn one representation: unified network for multiple tasks. arXiv:2105.04206
42. Wang C-Y, Bochkovskiy A, Liao H-YM (2022) Yolov7: trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv:2207.02696
43. Wojek C, Walk S, Schiele B (2009) Multi-cue onboard pedestrian detection. In: 2009 IEEE conference on computer vision and pattern recognition, pp 794–801
44. Yu F, Xian W, Chen Y, Liu F, Liao M, Madhavan V, Darrell T (2018) BDD100K: a diverse driving video database with scalable annotation tooling. arXiv:1805.04687
45. Zhang S, Benenson R, Schiele B (2017) Citypersons: a diverse dataset for pedestrian detection
46. Zhang L, Zhang C, Quan S, Xiao H, Kuang G, Li L (2020) A class imbalance loss for imbalanced object recognition. IEEE J Sel Top Appl Earth Obs Remote Sens 13:2778–2792
47. Zhao Q, Sheng T, Wang Y, Tang Z, Chen Y, Cai L, Ling H (2018) M2det: a single-shot object detector based on multi-level feature pyramid network. arXiv:1811.04533
48. Zhou B, Zhao H, Puig X, Fidler S, Barriuso A, Torralba A (2016) Semantic understanding of scenes through the ADE20k dataset. arXiv:1608.05442