Multimedia Tools and Applications (2022) 81:14247–14282
https://doi.org/10.1007/s11042-022-12447-5

Road object detection: a comparative study of deep learning-based algorithms

Bharat Mahaur[1] · **Navjot Singh**[2] · **K. K. Mishra**[1]

Received: 28 April 2021 / Revised: 18 August 2021 / Accepted: 25 January 2022 /
Published online: 25 February 2022

Abstract

Deep learning field has progressed the vision-based surround perception and has become the most trending area in the field of Intelligent Transportation System (ITS). Many deep learning-based algorithms using two-dimensional images have become an essential tool for autonomous vehicles with object detection, tracking, and segmentation for road target detection, primarily including pedestrians, vehicles, traffic lights, and traffic signs. Autonomous vehicles rely heavily on visual data to classify and generalize target objects which can satisfy pedestrians' and other vehicles' safety requirements in their environment. In real-time, outstanding results are obtained by deep learning-based algorithms for object detection. While several studies have thoroughly examined different types of deep learning-based object detection methods, there are a few comparable studies that either test the detection speed or accuracy of the object detection algorithms. In addition to speed and accuracy, autonomous driving also depends on model size and energy efficiency. However, there is a lack of comparison on various such metrics among existing deep learning-based methods. This article aims to provide a detailed and systematic comparative analysis of five independent mainstream deep learning-based algorithms for road object detection, namely the R-FCN, Mask R-CNN, SSD, RetinaNet, and YOLOv4 on a large-scale Berkeley DeepDrive (BDD100K) dataset. The experimental results are analyzed using the mean Average Precision (mAP) value and inference time. Additionally, various practical metrics, such as model size, computational complexity, and energy efficiency of deep learning-based models are precisely computed. Furthermore, the performance of each algorithm is evaluated under different road environmental conditions at various times of day and night. The comparison presented in this article helps to gain insight into the strengths and limitations of the popular deep learning-based algorithms under practical constraints with their real-time deployment feasibility. Code is publicly available at: https://github.com/bharatmahaur/ComparativeStudy

Keywords Autonomous vehicles · Intelligent transportation system (ITS) · Object detection · Deep learning

✉ Navjot Singh
navjot@iiita.ac.in

Extended author information available on the last page of the article.

1 Introduction

The automobile industries have developed rapidly since the first demonstration in the 1980s [29], the vehicle navigation and intelligence system have improved. The increase in road vehicles raises issues like traffic congestion, road safety, and pollution [47]. The autonomous driving is a very challenging task; a small error in the system can lead to fatal accidents. In order to achieve a high level of safety, visual data plays an important role in enabling advanced driver-assistance systems in autonomous vehicles [47]. The low cost and wide availability of vision-based sensors offer great potential to detect road incidents. Additionally, emerging autonomous vehicles use various sensors and deep learning methods for the detection and classification of objects to improve safety by monitoring the current road environment.

Object detection is a method to localize and classify target objects in an image to acquire a complete understanding of the image. It is currently one of the first fundamental tasks in vision-based autonomous driving. The object detection methods make bounding boxes around the detected objects with the predicted class label and confidence score associated with each bounding box. Figure 1 shows an example of object detection method to identify and locate target objects in an image. The object detection and tracking is a challenging task and plays an essential role in many visual-based applications. At present, the field of Artificial Intelligence (AI) provides several methods for advancing the automation levels by improving the environment perception. The levels of automation were defined by the society of automotive engineers in 2014 [10]. Autonomous vehicles have developed from Level-0 class with no automation to Level-1 with driver assistance automation [10]. The Level-2 class with partial automation enables the vehicle to assist in some automated functionality, but the driver controls many safety critical actions [10]. For Level-3 class with conditional automation, the intelligent vehicle must monitor the complete surroundings in real-time. The driver can only take control over the vehicle when prompted by the system [10]. However, to achieve actual autonomous driving in vehicles, there is a long way to reach the Level-4 class with high automation, and finally, the Level-5 class with full automation [10]. A summary of different automation levels with key features is shown in Table 1.

The sensors used for the detection and monitoring of vehicles may be identified as containing three components, the transducer, the unit for signal processing, and the device for data processing. In an autonomous vehicle, all sensors are particularly important because the vehicle obtains correct information about its surrounding environment. At present, sensors used in the vehicles primarily include the Monocular Camera, Binocular Camera, Light

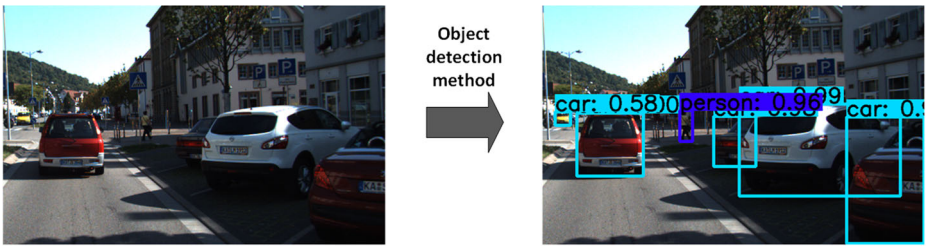

Fig. 1 (left) Objects before detection; (right) Objects after detection showing the bounding box coordinates of the detected object with the predicted confidence score and class label

Table 1 Different levels of driving automation [10]

Levels	Key features
Level 0	No Automation - Fully driver-controlled.
Level 1	Driver Assistance - Some driving assistance features such as cruise control, automatic braking, and lane keeping are included.
Level 2	Partial Automation - Some automated functions like steering assist, automatic parking, collision braking are included. The driver is required to monitor the environment permanently.
Level 3	Conditional Automation - Monitoring the environment at all times is not required. Still, the driver should always be prepared to take charge of the vehicle.
Level 4	High Automation - The vehicle is capable of executing all driving responsibilities under specific situations.
Level 5	Full Automation - A driverless journey in which the vehicle controls all driving responsibilities in every situation at all times.

Detecting and Ranging (LiDAR), Global Navigation Satellite System (GNSS), Global Positioning System (GPS), Radio Detection and Ranging (Radar), Ultrasonic sensor, Odometer, and many more [23]. All these sensors have their benefits and drawbacks. In particular, a LiDAR may not perform accurately in challenging weather conditions. Besides, it is equivalently costly as well. The radar only detects objects in close range. If a vehicle is moving faster, the radar may detect objects at less than their specified range [23]. Furthermore, both binocular cameras and monocular cameras may produce worse detection results in low light. The GPS chip is a power-hungry device that drains the battery rapidly and is costly [18]. The GPS may also have inaccuracy due to environmental interference.

Early autonomous driving systems relied heavily on sensory data for accurate environment perception. Although, localization in object detection has become challenging due to variations in viewpoints, poses, scales, rotation, lighting, and occlusions [55]. The era of computer vision begins with the development of the deep neural network [29, 55]. With the advances in sensing and computational technologies in the field of computer vision, the performance of traditional manual feature-based object detection algorithms has been comparable to that of the deep learning-based object detection algorithms because of continuous growth in large volumes of data and fast development of hardware, particularly Multicore Processors and Graphical Processing Units (GPUs). Furthermore, the deep learning-based algorithms exceed the traditional algorithms in terms of detection speed and accuracy. However, the trade-off for speed and accuracy comes at the cost of energy consumption [30]. As the performance has been significantly improved, research in designing energy efficient models is challenging. The deep learning methods have gained much attention due to the

promising results it has achieved in many applications of computer vision, such as image classification [23, 29, 55], object detection and segmentation [15, 38], deep multi-modal perception [13, 23, 38], moving object detection and tracking [10, 47], and safety modeling [23, 44]. In this article, we focus on deep learning-based approaches for detecting road targets in autonomous driving applications. Moreover, the deep learning detection offers a wide range of applications in the field of Intelligent Transportation System (ITS), including object counting, overtake detection, object classification, lane change detection, emergency vehicle detection, traffic control, traffic sign and traffic light identification, license plate recognition, and many more.

1.1 Related works

In particular, road object detection has been a hot topic for many researchers over the past decade. The work in [9, 12, 22, 45, 53] aim to analyze the deep learning-based algorithms for road target detection without considering up-to-date algorithms. However, many new algorithms have emerged in the deep learning domain, and it is worth summarizing them. The work in [9, 12, 45] use datasets limited in one or more significant aspects, including low fine-grained recognition, poorly annotated images, and imbalance classes. However, experimenting on a diverse large-scale dataset with fine annotated images similar to actual road conditions to test the applicability of deep learning-based algorithms is worth considering. The work in [14, 25, 45, 53] provide subjective analysis, while the work in [9, 12] lack analysis of deep learning-based detection algorithms for road targets in weather environments. However, analysis of algorithms for road object detection with precise metrics under challenging weather at various times of day and night is worth considering. The work in [48, 53] focused on the detection of a single object, namely vehicles, while the work in [9, 14] focused on two objects, namely pedestrians and vehicles. However, traffic signs and traffic lights are equally important for an autonomous vehicle on the road, so it is worth considering them. As discussed in existing works [9, 12, 14, 22, 25, 45, 48, 53], a comparison among different types of deep learning-based methods have focused mainly on detection speed and accuracy with one or two road objects on a limited dataset. Furthermore, it is equally important to consider various practical metrics, such as model size, complexity, and energy efficiency with primary road targets, including vehicles, pedestrians, traffic signs, and traffic lights under dynamic weather conditions on a diverse large-scale dataset. Therefore, there is a lack of detailed analysis and comparison between the different deep learning-based object detection algorithms for road target detection in autonomous driving applications. Although there is some overlap with one of the existing work [53], the overall comparison proposed in this article does not overlap. Still, we present a direct comparison of the proposed article with [53] in Appendix.

1.2 Contributions

Despite the fact that many studies have been proposed for the analysis of deep learning-based methods for road object detection, there is no work that provides a detailed review of detection performances between the different state-of-the-art object detection models considering primary road objects, several practical metrics, and generalization ability under different road environmental conditions, which makes it difficult for researchers to fully understand this field. The primary purpose of this comparative study aims to fill the gap in the literature. In addition, this article presents a more detailed and systematic analysis of deep learning-based object detection methods with key contributions as follows.

1. A comparative review of different aspects of the five popular and trending deep learning algorithms (R-FCN [11], Mask R-CNN [19], SSD [35], RetinaNet [34], and YOLOv4 [3]) from many object detection algorithms with their key contributions on popular benchmarks is presented.
2. The primarily applied deep learning-based algorithms for road object detection are compared on a new large-scale Berkeley DeepDrive (BDD100K) dataset [57]. The dataset contains self-driving images from different geographic, environmental, and weather diversity similar to actual road scenarios.
3. For an autonomous vehicle, four object classes that are most critical, namely vehicles, pedestrians, traffic signs, and traffic lights are considered. The results are analyzed and summarized to show the performance of each model in terms of detection accuracy using mean Average Precision (mAP) value, and inference time and speed on CPU and GPU.
4. Besides detection speed and accuracy, various practical metrics, such as model size, memory footprint, computational complexity, and energy efficiency are precisely measured. In addition, mean sensitivity and specificity are also computed for each object detection model.
5. The generalization ability of each model is evaluated using mAP values under different road weather conditions, including clear, partly cloudy, overcast, foggy, snowy, and rainy at various times of day and night. The parameters are chosen to ensure the credibility of experimental results.
6. Significant guidance for future research and the development of new trends in the field of deep learning-based object detection are included in this work.

The remaining paper is structured as follows. Section 2 presents a brief history of the evolution of the representative techniques in the traditional object detection field, followed by a detailed description of the most pioneering models in the field of deep learning-based object detection. Section 3 includes a description of the dataset, data augmentation methods, experimental environment, and object detection metrics for performance evaluation. In Section 4, the paper elaborates the comparative analysis of numerical results obtained from the experimental evaluation of object detection algorithms and some recommendations for the direction of future research. Section 5 summarizes the development of new trends in the field of deep learning-based object detection. Lastly, Section 6 concludes the paper.

2 Brief overview on object detection methods

2.1 Evolution of object detection

In early research, object detection was based on template matching techniques and geometric representations, focusing on some specific objects or part of an object (before 1990) [36]. Some early researchers framed object detection as a similarity measurement between object components, shapes, and edges (around 1995) [36]. Unfortunately, the results were not as expected, and the failure set the stage for machine learning-based detection methods (late 1990s) [36]. Several extracted features using AI were used for the detection of objects, such as appearance-based statistical models (before 1998), representations of wavelet features (around 2002), and gradient-based feature representations (around 2008) [60]. In traditional methods, image representation was difficult (before 1998); researchers focused on designing sophisticated features with limited resources and skills. With the design of invariant

features, the representations moved from global to local (around 1999) [60]. This continuous progress of local invariant handcrafted features gained popularity, finally led to the development of the first real-time object detector in 2001, known as the Viola-Jones (VJ) detector [36]. Another challenge in traditional methods was the detection of multi-scale objects with varying scales and different aspect ratios. With the advent of the VJ detector, researchers progressed to more intuitive detection methods by building feature pyramids [60] and sliding windows [36]. A combination of which achieved remarkable results in multi-scale object detection (around 2010) [36, 60]. The Convolutional Neural Network (CNN), also called ConvNets proposed by Y. LeCun et al. for object recognition with weight sharing and space displacement can be traced back to the early 1990s, which later became the prototype of Fully Convolutional Network (FCN) in 2010 [36, 60]. As the progress of handcrafted features slowed in the early 2010s, the Deep CNN grabbed the attention of many researchers in 2012 [36, 60]. In 2014, the deep learning-based techniques achieved a significant amount of progress, resulting in the development of the first deep learning-based object detector, known as Region-based CNN features (R-CNN) algorithm [17].

In general, the milestone in the object detection field has been through two periods: Traditional feature-based object detection algorithms (before 2014) and Deep learning-based object detection algorithms (since 2014) [60]. Figure 2 shows significant development in the milestones of object detection algorithms over the years.

2.2 Traditional feature-based object detection algorithms

Object detection using computer vision has been a challenging task in the literature for over two decades. Traditional feature-based methods were based on handcrafted descriptors and discriminative classifiers to obtain embedding for region proposals [36]. The traditional object detection algorithms include three primary modules; the first is informative region selection, the second is feature extraction, and the last is classification. In informative region selection, the aim is to search specific regions for relevant objects in an image using a multi-scale sliding window [60]. After regions are identified, several features are extracted to identify different objects in an image. The most common traditional-based feature extractors in the literature for object detection were SIFT [60], SURF [60], HOG [43], and Haar [32]. These feature extractors were robust to scale, illumination, and rotational variance. In the last step, the extracted feature vectors are passed

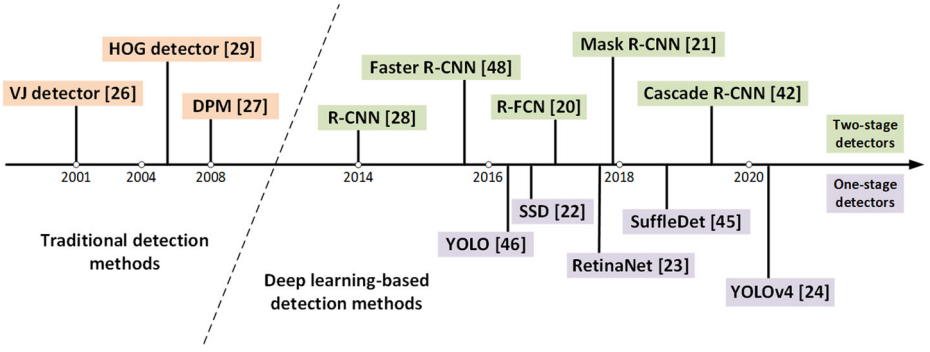

Fig. 2 Milestones in the field of object detection

through a classifier to identify the target class object. The most popular classifiers for traditional object detection were SVM [8], Adaboost [5], bagging [5] and DPM [60]. HOG [43] features are extracted in two steps: evaluating edge operators over the image, followed by discretizing and discarding the orientations of edge intensities into the histogram. HOG [43] features show good detection performance in different computer vision tasks, including vehicles and pedestrians, but are usually slow to compute. Haar [32] features are calculated by sums and differences of rectangles over an image patch. It was best suited for real-time detection as it was highly efficient in computing the symmetric structure of detecting objects. DPM [60] is an extension of HOG [43], based on a divide and conquer strategy consisting of a root filter and multiple part filters. The HOG [43] feature vector used in combination with the SVM [8] classifier has been commonly used to detect the object orientation. The HOG-SVM [52] has achieved great success in object detection. A combination of Haar [32] and HOG [43] was used for object detection and tracking. The Haar feature vector and the Adaboost [52] were popular in the computer vision field for the detection of various features in applications, including face, person, vehicle, and many more. The DPM [60] improved detection speed by over ten times compared to other traditional object detectors without compromising accuracy. Although the current object detectors have outperformed DPM [60], many of them are still profoundly influenced by its insights.

2.3 Deep learning-based object detection algorithms

Over the past decade, the field of deep learning has attracted much attention from many researchers, and several deep learning-based algorithms have emerged in recent years. Many objects under different road conditions in dynamic environments become difficult to identify, so to perform detection on these images captured by the camera is always challenging. The traditional detection methods require manual extraction of features by experts over time. But the deep learning methods require huge volumes of data to learn feature characteristics over time. The accuracy of the deep learning-based model depends heavily on its feature extraction network, commonly known as the backbone network [55]. The primary role of a backbone network is to extract features from the input image used by the model [58]. These extracted features are used to classify the object class. A backbone network is characterized by the number of layers and parameters, which also impacts its performance. Table 2 summarizes various backbone networks commonly used in deep learning-based object detection in terms of the number of layers, year, parameters, and top-1 ImageNet accuracy [58]. The most common metrics to

Table 2 Comparison of backbone networks in deep learning

Model	Layers	Year	Parameters (M)	Accuracy (%)
AlexNet [28]	7	2012	62.36	63.32
VGG [46]	16	2014	138.37	73.01
GoogLeNet [50]	22	2014	6.64	69.79
ResNet [20]	50	2015	25.56	76.13
ResNeXt [56]	50	2016	25.03	77.85
DarkNet [41]	53	2018	61.57	77.23

measure the performance of the model are precision, recall, and inference time. The average detection precision is computed under different recall values. However, the mean of average precision over all classes is calculated for object detection, also known as mean Average Precision (mAP) [55]. The object detection metrics will be discussed in detail in Section 3.4.

In the field of computer vision, the visual recognition done by the CNN feature extraction (backbone networks) is close to that of human visual mechanism, representing a complete process from the edge to the part of an object. Because of the continuous rise in data and fast hardware advancement, particularly the GPUs (mainly used for parallel processing), the deep learning-based algorithms for object detection have shown excellent performance compared to the traditional detection algorithms which gave recognition to deep learning approaches to the industries worldwide. With this evolution, many techniques emerged in deep learning-based object detection, including object proposals [36], feature and bounding box (BB) regression [36], multi-reference and multi-resolution detection [60], hard negative mining [60], network pruning, and quantification [36, 60], etc. Focusing on real-time performance in terms of accuracy and speed in the academic field, the deep learning-based algorithms for object detection are developed in two directions; two-stage object detection algorithms and one-stage object detection algorithms [2] emphasizing the accuracy and detection speed, respectively.

Deep learning has progressed many state-of-the-art techniques for various perception tasks in autonomous driving, including object detection, tracking and segmentation. Multiple sensing modalities (LiDARs, Radars, and Cameras) are fused to exploit a more accurate scene understanding for deep multi-modal object detection and segmentation [13]. Many machine learning-based techniques can be applied to road safety modeling for real-time crash prediction by frequency and severity [44]. However, this article focuses on deep learning-based approaches for detecting primary road targets applied to autonomous driving.

Several algorithms have emerged in recent years that may consider the object specificity on the road, such as dynamic road environments, background lighting conditions, position, dimension, color, shape, and size. Many existing algorithms, including but not limited to Cascade R-CNN [7], RefineDet [59], SqueezeDet [54], and ShuffleDet [37], result from a combination of various features in an attempt to achieve better results than their compared counterparts. For instance, Cascade R-CNN [7] is based on multi-stage R-CNN [17] to improve localization accuracy, while RefineDet [59] improves accuracy based on two-stage regression, similar to R-FCN [11]. On the other hand, SqueezeDet [54] and ShuffleDet [37] are lightweight detectors mainly targeted for mobile platforms due to their small size and efficient computing resources. SqueezeDet [54] is a single shot model, similar to SSD [35] where region proposal and localization of multiple objects is based on the modified detection pipeline from YOLO family [39–41]. ShuffleDet [37] is another modified variant of SSD [35] where regression is based on grouped convolutions, similar to Mask R-CNN [19]. While these lightweight models aim at high efficiency, they usually lack detection accuracy [55, 58]. Autonomous vehicles have high requirements for the real-time detection of primary road targets. Therefore, this article focuses on five independent state-of-the-art deep learning-based object detection algorithms, namely the R-FCN [11], Mask R-CNN [19], SSD [35], RetinaNet [34], and YOLOv4 [3] from many deep learning-based algorithms based on significant improvements proposed by the model over the years to provide a comprehensive and comparative research on the analysis of speed, accuracy, complexity, and efficiency in performance results.

2.3.1 Two-stage object detection models

The two-stage detectors are also referred to as Region-based methods [60]. In this type of object detection model, the processing of an image is carried in two steps. In the first step, a series of sparse candidate frames are generated, i.e., region proposals are extracted from a scene, and in the second step, the candidate frames are verified, followed by classification and regression to improve the scores and locations. The main advantage is the high accuracy and localization of object recognition. The two-stage detector models are slower than one-stage detector models and require more complex training. In this article, R-FCN [11] and Mask R-CNN [19] are used as the representative algorithms of this two-stage object detection models.

R-FCN The Region-based Fully Convolutional Networks (R-FCN) [11] algorithm was proposed by J. Dai et al. in 2016. It is based on the modified structure of the Faster R-CNN [42] to maximize the sharing of convolution parameters to address the contradiction between position sensitivity and translation invariance and increase the speed [11]. Translation variance in detecting objects and translation invariance in classifying images may cause high inconsistencies. The R-FCN addresses this problem partially under the same detection precision, but still, the detection speed is much inferior compared to the one-stage algorithms. The architecture of R-FCN is shown in Fig. 3. The R-FCN uses the convolution layers to produce the position-sensitive score maps, for instance of size k×k×(C+1), where (C+1) represents the total number of object classes with one additional background class, and each object class having k×k score maps. Also, the R-FCN generates Region of Interest (RoI) proposals using the fully convolutional RPN (position-sensitive RoI pooling layer). The different RoI regions can correspond to different positions of the score map. Each RoI region is divided into k×k subregions, each of which corresponds to one area of the score map [11]. This process is repeated for evaluating the score of each class. If the response value of the class match values for k×k subregions of each class, then an average of subregions values (average voting) is used to obtain the single object match score for each class [11]. Finally, the RoI is classified with a softmax function for the remaining C+1 object class to get the probability of belonging to each class. A class-agnostic bounding box regression, named as regression score map [11] is obtained by appending another 4×k×k convolution layer.

Experiments show that using ResNet-101 [20] as a backbone in R-FCN achieved the mAP of 31.5 on the MS COCO dataset [11]. The R-FCN has the same detection precision

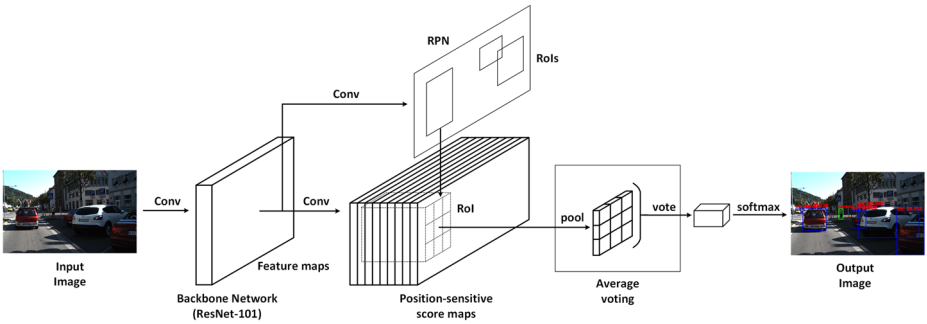

Fig. 3 Architecture of the R-FCN [11]

but is much faster than Faster R-CNN [11]. In this work, the R-FCN is compared with the other object detection algorithms, and it is found that R-FCN outperforms SSD [35] in terms of detection accuracy.

Mask R-CNN The Mask Region-based Convolutional Neural Networks (Mask R-CNN) [19] algorithm was proposed by K. He et al. in 2017. It is an extending work of Faster R-CNN [42] primarily for instance segmentation task. The Faster R-CNN [42] proposed in 2015 replaces the traditional selective search with Feature Pyramid Network (FPN) [33] to perform detection but fails to address the problem of translation invariance and position sensitivity. Instance segmentation can be classified into two parts: first is object detection, and second is semantic (pixel-level) segmentation. Mask R-CNN extracts feature proposals from scrutinized images using Faster R-CNN [42] method and then performs segmentation by object detection with simultaneous generation of bounding boxes and high-quality masks of highest Intersection over Union (IoU) prediction with an increase in the computation speed. Mask R-CNN has two significant contributions. Firstly, it uses a more accurate RoI align module, called RoIAlign [19] to replace the traditional RoI pooling module because pixel-level segmentation needs much more fine-grained alignment. The traditional RoI pooling method has poor location misalignment due to quantization. Secondly, an extra branch from the RoIAlign module is inserted. This additional branch extracts a small feature map or bin from each RoI, which accepts the output of the RoIAlign layer to be fed into the two convolution layers. At the last step, the final output obtained from these layers is the mask (BB) itself, as shown in Fig. 4. The RoIAlign module initially calculates the floating-number of the coordinates of each RoI bin, and then a bilinear interpolation operation is performed to find the exact feature values at regularly sampled locations in each feature map or bin to solve the problem of quantization [19]. The results are then aggregated using a max or average pooling technique to find values of each feature map or bin to produce a more precise pixel-to-pixel alignment.

Experiments show that using FPN [33] with ResNeXt-101 [56] as a backbone in Mask R-CNN achieved the mAP of 39.8 on the MS COCO dataset [19]. The Mask R-CNN is the recent release in the family of two-stage object detection models and overcomes many limitations of previous R-CNN algorithms. In this work, the Mask R-CNN is compared with the other object detection algorithms, and it is found that Mask R-CNN outperforms RetinaNet [34], R-FCN [11], and SSD [35] in terms of detection accuracy.

2.3.2 One-stage object detection algorithms

In one-stage detection algorithms, there is no region proposal step or intermediate region process for detection, and the prediction results are acquired directly from the image. In

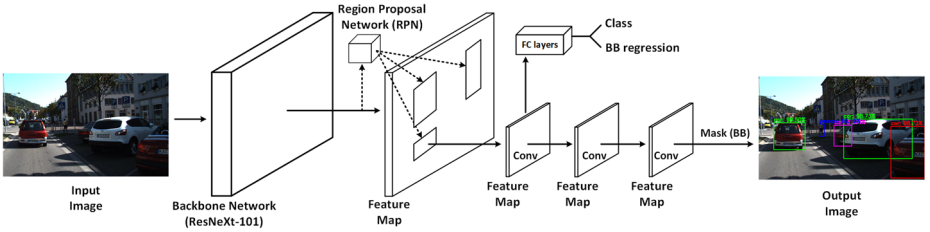

Fig. 4 Architecture of the Mask R-CNN [19]

this type of model, the input image is sampled at different positions uniformly using variable scales and aspect ratios, followed by sampling a CNN layer for accurate extraction of features to perform classification and regression. The main advantage is the fast detection speed. The one-stage detection models are time efficient, easier to optimize, and more suitable for real-time devices. In this work, SSD [35], RetinaNet [34], and YOLOv4 [3] are used as the representative algorithms of this one-stage object detection models.

SSD The Single Shot MultiBox Detector (SSD) [35] model was proposed by W. Liu et al. in 2015. In earlier approaches by R-CNN [46] and R-FCN [17], Region Pyramid Network (RPN) [34] was used to generate regions of interest, and then classification was performed on these regions using fully connected or convolution layers. The disadvantages of R-CNN [46] algorithm is that the small-scale objects detection is difficult, and their positioning is incorrect. Therefore, SSD can overcome these limitations to some extent. It performs both the operations in only a single shot. It has two significant contributions. Firstly, feature maps of different sizes ($10\times10, 5\times5, 3\times3$, etc.) are extracted from the scene where small-scale feature maps are generally used to detect large objects and large-scale feature maps are generally used to detect small objects (multi-reference detection). Secondly, in reference to each feature map anchor boxes of different scales and aspect ratios are generated (multi-resolution detection). VGG-16 [46] is used as a feature extraction network in SSD. On top of the VGG-16 [46] network, six more convolution layers are added for detection, as shown in Fig. 5. If an image and a set of truth or object labels are given as an input to the SSD, it produces a sequence of feature maps of different scales, followed by a 3×3 convolution filter on each feature map to produce default bounding boxes. As the image processes, bounding box offset and the class probabilities are predicted simultaneously for each box. SSD runs detection only at the top layer to produce the best predicted bounding box and label.

Experiments show that using VGG-16 [46] as a basic feature extraction network, the SSD512 (input image size: 512x512) algorithm achieved mAP of 28.8 on the MS COCO dataset [35]. In this work, the detection precision of SSD is found to be comparable to that of two-stage detector R-FCN [11]. The SSD shows fast detection speed and high energy efficiency when compared to the other one-stage and two-stage object detection algorithms.

RetinaNet The one-stage detectors are known for their simplicity and high speed, but the accuracy of one-stage detection models is always lower than two-stage detection models. After identifying possible reasons behind it, T. Y. Lin et al. proposed RetinaNet [34] in 2017. The one-stage models have a higher magnitude compared to two-stage models because of the lack of region proposals. Higher magnitude means that one-stage detectors are more adaptable towards real-time challenges due to their high speed and low computing requirements [36]. During training, the initial stage filters out a large number of negative locations and proposes a small number of candidate locations. This is mainly because of the extreme foreground-background class imbalance encountered during the training of dense detectors

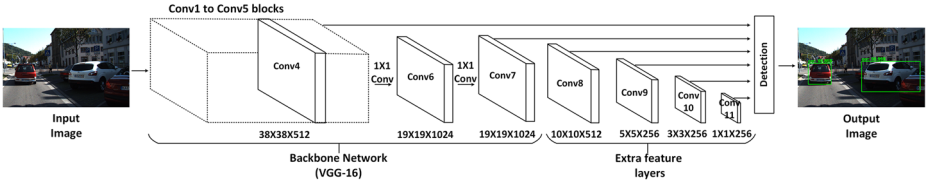

Fig. 5 Architecture of the SSD [35]

[34]. To solve this problem, the authors propose a new loss function, called Focal Loss [34] to replace the binary cross-entropy from previous works [11, 35, 46]. The loss of simple samples covers the loss of difficult cases by adding an estimated value of weight to each sample. After adding the value, the predicted weight is the probability of the network for classifying the sample as simple or difficult. For an effective classification, the weights of simple samples must be reduced relatively by increasing the weights of hard samples. The focal loss function focuses more on misclassified hard training examples and avoids a large number of easy examples by balancing these weights, thereby increasing the accuracy of the model so that the one-stage detector models can perform better compared to two-stage models while maintaining high detection speed. The RetinaNet uses ResNet [20] as a backbone along with FPN [33] as a feature extraction network. The RetinaNet model has two stages, as shown in Fig. 6. In the first stage, a sparse set of region proposals are generated using FPN, and in the seconds stage, each candidate location is classified.

Experiments show that using the Focal Loss as a classification function and ResNet-101 [20] with FPN [33] as a backbone; the RetinaNet model achieved mAP of 39.1 on the MS COCO dataset [34]. In this work, the RetinaNet is compared with the other object detection algorithms, and it is found that RetinaNet outperforms R-FCN [11] and SSD [35] in terms of detection accuracy.

YOLOv4 The You Only Look Once (YOLO) algorithms [39–41] have been the most balanced object detectors in speed and accuracy. Various advancements have been suggested to overcome the shortcomings of its previous three generations, including poor detection accuracy [39, 40], unsatisfactory detection of small objects [39, 40], low recall [39–41], high localization error [39, 40], unusual aspect ratios [39, 41], difficulty in predicting multiple object in range [39, 40], and strong constraints on BB predictions [39–41]. A. Bochkovskiy et al. proposed YOLOv4 [3] in 2020 with significant changes from the previous versions. It is the most recent and advanced release of the YOLO family. YOLOv3 [41] uses Darknet-53 [41] and CSPNet [41] as the main network. The Darknet-53 [41] has the same accuracy as that of ResNet [20] but is faster. The dense block comprises of multiple convolutional layers with batch normalization, Rectified Linear Unit (ReLU). A dense net is formed consisting of multiple dense blocks with transition layers followed by convolution. Cross-Stage Partial (CSP) connections divides the input feature maps of dense blocks into two parts. One part goes as an input to the next transition layer, and the other part goes through the

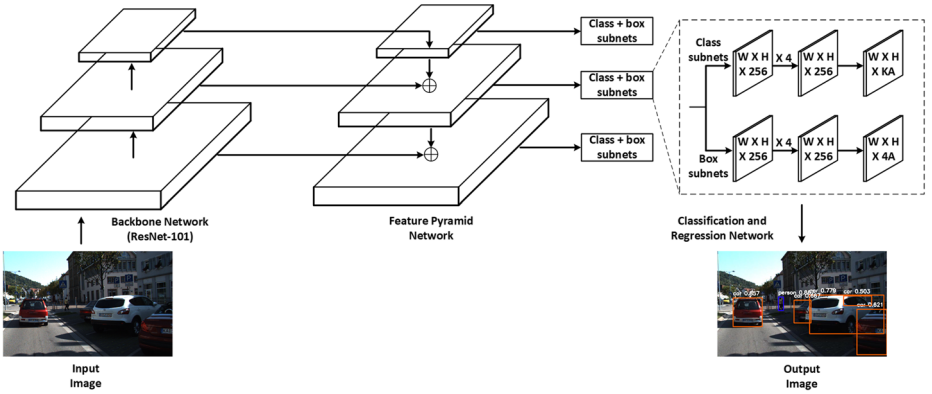

Fig. 6 Architecture of the RetinaNet [34]

dense block. Based on this, YOLOv4 proposes a new feature extractor, CSPDarknet-53 [3]. This extractor enhances the CNN network learning ability and improves the accuracy while reducing the amount of calculation and memory cost. YOLOv4 uses Spatial Pyramid Pooling (SPP) module along with Path Aggregation Network (PANet) is used as the neck over CSPDarknet-53 because it increases the receptive field, differentiates the most significant features, and does not causes any speed reduction. The original YOLOv3 [41] network is used as the head. A simple architecture of YOLOv4 is shown in Fig. 7. Few other features in YOLOv4, such as the selection of optimal hyper-parameters while applying Genetic Algorithm (GA), and Cross mini-Batch Normalization (CmBN) apart from the architecture, enables the detector to be more suitable for training on a single GPU. YOLOv4 proposes two new components classified as Bag of Freebies (BoF) [3] and Bag of Specials (BoS) [3]. The BoF consists of multiple training strategies to get better accuracy without increase in the hardware cost. BoF uses techniques, such as Complete-IoU loss, DropBlock regularization, Class label smoothing, Mosaic and CutMix data augmentation, Self-Adversarial Training (SAT), Cosine annealing scheduler, and many others to increase the flexibility of input images [3]. This increases the robustness of the object detection model for the images obtained from dynamic environments. BoS consists of methods, such as Mish and Leaky-ReLU activations, Distance-IoU post-processing, PAN path-aggregation block, SPP-block, SAM-block, and many others that may slightly increase the inference cost, but provide significant improvements in speed and accuracy for real-time object detection. YOLOv4 includes many innovative ideas compared to the existing models, thereby showing a prominent advantage in real-time performance.

Based on the above-mentioned improvements, the YOLOv4 algorithm achieved mAP of 43.5 on the MS COCO dataset [3]. In this article it is found that YOLOv4 is currently the best choice for road object detection with exceptionally good detection accuracy, fast speed and low energy consumption compared to the previous one-stage and two-stage detectors.

Recently, a new object detection algorithm based on YOLOv3 [41] was released in 2020, known as YOLOv5 [27]. The authors claim to provide an excellent balance of detection speed and accuracy. However, for the detection of complex road targets, it is difficult for YOLOv5 [27] to accurately detect small objects, such as traffic signs and traffic lights. Although the innovations in the architecture of YOLOv4 and YOLOv5 [27] are very similar, we provide a detailed comparison of YOLOv5 [27] with YOLOv4 in Section 4.3.

In summary, the deep learning methods can be classified into two types of object detectors: two-stage detectors and one-stage detectors; the two-stage object detection can be considered as a coarse-to-fine process, while the one-stage object detection can be considered as a complete-in-one-step procedure [60]. Earlier studies [9, 12, 14, 22, 53] show

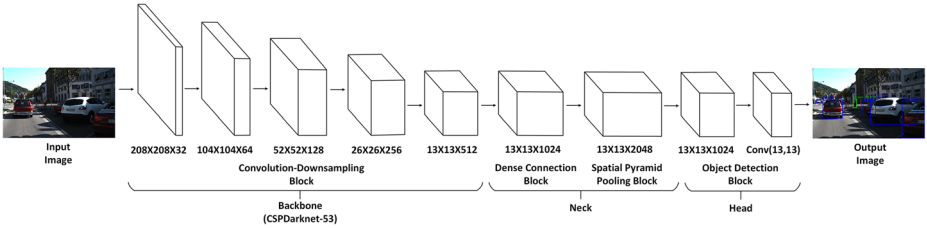

Fig. 7 Architecture of the YOLOv4 [3]

analysis of deep learning-based object detection models ignoring several other essential aspects in comparison. Table 3 lists the observed strengths and limitations of two-stage and one-stage detection models along with the significant innovations the model proposes in the field of deep learning-based object detection.

3 Experimental evaluation

3.1 Dataset description

The selection of dataset is considered very necessary for exploiting various tasks in real-time autonomous driving. Table 4 shows a comparison of a few different datasets available for autonomous driving applications, primarily for road object detection. The annotations

Table 3 Innovations with strengths and limitations of various two-stage and one-stage deep learning models

Model	Innovations		Strengths	Limitations
Two-stage	R-FCN [11]	– Proposes position-sensitive score maps for accurate RoI pooling – Performs class-agnostic bounding box regression for effective localization	– Fast computation speed – Low computational RoI operations	– Complicated training
	Mask R-CNN [19]	– Performs instance segmentation for object detection – Uses RoIAlign layer for pixel-to-pixel alignment	– Provides semantic segmentation – Accurate RoI frame regression	– High number of false detections
One-stage	SSD [35]	– Uses multi-scale feature map for small object detection – Adopts anchor boxes of different scales and aspect ratios for accurate positioning	– Simple model – Low computing requirements	– Low accuracy in dynamic environments
	RetinaNet [34]	– Uses multi-scale ResNet with FPN as a feature extraction tool – Proposes a focal loss function to adaptively balance the weights of different samples	– Improved detection precision on small objects – Stable training for class imbalance	– Requirements for real-time detection
	YOLOv4 [3]	– Proposes a new feature extractor CSPDarknet-53 – Introduces BoF and BoS methods for detectors and backbone network	– Fast detection accuracy – Training optimized for a single GPU	– Small target detection accuracy could be better

Table 4 Comparison of autonomous driving datasets for road object detection

Dataset	Year	Images (K)	Classes	Annotations (M)	Weather diversity
KITTI [16]	2012	15	8	0.2	Low
BDD100K [57]	2018	100	10	1.8	High
ApolloScape [24]	2018	144	27	2.1	Low
EuroCity [4]	2019	47	30	0.8	Medium
nuScenes [6]	2019	1386	23	224	Medium

include only four classes (as per availability), namely vehicle, pedestrian, traffic sign, and traffic light. In comparison to other datasets, the BDD100K dataset [57] is selected for experimental evaluation containing 100K images (70K training images, 20K testing images, and 10K validation images) from different geographic, environmental, and weather scenarios, such as streets, tunnel, gas station, parking lot, residential, and highways under diverse conditions at various times of day and night in real traffic urban streets; with up to 90 objects per image. Each object has boolean values for occlusion and truncation. This article selects BDD100K dataset [57] due to its huge challenging number of self-driving images under complex road scenarios and diverse weather conditions. Additionally, the dataset provides real-world driving images to test the deep learning-based algorithms under constraints performing tasks of various complexities. A snapshot of multiple images from BDD100K dataset [57] is shown in Fig. 8.

The BDD100K dataset [57] originally contains objects, namely person, rider, bike, car, bus, truck, motor, train, traffic light, and traffic sign. In this evaluation, the irrelevant samples are eliminated because of class imbalance and specifically four object classes that are considered most critical for an autonomous vehicle, namely vehicle (bike, car, bus, truck, and motor), pedestrian (person and rider), traffic sign, and traffic light are retained. It is

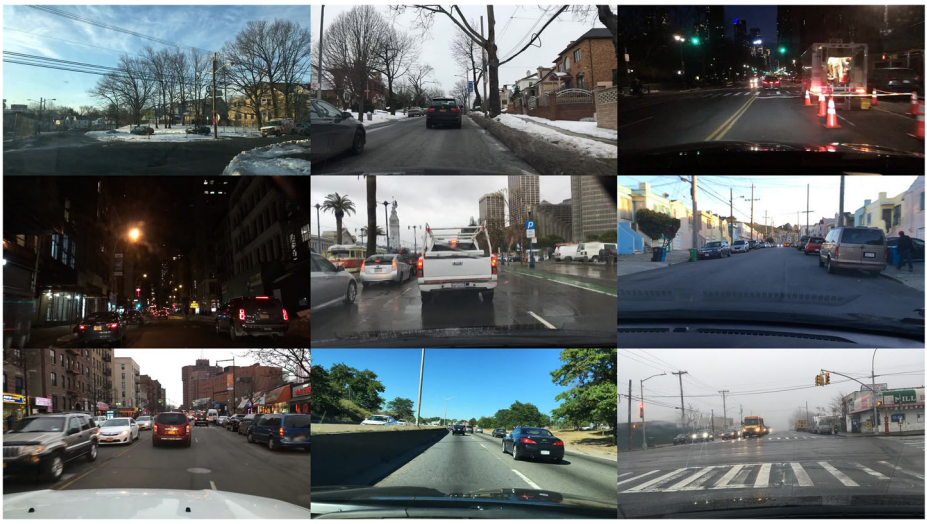

Fig. 8 A snapshot of multiple images from BDD100K dataset

worth mentioning that a few images in the dataset are found invalid (incorrect label or coordinates). After resolving the number of annotated objects that are considered in this article are listed in Table 5.

3.2 Dataset augmentation

Data augmentation is a set of different techniques used to enhance the size and quality of training models. The objective is to increase the generalization ability without changing the category of images. This article performs few basic augmentation operations on the BDD100K dataset [57], including image displacement, horizontal flipping, random scaling, random cropping, motion blurring, and random noise adding, as shown in Fig. 9. In addition to basic augmentation methods, this article uses Mosaic data augmentation [27]. The Mosaic is a new data augmentation technique proposed by Glenn Jocher in 2020 that uses four training images instead of a single image [27]. This has two main advantages. Firstly, the model learns to recognize objects outside of the normal contexts found in the dataset. Secondly, it improves the batch normalization statistics using small mini-batches, which makes the training possible on a single GPU. The implementation of Mosaic data augmentation is shown in Fig. 10.

3.3 Experimental setup

The popular deep learning frameworks, Tensorflow [1] and Caffe [26] are used to train the models as per official documentation. The experimental environment consists of CUDA v10.1, cuDNN v7.6.4, OpenCV v4.5.0, Ubuntu 18.04.5 OS, 16 GB RAM, Intel i7-8750H CPU, and Nvidia GeForce RTX 2080 Ti GPU. The augmentation is applied to training and validation images to increase the learning ability. Therefore, the optimal batch sizes of 2 (for Mask-RCNN), 4 (for R-FCN and RetinaNet), and 8 (for SSD and YOLOv4) are used to speed up this evaluation. The Stochastic Gradient Descent (SGD) [49] is used as an optimizer during training. The initial value of the learning rate is set to 0.1, the momentum rate is set to 0.949, and the decay rate is set to 0.0001 to reduce training loss and prevent gradient explosion. The learning rate is reduced by 0.5 at every 50th epoch. No other parameters were modified and taken as default by the frameworks. The training lasted for 200 epochs. After training, the network weights are obtained to test the experimental results on images from BDD100K test dataset.

3.4 Object detection metrices

For evaluation of detection performance, the mean Average Precision (mAP) is computed. mAP is currently the most popular object detection measure for evaluating detection methods, as described by PASCAL VOC Challenge [36, 60]. The basic parameters used are True positive (TP), False positive (FP), False negative (FN), and True Negative (TN). TP is the

Table 5 Number of annotated objects in BDD100K dataset

Set	Vehicles	Pedestrian	Traffic signs	Traffic lights
Training	765,066	95,866	239,686	186,117
Testing	218,587	27,524	68,984	53,291
Validation	109,807	13,911	34,908	26,885

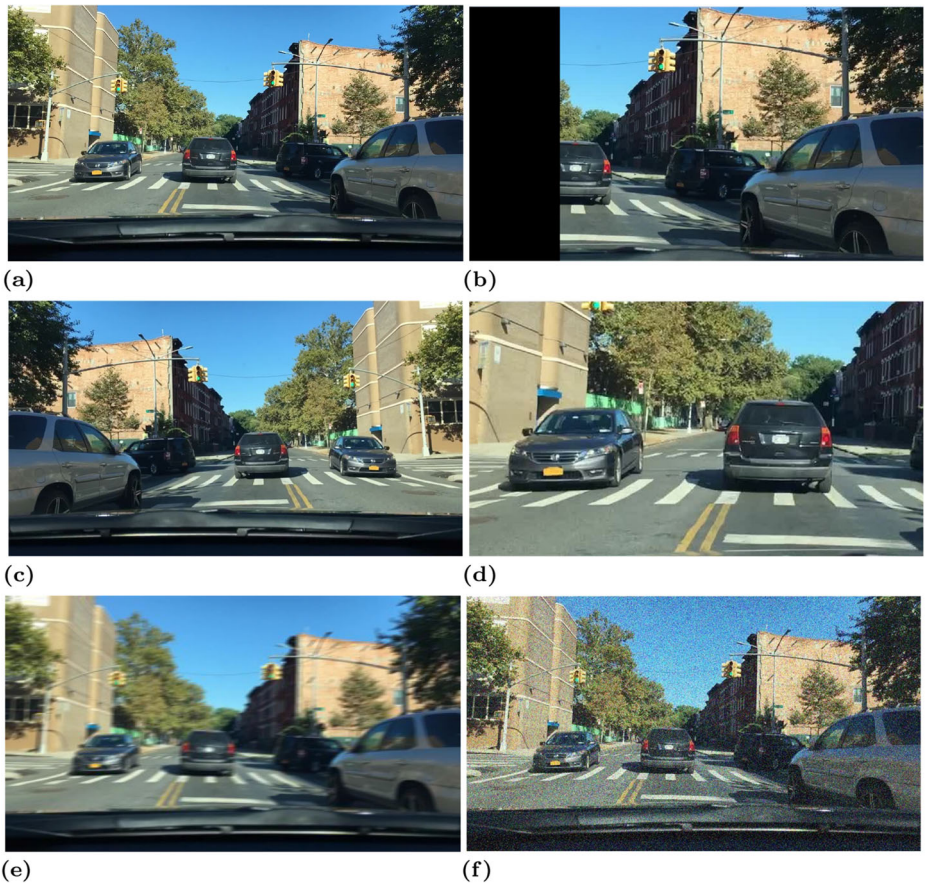

Fig. 9 Data augmentation methods (**a**) original image (**b**) displacement (**c**) horizontal flipping (**d**) random cropping (**e**) motion blurring (**f**) random noise adding

number of detections in both ground truth and results. FP is the number of detections in results only, but not in the ground truth. FN is the number of detections in ground truth only, but not in results. TN is the number of correct negative detections in both ground truth and results. Precision (P) is the percentage of correct positive predictions, defined in (1). Recall (R) is the percentage of correct positive predictions among all ground truths, defined in (2).

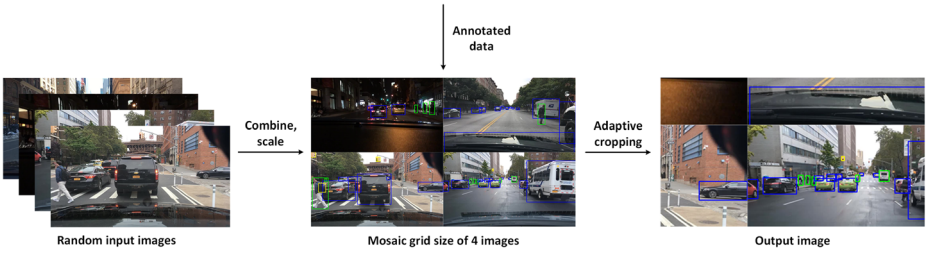

Fig. 10 Implementation of the Mosaic data augmentation

This article uses four classes, therefore multi-class sensitivity is defined in (3). Similarly, multi-class specificity is defined in (4). The P and R cannot comprehensively determine the performance of a model, so Average Precision (AP) is obtained individually for each class. The AP is considered to be a more accurate metric than F1-score because it displays the relationship between P and R globally; therefore, in this article, F1-score is not calculated. Initially, a Precision-Recall (PR) curve is computed on the basis of prediction results against all the ground truth values. A prediction is considered as TP if it satisfies the following two criteria: (1) the bounding box it represents has an IoU $> 50\%$ (default threshold [36]) relative to its corresponding ground truth bounding box, and (2) the predicted class label must be same as the ground truth. The curve is then updated by monotonically decreasing the precision. This is obtained by considering the precision for recall R to the maximum precision obtained for whose recall value is greater than or equal to R. Finally, AP is the area under the PR curve calculated under the default threshold, defined in (5). The mAP is simply the average of AP over all classes, defined in (6).

$$Precision\,(P) = \frac{TP}{TP + FP} = \frac{TP}{all\,detections} \tag{1}$$

$$Recall\,(R) = \frac{TP}{TP + FN} = \frac{TP}{all\,ground\,truths} \tag{2}$$

$$mean\,Sensitivity = \frac{1}{4}\sum_{i=1}^{4}\frac{TP_i}{TP_i + FN_i} \tag{3}$$

$$mean\,Specificity = \frac{1}{4}\sum_{i=1}^{4}\frac{TN_i}{TN_i + FP_i} \tag{4}$$

$$Average\,Precision\,(AP) = \sum_{n}(R_{n+1} - R_n)\max_{R:\tilde{R}\geq R_{n+1}} P(\tilde{R}) \tag{5}$$

where $P(\tilde{R})$ is the measured precision at recall $\tilde{R}$

$$mean\,Average\,Precision\,(mAP) = \frac{1}{4}\sum_{i=1}^{4} AP_i \tag{6}$$

where AP_i is the AP at class i

For validating the computational efficiency of the model, Floating Point Operation per second (FLOPs) is computed by dividing the GPU computation time from the total FLOPs [30, 54]. The size of the model is computed based on the total number of parameters with intermediate activation maps. For a fair comparison, FLOPs and memory footprint (or primary memory usage) are calculated for a batch size of one. In general, GPU power usage is measured with Nvidia's system monitor interface (nvidia-smi). The power value is sampled with a fixed interval of 0.1 seconds and compute average GPU power [30]. The energy consumption per image is measured as the ratio of average power over inference speed [30], defined in (7). Energy efficiency is defined as the total amount of energy consumed to process an input image; a lower value of energy consumption means more energy efficiency.

$$\frac{Average\,power\,(Joule/Second)}{Inference\,speed\,(Frame/Second)} \tag{7}$$

4 Results and discussion

4.1 Results

From earlier studies [38, 55], it is found that occluded and truncated objects can significantly impact the performance of an object detection model. Therefore, in this comparative study, the test dataset is divided into two category levels, i.e., objects without occlusion and truncation, and objects with occlusion and truncation. Figure 11 shows the PR curves obtained during model training of each object detection algorithm for each class without occlusion and truncation. Figure 12 shows the PR curves obtained during model training of each object detection algorithm for each class with occlusion and truncation. Table 6 lists the detection performance in terms of AP of each class and mAP of object detection models. Table 7 lists the performance of object detection algorithms with inference time per

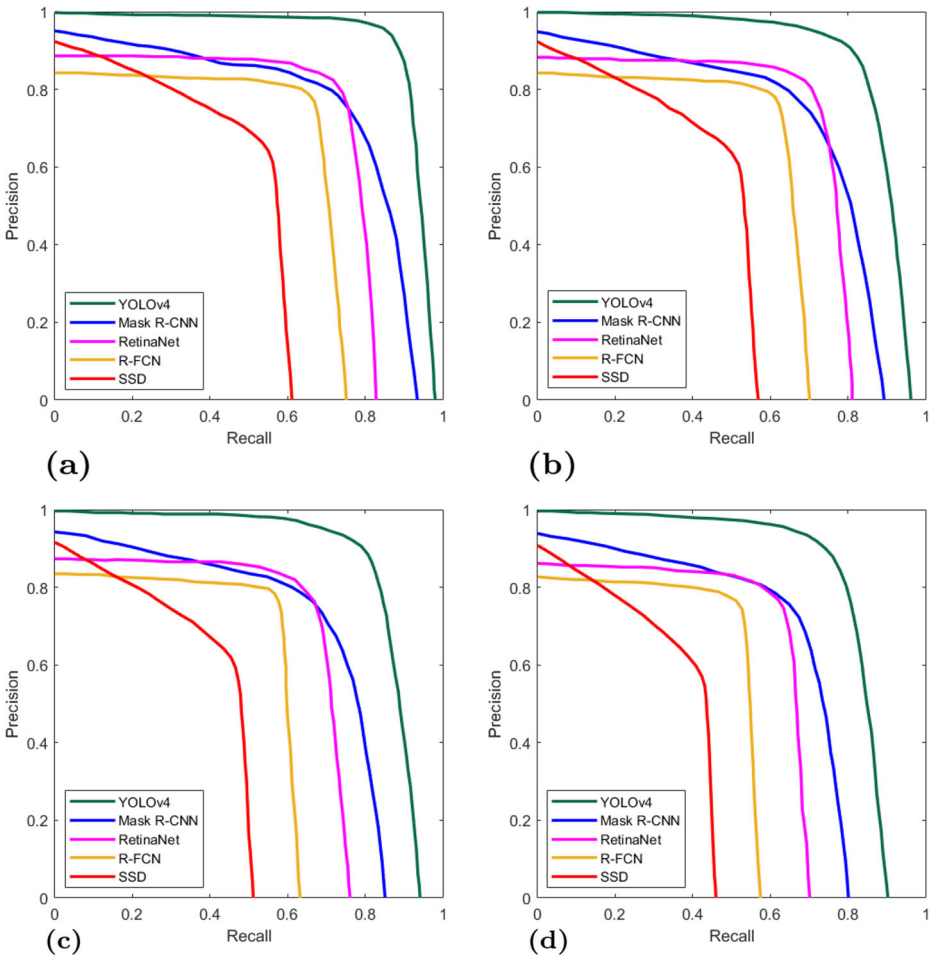

Fig. 11 PR curves of object detection models without occlusion and truncation on BDD100K dataset (**a**) Vehicle (**b**) Pedestrian (**c**) Traffic sign (**b**) Traffic light

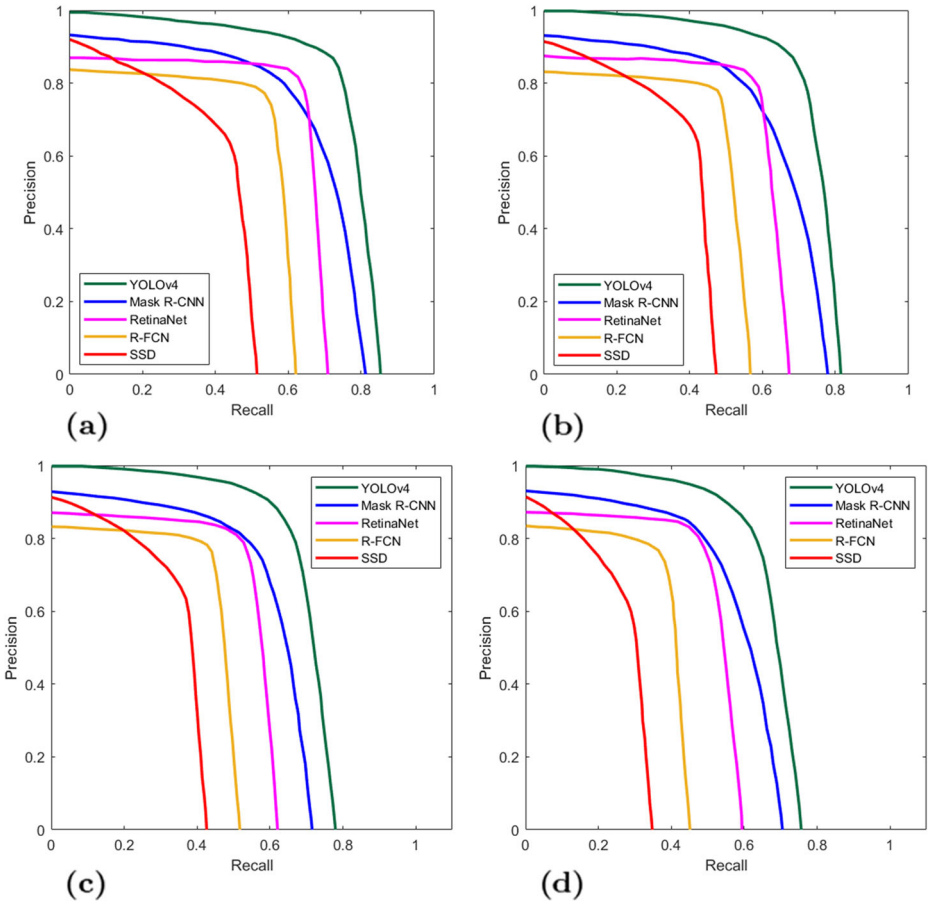

Fig. 12 PR curves of object detection models with occlusion and truncation on BDD100K dataset (**a**) Vehicle (**b**) Pedestrian (**c**) Traffic sign (**b**) Traffic light

image, and speed in frame per second on CPU and GPU. Table 8 lists the sensitivity and specificity per class and mean sensitivity and specificity of object detection models. Table 9 lists the comparison in terms of backbone architecture used in this experimentation, size of the model, memory footprint, FLOPs, and energy efficiency. All results are obtained on the images from BDD100K test dataset. The best values in Tables 6, 7, and 8 are represented in bold.

From Figs. 11 and 12 following inferences can be drawn:

- The one-stage detection model YOLOv4 achieves the highest detection precision and recall for target detection in all levels.
- The two-stage detection model Mask R-CNN shows better precision and recall than RetinaNet, R-FCN, and SSD for target detection in all levels.
- The R-FCN shows better precision compared to SSD for target detection in all levels.
- The recall of Mask R-CNN is close to YOLOv4 for target detection with occlusion and truncation.

Table 6 Detection accuracy of object detection models on BDD100K dataset

Model	Without occlusion and truncation (WOT)					With occlusion and truncation (WIOT)				
	$AP_{vehicle}$ (%)	$AP_{pedestrian}$ (%)	$AP_{Traffic\ sign}$ (%)	$AP_{Traffic\ light}$ (%)	mAP_{WOT} (%)	$AP_{vehicle}$ (%)	$AP_{pedestrian}$ (%)	$AP_{Traffic\ sign}$ (%)	$AP_{Traffic\ light}$ (%)	mAP_{WIOT} (%)
R-FCN [11]	58.02	55.67	51.96	45.05	52.86	48.17	44.54	39.65	34.45	41.86
Mask R-CNN [19]	73.08	68.88	66.27	62.08	67.61	66.11	64.64	56.79	55.21	60.69
SSD [35]	46.46	44.33	39.67	35.46	41.52	37.15	35.26	33.47	24.32	32.62
RetinaNet [34]	68.05	62.67	60.96	58.05	62.57	59.17	58.54	50.65	47.45	54.02
YOLOv4 [3]	**93.26**	**90.04**	**87.52**	**83.26**	**88.67**	**78.25**	**73.63**	**71.44**	**68.86**	**73.09**

Table 7 Inference time and speed of object detection models on BDD100K dataset

Model	CPU time (ms)	GPU time (ms)	CPU speed (fps)	GPU speed (fps)
R-FCN [11]	2967.36	178.96	3.37	5.58
Mask R-CNN [19]	3412.57	326.78	2.93	3.06
SSD [35]	**72.98**	**19.12**	**13.70**	**52.31**
RetinaNet [34]	3686.12	97.66	2.71	10.24
YOLOv4 [3]	121.53	20.19	8.23	49.52

– The recall of SSD is the lowest among all the models which shows a high miss detection rate for targets in all levels.

From Table 6 following inferences can be drawn:

– The one-stage detection model YOLOv4 achieves the highest detection accuracy for target detection in all levels.

Table 8 Performance of object detection models on BDD100K dataset

Model	Class	Sensitivity (%)	Specificity (%)
R-FCN [11]	Vehicle	68.20	70.24
	Pedestrian	62.69	63.16
	Traffic sign	58.43	60.87
	Traffic light	55.17	56.46
	mean	61.14	62.68
Mask R-CNN [19]	Vehicle	89.21	83.26
	Pedestrian	86.58	80.32
	Traffic sign	82.64	75.26
	Traffic light	79.76	73.15
	mean	84.58	77.80
SSD [35]	Vehicle	63.21	66.21
	Pedestrian	57.82	61.84
	Traffic sign	46.81	50.36
	Traffic light	42.49	44.27
	mean	54.59	55.69
RetinaNet [34]	Vehicle	81.19	85.12
	Pedestrian	80.27	82.16
	Traffic sign	79.81	81.49
	Traffic light	79.38	80.62
	mean	80.17	82.35
YOLOv4 [3]	Vehicle	**91.24**	**92.49**
	Pedestrian	**88.19**	**89.14**
	Traffic sign	**85.65**	**86.20**
	Traffic light	**83.14**	**84.16**
	mean	**87.06**	**87.90**

Table 9 Comparison of object detection models in terms of energy efficiency and other aspects on BDD100K dataset

Model	Backbone	Model size (MB)	Memory footprint (GB)	FLOPs (G)	Energy efficiency (J/image)
R-FCN [11]	ResNet-101	436.12	3.89	306.81	127.64
Mask R-CNN [19]	ResNeXt-101-FPN	573.48	5.34	614.23	168.19
SSD [35]	VGG-16	136.04	0.73	95.86	5.38
RetinaNet [34]	ResNet-101-FPN	471.74	3.32	359.32	47.31
YOLOv4 [3]	CSPDarkNet-53	269.53	1.89	179.65	7.16

- The two-stage detection model Mask R-CNN shows better detection accuracy over RetinaNet, R-FCN, and SSD.
- The SSD shows the lowest AP among all the models in all levels.

From Table 7 following inferences can be drawn:

- The one-stage detector SSD is faster than other two-stage and one-stage detection models.
- The YOLOv4 shows almost the same detection speed compared to SSD on GPU.
- The R-FCN is faster than Mask R-CNN and RetinaNet on CPU.
- The Mask R-CNN is slower than other two-stage and one-stage detection models on GPU.
- The RetinaNet is faster than R-FCN and Mask R-CNN on GPU.

From Table 8 following inferences can be drawn:

- The one-stage detection model YOLOv4 achieves high sensitivity and specificity for target detection in all classes.
- The two-stage detector Mask R-CNN shows low specificity than RetinaNet in all classes, indicating a high number of false detections.

From Table 9 following inferences can be drawn:

- The one-stage detection model SSD is more energy efficient due to its small model size and fewer FLOPs compared to other object detection models.
- The two-stage detector Mask R-CNN has a large model size, high memory footprint, more FLOPs, and is highly energy inefficient than other object detection models.
- The YOLOv4 is slightly less energy efficient compared to SSD.
- The R-FCN has fewer FLOPs and is more energy inefficient compared to RetinaNet.

In this evaluation, the vehicles are considered as large objects, pedestrians as medium objects, and traffic signs and traffic lights as small objects. Figure 13 shows the performance of object detection models on few test images. Researchers who are new to this field can visualize and understand the detection of road objects inferred by autonomous vehicles under practical constraints. This article selects few distinct images based on the occlusion and truncation levels to visually show the effect of each algorithm. The object detection algorithms from top to bottom are R-FCN, Mask R-CNN, SSD, RetinaNet, and YOLOv4. It can be seen that in large target detection, all models have good performance, but there is a huge difference in the detection of small targets. The selection of YOLOv4 can be considered best for target detection in all levels. The SSD has started to miss targets as

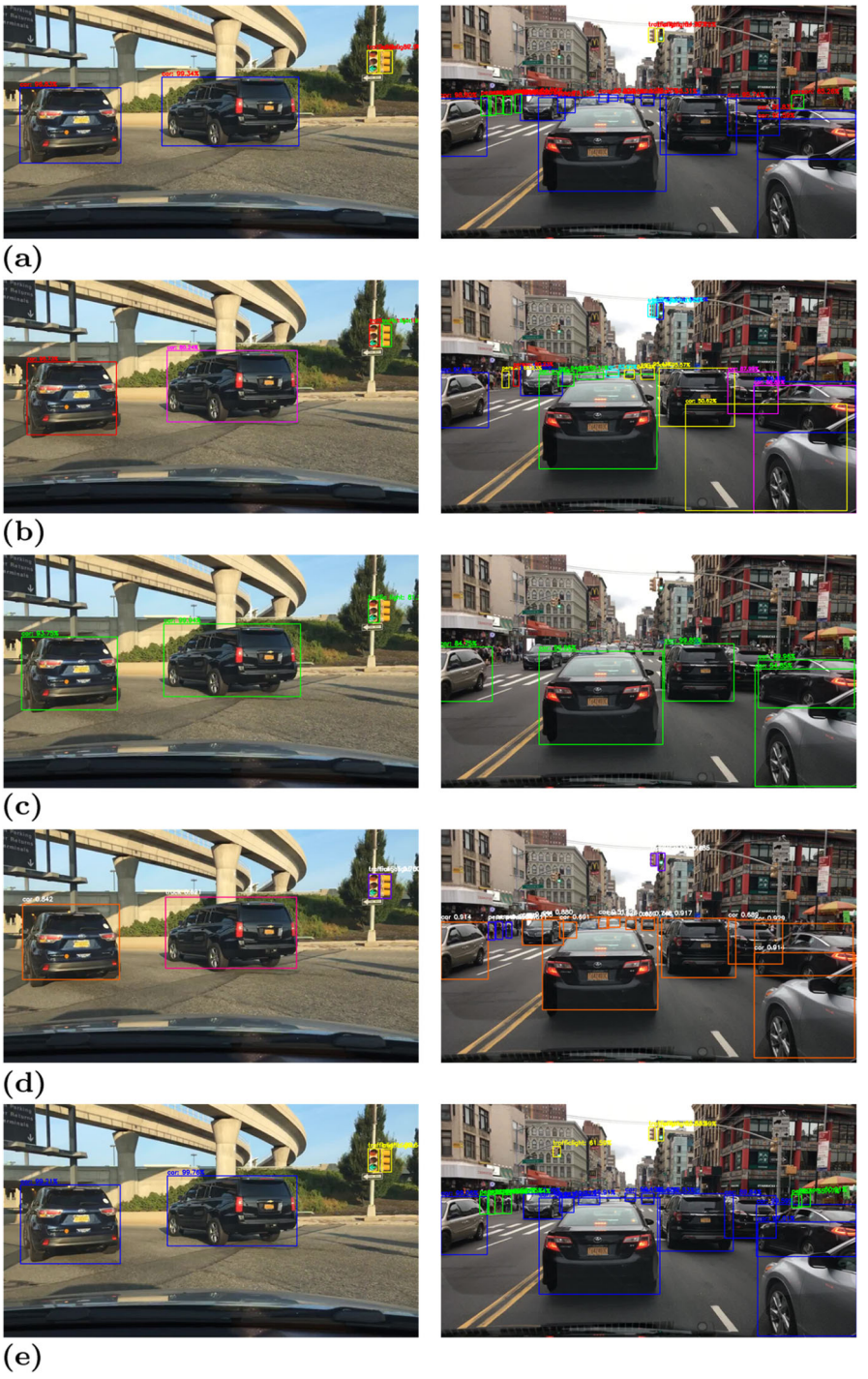

Fig. 13 Performance of five deep learning models in two category levels (from left to right columns: WOT and WIOT) (**a**) R-FCN, (**b**) Mask R-CNN, (**c**) SSD, (**d**) RetinaNet, and (**e**) YOLOv4

occlusion and truncation increase, the vehicles and pedestrians occluded in the lower left and right are not detected accurately. As the detection size reduces with occlusion and truncation, the YOLOv4 and, in few cases, Mask R-CNN detect the medium and small targets accurately.

In addition to further test the accuracy on the BDD100K dataset, to investigate the generalization ability of each model under complex weather scenarios, various models are used to test under different road environmental conditions during the clear, partly cloudy, overcast (as shown in Fig. 14), foggy, snowy, and rainy (as shown in Fig. 15) at different times of day and night. Researchers who are new to this field can visualize and understand the impact of challenging weather environments on the performance of the models

Fig. 14 Performance of five deep learning models in different road environmental conditions (from left to right columns: clear, partly cloudy, and overcast) (**a**) R-FCN, (**b**) Mask R-CNN, (**c**) SSD, (**d**) RetinaNet, and (**e**) YOLOv4

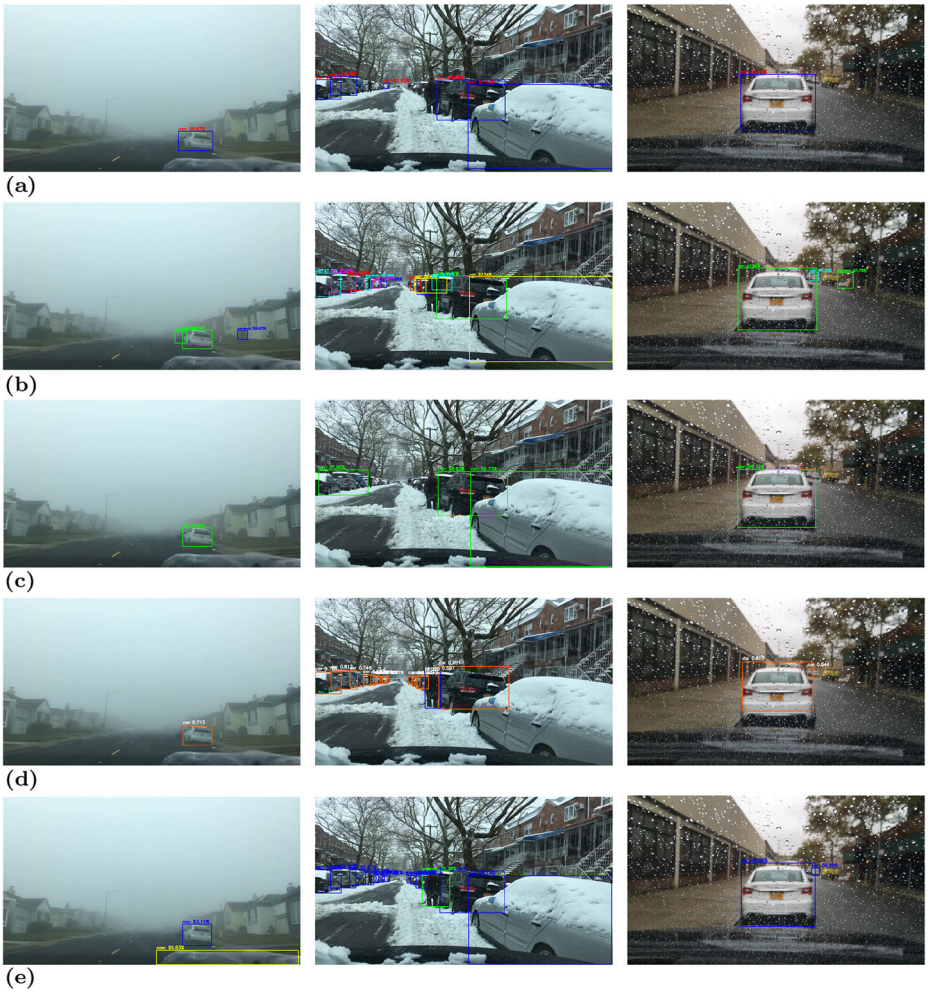

Fig. 15 Performance of five deep learning models in different road environmental conditions (from left to right columns: foggy, snowy, and rainy) (**a**) R-FCN, (**b**) Mask R-CNN, (**c**) SSD, (**d**) RetinaNet, and (**e**) YOLOv4

inferred by autonomous vehicles. The generalization ability of models is measured by overall detection accuracy on four classes from different weather images in the test dataset. Table 10 lists the generalization ability of object detection models using mAP values for all road weather conditions at daytime, dawn/dusk, and night. The criteria for generalization ability are set as low (< 35), medium (≥ 35 and < 70), and high (≥ 70). The one-stage detector YOLOv4 has shown to be very strong in the generalization of real scenarios with an excellent detection capability in other road environments as well. Moreover, the two-stage detector Mask R-CNN has shown very high generalization ability with the detection for the majority of road targets in low-lighting conditions, especially in the night time.

Table 10 Generalization ability of object detection models on BDD100K dataset

Model	Weather scenarios						
		mAP_{clear} (%)	$mAP_{Partly\ cloudy}$ (%)	$mAP_{overcast}$ (%)	mAP_{foggy} (%)	mAP_{snowy} (%)	mAP_{rainy} (%)
R-FCN [11]	daytime	72.43	71.14	70.23	52.14	34.05	25.86
	dawn/dusk	71.85	70.19	59.39	33.58	28.43	21.37
	night	71.26	70.03	50.13	27.28	22.69	18.15
Mask R-CNN [19]	daytime	77.16	74.29	73.07	67.58	48.06	36.47
	dawn/dusk	75.14	72.06	71.58	59.31	34.53	30.62
	night	73.62	71.80	70.62	47.28	29.47	25.34
SSD [35]	daytime	70.11	57.82	49.54	40.31	26.20	20.76
	dawn/dusk	59.62	41.48	36.84	30.20	23.34	16.96
	night	47.01	34.07	31.17	24.29	19.47	14.38
RetinaNet [34]	daytime	74.20	71.46	70.63	64.21	43.84	33.62
	dawn/dusk	72.46	70.35	66.14	56.75	33.46	28.17
	night	66.86	62.98	56.36	34.08	28.65	22.37
YOLOv4 [3]	daytime	83.52	78.19	75.76	70.85	61.64	45.23
	dawn/dusk	79.30	76.73	73.21	63.83	49.47	34.49
	night	74.87	71.48	70.24	49.65	34.27	29.23

4.2 Analysis of results

The two-stage detectors generally showed better detection accuracy compared to one-stage detectors. As per the above experimental results, it is observed that the one-stage detection algorithm YOLOv4 has outstanding performance in terms of speed and accuracy, and reasonably high energy efficiency than other one-stage and two-stage detection algorithms for road object detection because it uses a new backbone, CSPDarkNet-53, which increases the accuracy of classifier and detector. Moreover, adding SPP block with PANet increases the receptive field that separates the most significant context features without reducing the network speed. This makes the YOLOv4 detection accuracy exceed the two-stage detectors, with the advantage of fast detection speed on the BDD100K dataset. The two-stage detection algorithm Mask R-CNN shows good detection accuracy because it uses RoIAlign to produce a more accurate pixel-to-pixel bounding box alignment. As shown in Figs. 11 and 12, the SSD shows a low precision and recall because the model runs detection for medium and small targets from very few layers; therefore, it has insufficient semantic and edge information for detecting the majority of road objects. Whereas the R-FCN shows a low precision and recall because it does not use global average pooling. Therefore, final predictions are based on the position-sensitive feature maps decreasing the overall precision and recall. The Mask R-CNN shows a low precision because it ignores the spatial information between receptive fields, which means that it does not consider the instance details between the edge pixels. Also, the Mask R-CNN uses multi-scale FPN, and the RoIAlign module reduces misalignments between classification confidence and predicted boxes, improving the detection of medium and small targets. The recall rate of Mask R-CNN is close to YOLOv4 for object detection with occlusion and truncation because the number of false detections is significantly higher than correct detections; therefore, it cannot be considered a balanced algorithm. In this article, it is found that the accuracy of two-stage models is lower than one-stage models. As the detection level changes, the performance decreases with the occlusion and truncation transformation, particularly for the SSD because it has started to miss targets. The SSD shows high precision and low recall, thereby increasing the miss detection rate, indicating that it was more unchecked while detecting medium and small targets; therefore, it cannot be considered a balanced algorithm. On the contrary, the RetinaNet has strong adaptability to multi-scale images because of using FPN and Focal Loss; therefore, it achieves good results for medium and large target detection. As shown in Table 6, the RetinaNet can be considered as a balanced algorithm. Still, the YOLOv4 is a more balanced algorithm than other models because it maintains high precision on high recall in all category levels for all road objects.

From Table 7, it is clear that SSD can be considered a fast object detection algorithm in two-stage and one-stage detection models without any additional hardware requirements. The detection speed of YOLOv4 is almost the same compared to SSD with GPU requirements. The detection speed of Mask R-CNN is slow because of using an additional branch by RoIAlign module, which increases the computation time for final classification and bounding box regression. As shown in Table 8, the YOLOv4 shows high sensitivity and specificity in all classes. Meanwhile, the two-stage detection algorithm Mask R-CNN shows low mean specificity as the RPN fails to perform bounding box regression on small objects, which indicates a high number of false detections. The RetinaNet shows higher mean specificity than Mask R-CNN due to the hard mining of negative classes during training.

As shown in Table 9, the two-stage detector Mask R-CNN has a large model size and high memory footprint. In addition, the Mask R-CNN shows a high number of FLOPs

because it exploits excessive receptive fields for overlapped detection of objects, resulting in increased computations and more energy consumption. The R-FCN uses a large number of RPN calculations and lacks optimization of anchor scales and ratios, which leads to inefficient energy consumption. The one-stage detection model SSD has a small model size and fewer FLOPs, resulting in high energy efficiency. Although YOLOv4 has slightly more energy consumption than SSD due to more parameters but yields better results than other object detection models. The RetinaNet uses RPN to filter many negative candidate proposals, resulting in more FLOPs and energy consumption than other one-stage detection models.

To further analyze the detection performance, this article uses specific test samples similar to the actual road scenario from the BDD100K test dataset. As shown in Table 10, the performance results of YOLOv4 show excellent generalization ability even in complex and low-lighting conditions because of using a robust feature extractor, CSPDarkNet-53, but limited in rainy weather conditions. The generalization ability of the two-stage detector Mask R-CNN is very high, especially in low-lighting scenarios because of using RoIAlign layer for accurate localization but is limited in snowy and rainy conditions. The R-FCN shows high generalization ability, especially in low-lighting conditions, but is limited in foggy, snowy, and rainy weather conditions. The RetinaNet also shows high generalization ability under different weather scenarios but is very limited in snowy, rainy, and low-lighting conditions, especially in the night time. The SSD shows low generalization ability, especially in foggy, snowy, rainy, and low-lighting conditions.

As per the evaluation of the above experimental results, Table 11 summarizes the comparison of various object detection algorithms in terms of overall detection accuracy (average of mAP_{WOT} and mAP_{WIOT} from Table 6), inference speed (GPU speed from Table 7), computational complexity (FLOPs from Table 9), and efficiency (energy efficiency from Table 9). The criteria for overall detection accuracy are set as low (< 50), medium (≥ 50 and < 75), and high (≥ 75). The criteria for inference speed are set as fast (> 30), moderate (≤ 30 and > 10), and slow (≤ 10). The criteria for computational complexity are set as low (< 100), medium (≥ 100 and < 350), and high (≥ 350). The criteria for energy efficiency are set as low (> 50), medium (≤ 50 and > 10), and high (≤ 10).

In summary, the one-stage detection model YOLOv4 has shown outstanding accuracy with fast detection speed and high energy efficiency, satisfying the current standards of real-time object detection, and can be deployed to the autonomous driving environment for actual vehicles. The two-stage detector Mask R-CNN shows good detection accuracy and high generalization under various low-lighting scenarios but cannot be applied to real-time applications due to its slow speed and high computing requirements, and is more suitable for applications, such as identifying drivable areas, lane detection, etc. The RetinaNet shows

Table 11 Summary of various performances of object detection models on BDD100K dataset

Model	Accuracy	Speed	Complexity	Efficiency
R-FCN [11]	Low	Slow	Medium	Low
Mask R-CNN [19]	Medium	Slow	High	Low
SSD [35]	Low	Fast	Low	High
RetinaNet [34]	Medium	Moderate	High	Medium
YOLOv4 [3]	High	Fast	Medium	High

good accuracy but has slow detection speed and cannot be applied to real-time applications when the vehicle is running unless its hardware requirements are fulfilled. It is more suitable for low speed scenarios, such as toll system, automated parking system, etc. The R-FCN has shown low accuracy and is more suitable for applications like traffic detection, obstacle detection, etc. The SSD shows extremely fast detection speed but low accuracy and is highly energy efficient without any additional hardware requirements, and can be suitable for deployment on various embedded devices and mobile platforms due to its small model size and low power requirements, such as Unmanned Aerial Vehicles (UAVs) that can perform real-time detection with limited computing power.

4.3 Discussion

In earlier two-stage and one-stage target detection models, the two-stage models always had better accuracy and slow speed than the one-stage models. From Table 11, it is clear that the one-stage detection algorithm YOLOv4 achieves high detection accuracy while maintaining fast detection speed and high energy efficiency outperforming the two-stage detection algorithm Mask R-CNN because of using a new backbone network, CSPDarkNet-53. In addition, YOLOv4 uses SAT, BoF, and BoS methods to increase detection accuracy during detector training. The Mask R-CNN has shown high generalization ability because of using RoIAlign module for accurate localization. But it cannot be applied to real-time autonomous driving applications because of its high computational cost, slow detection speed, and inefficient energy consumption. The RetinaNet has shown good detection accuracy because it uses a special loss function called, Focal Loss to optimize the weights of easy samples; therefore, the model concentrates more on classifying difficult samples during training. With the use of RPN, RetinaNet can be suitable for applications that require good detection accuracy with certain hardware requirements, such as GPU for real-time performance. The R-FCN has low accuracy and cannot be applied to real-time applications due to its slow detection speed and low energy efficiency. Through comparative analysis, this article finds that even though SSD is an old object detection algorithm, it still has a huge advantage in real-time detection compared to other object detection algorithms without any additional hardware requirements. The SSD uses a simple feature extraction network that effectively reduces the model size and complexity. In addition, the SSD has low computation requirements and lightweight architecture, thereby making it highly energy efficient. This article considers SSD suitable for deployment on embedded devices and mobile platforms, such as UAVs to perform real-time environment perception in autonomous driving applications instead of traditional-feature based object detection algorithms. The YOLOv4 has low computation requirements, but the network introduces certain complexity in the model depending on the selection of architecture and other features. This article finds that the YOLOv4 can be very efficient in detecting difficult road targets under complex road scenarios and weather conditions. It is worth mentioning that YOLOv4 sometimes fails to perform detections on small occluded and truncated objects. In addition, this article also tests various deep learning-based algorithms under different road scenarios. The YOLOv4 has shown excellent generalization ability in multiple road environmental scenarios. Though YOLOv4 has a medium complexity, it satisfies the current requirements for real-time autonomous driving applications with strong detection robustness, thereby proving its feasibility in practical engineering. The YOLOv4 model introduces many new features, and through the combination of various such features, state-of-the-art results can be achieved.

In the past year, the YOLO algorithm was upgraded to its fourth and fifth versions. The YOLOv4 was proposed in April 2020, and a month later, YOLOv5 was released in May

2020. The YOLOv5 architecture uses CSPDarkNet as a backbone network, PANet as a neck, and YOLOv3 based detection principle, similar to YOLOv4. Moreover, the key improvements of YOLOv5 include Mosaic data augmentation (same as YOLOv4) and auto-learning anchor-based bounding box. The integration of anchor box selection is an engineering difference in which the network automatically learns the best shape and size of different anchor boxes for a given dataset during training. The authors of YOLOv5 claim to achieve better detection speed and accuracy than earlier object detection models. Furthermore, the YOLOv5 releases four different model versions depending upon the number of parameters, namely YOLOv5s (small), YOLOv5m (medium), YOLOv5l (large), and YOLOv5x (extra-large). The YOLOv5s has a small model size and can be suitable for real-time deployment in autonomous driving, similar to YOLOv4, but has low detection accuracy for complex road targets, such as traffic signs and traffic lights. In addition, even though YOLOv5x may show a high detection accuracy, but has a large model size and high computational cost compared to YOLOv4. Therefore, it does not satisfy the current requirements for real-time autonomous driving. As per the official YOLOv5 documentation, training the YOLOv5x over a large dataset requires high-end GPU devices, but the limitations we find are in terms of computational resources. A more detailed analysis of YOLOv5 with object detection models on a multi-GPU setup will be considered in future work.

To the best of our knowledge, this is the first comparative study to show the performance of one-stage detection models exceeds the two-stage detection models on a diverse large-scale dataset. This article selects five models based on significant improvements made over the decade and train the individual models with Mosaic data augmentation to increase the learning ability under complex scenarios. In addition, various practical metrics, such as model size, memory footprint, computational complexity, and energy efficiency are measured to provide a more detailed analysis of object detection models for road target detection. Researchers who are deeply passionate to analyze and apply deep learning-based algorithms in the field of ITS can get inspiration from this comparative study; for instance, the researchers can apply various new features to existing models, such as Mosaic training, SAT, and CmBN to efficiently train the model at low cost and improve the detection performance. The selection of optimal hyper-parameters, modifying the loss function, and hard training difficult samples instead of adding more layers in the model are additional possibilities that researchers can explore. A trade-off between computational cost and efficiency can be a practical consideration for improving real-time model accuracy. Moreover, accurate detection of very small target objects, such as traffic lights under low-lighting conditions with low energy consumption, is still a challenging task to be solved in the future of deep learning.

5 Development trends

In general, the object detection procedures involve an initial step before producing the final output, known as post-processing. This operation is computationally inexpensive when compared to the detection time. In most cases, the highest prediction result of the detected object is used for the accuracy calculation. An efficient post-processing method can improve the performance of many object detection models with minimum computational requirements. A common post-processing method, such as non-maximal suppression [55] and its improvements, can eliminate high IoU and high classification confidence objects, resulting in incorrect detection and classification. Therefore, exploiting a more simple, efficient,

and accurate post-processing technique can be an interesting direction for researchers in the object detection domain.

Recently, many researchers have proposed several deep learning-based models for object detection, but the solutions are limited to a strict local environment due to the high complexity of dynamic scenarios. The pre-existing object detectors, i.e., two-stage focus on high localization and precision, and one-stage focus on high inference speed [36], both have advantages and disadvantages in the practical engineering field. Further, they use multi-scale anchors for learning bounding box coordinates to improve accuracy; still, it is difficult to select optimal parameters of anchors. Therefore, to address this issue and fully inherit the advantages of both types of detectors while overcoming their limitations, advanced anchor-free detectors have attracted much research in recent times. Although these methods achieve better efficiency, they usually compromise accuracy. Therefore, maintaining the balance between accuracy and computational complexity remains a big challenge for many researchers.

The generic deep learning-based object detectors rely heavily on experiences that have contributed to more sophisticated algorithms over the years. As the computing power has grown exponentially, the field of Automated Machine Learning (AutoML) [21], more specifically neural architecture search has shown promising outcomes in object recognition tasks with the primary aim to reduce human assistance. Despite their recent success, the search approaches require extra time and computations, in addition to the architecture search cost. However, the search for optimal hyper-parameters for a model is limited to a single benchmark analysis. Also, in real-world scenarios, the search for an optimal architecture may not be adaptive without additional training. Therefore, designing characteristics of new object detection models with computational constraints in real-time scenarios using neural architecture search can be an interesting future direction for researchers in the domain.

Several improvements have been proposed to many existing object detection models over the years, but it is found that the existing models can be very difficult to improve under the original framework because of their applications in specific domains. Therefore, a new set of object detection models named EfficientDet [51] was proposed by M. Tan et al. in 2020. EfficientDet [51] uses a weighted Bi-directional Feature Pyramid Network (BiFPN) and EfficientNet [51] backbones to achieve better accuracy and efficiency in real-time scenarios. EfficientDet-D7 achieved state-of-the-art accuracy of 53.7 on MS COCO dataset [51]. Designing many new features with optimal parameters for backbone and detector can achieve state-of-the-art results, which can be a very interesting challenge for researchers.

Many deep learning-based object detectors that aim for high accuracy require a significant amount of computing resources to meet real-time requirements. As the backbone network accounts for more than 80% of the computations for object detection tasks, these models are not suitable for deployment on edge devices and mobile platforms. In recent years, a new area in research has progressed at designing efficient networks with computational resource constraints, known as lightweight networks. Recently proposed lightweight networks, including MobileNets [58], SqueezeNet [54], and ShuffleNet [37] are highly energy efficient and have shown excellent performance for image classification tasks. However, for more complicated tasks, such as object detection and segmentation, they still lack accuracy by a large margin. Therefore, with the rise of more on-device applications, designing small, efficient, and accurate lightweight detectors can be a very interesting future direction for researchers.

With the advances of 3D sensors, such as LiDAR point cloud [31], 3D point cloud [31], and RGB-D depth sensor [31], 3D object detection with multiple modalities of data has

become an active research area in the field of computer vision. The LiDAR point cloud provides reliable depth information and can accurately locate objects and characterize their shapes, but the detection is relatively sparse and complex using only point clouds. The LiDAR point cloud can be combined with RGB techniques for a more accurate scene understanding to exploit various tasks, including shape classification, 3D object detection and tracking, and 3D segmentation of complex objects. The object recognition techniques based on the combination of point clouds and RGB methods can outperform the 2D counterparts, but satisfying the real-time requirements and safety concerns with an increase in complexity can be an interesting challenge for researchers in the domain.

6 Conclusion

In this article, we present a comparative study on five popular and independent deep learning-based algorithms for road object detection. The BDD100K dataset is used for training, validating, and testing the individual deep learning models to detect four primary road objects, namely vehicles, pedestrians, traffic signs, and traffic lights. The comprehensive performances of the object detection models are compared using four primary metrics: (1) Precision rate, recall rate, AP of four object classes, and mAP on the BDD100K dataset; (2) Inference time and speed on CPU and GPU on the BDD100K dataset; (3) Model size, memory footprint, computational complexity, and energy efficiency on the BDD100K dataset; and (4) Generalization ability using mAP values on different road weather scenarios at various times of day and night to intuitively evaluate the applicability of individual algorithms on the BDD100K dataset. Besides, this work also computes mean sensitivity and specificity for each model. This article provides a benchmark illustrating researchers for comparing results on popular deep learning-based object detection algorithms for road target detection in autonomous driving applications. With the fast developments in the field of intelligent driving, deep learning-based object detection is still a subject worthy of studying. In order to deploy on more accurate scenarios for real-time detection, the need for high accuracy and efficient detection systems is becoming more and more urgent. The main challenge is to balance accuracy, efficiency, and real-time performance. Although recent achievements have been proven effective, a lot of research is still required for solving this challenge.

Appendix

As stated in the related works section of this article, there is some overlap with [53]. There are eight main differences between the proposed article and [53]. (1) This article compares up-to-date algorithms; for instance, Mask R-CNN [19] was released in early 2017, but the authors in [53] focused on Faster R-CNN [42]. (2) Additionally, this article considers four primary classes for road object detection, namely pedestrians, vehicles, traffic signs, and traffic lights on a diverse large-scale BDD100K dataset [57]. In contrast, [53] considered only a single class, namely vehicles on a small dataset. (3) In addition to basic augmentation methods, unlike [53], this article applies Mosaic data augmentation [27] during training of each model. (4) This article computes inference time and speed on CPU and GPU, while [53] evaluated frame per second. (5) Moreover, this article provides a detailed comparison on various practical metrics, such as model size, memory footprint, and energy efficiency. In contrast, the authors in [53] do not provide any comparison on such metrics. (6) This article

uses precise numerical representation for all metrics, including sensitivity, specificity, and computational complexity. In contrast, [53] provided subjective representation to sensitivity, specificity, and complexity. However, it is unclear what each level of indicator represents and how these metrics were evaluated in [53]. (7) Furthermore, this article computes generalization ability based on mAP values on different road weather scenarios at various times of day and night, whereas [53] provided subjective analysis on the generalization ability of models. (8) Lastly, unlike [53], this article provides a more in-depth and systematic analysis with supporting tables and figures.

References

1. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M et al (2016) Tensorflow: a system for large-scale machine learning. In: 12th {USENIX}, symposium on operating systems design and implementation ({OSDI}), vol 16, pp 265–283
2. Aziz L, bin Haji Salam S, Ayub S (2020) Exploring deep learning-based architecture, strategies, applications and current trends in generic object detection. A comprehensive review. IEEE Access
3. Bochkovskiy A, Wang CY, Liao HYM (2020) Yolov4: optimal speed and accuracy of object detection. arXiv:200410934
4. Braun M, Krebs S, Flohr F, Gavrila DM (2019) Eurocity persons: a novel benchmark for person detection in traffic scenes. IEEE Transactions on Pattern Analysis and Machine Intelligence 41(8):1844–1861
5. Broggi A, Cardarelli E, Cattani S, Medici P, Sabbatelli M (2014) Vehicle detection for autonomous parking using a soft-cascade adaboost classifier. In: IEEE Intelligent vehicles symposium proceedings. IEEE, pp 912–917
6. Caesar H, Bankiti V, Lang AH, Vora S, Liong VE, Xu Q, Krishnan A, Pan Y, Baldan G, Beijbom O (2020) nuscenes: a multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11621–11631
7. Cai Z, Vasconcelos N (2018) Cascade r-cnn: delving into high quality object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6154–6162
8. Chen Z, Chen K, Chen J (2013) Vehicle and pedestrian detection using support vector machine and histogram of oriented gradients features. In: International conference on computer sciences and applications. IEEE, pp 365–368
9. Chen L, Lin S, Lu X, Cao D, Wu H, Guo C, Liu C, Wang FY (2021) Deep neural network based vehicle and pedestrian detection for autonomous driving. A survey. IEEE Transactions on Intelligent Transportation Systems
10. Coppola P, Silvestri F (2019) Autonomous vehicles and future mobility solutions. In: Autonomous vehicles and future mobility
11. Dai J, Li Y, He K, Sun J (2016) R-fcn: object detection via region-based fully convolutional networks. arXiv:160506409
12. Devi S, Malarvezhi P, Dayana R, Vadivukkarasi K (2020) A comprehensive survey on autonomous driving cars: a perspective view. Wirel Pers Commun 114:3
13. Feng D, Haase-Schütz C, Rosenbaum L, Hertlein H, Glaeser C, Timm F, Wiesbeck W, Dietmayer K (2020) Deep multi-modal object detection and semantic segmentation for autonomous driving: datasets, methods, and challenges. IEEE Trans Intell Transp Syst 22(3):1341–1360
14. Feng D, Harakeh A, Waslander S, Dietmayer K (2020) A review and comparative study on probabilistic object detection in autonomous driving. arXiv:201110671
15. Garcia-Garcia A, Orts-Escolano S, Oprea S, Villena-Martinez V, Martinez-Gonzalez P, Garcia-Rodriguez J (2018) A survey on deep learning techniques for image and video semantic segmentation. Appl Soft Comput 70:41–65
16. Geiger A, Lenz P, Stiller C, Urtasun R (2012) Are we ready for autonomous driving? The Kitti vision benchmark suite. In: Conference on computer vision and pattern recognition (CVPR)
17. Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587
18. Gupta A, Mahaur B (2021) An improved dv-maxhop localization algorithm for wireless sensor networks. Wirel Pers Commun 117(3):2341–2357

19. He K, Gkioxari G, Dollár P, Girshick R (2017) Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision, pp 2961–2969
20. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
21. He X, Zhao K, Chu X (2021) Automl: a survey of the state-of-the-art. Knowl-Based Syst 212:106622
22. Hnewa M, Radha H (2020) Object detection under rainy conditions for autonomous vehicles: a review of state-of-the-art and emerging techniques. IEEE Signal Process Mag 38(1):53–67
23. Huang Y, Chen Y (2020) Autonomous driving with deep learning: a survey of state-of-art technologies. arXiv:200606091
24. Huang X, Cheng X, Geng Q, Cao B, Zhou D, Wang P, Lin Y, Yang R (2018) The apolloscape dataset for autonomous driving. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 954–960
25. Husain AA, Maity T, Yadav RK (2019) Vehicle detection in intelligent transport system under a hazy environment: a survey. IET Image Process 14(1):1–10
26. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T (2014) Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM international conference on multimedia, pp 675–678
27. Jocher G et al (2021) ultralytics/yolov3: v9.5.0 - YOLOv5 v5.0 release compatibility update for YOLOv3. https://doi.org/10.5281/zenodo.4681234
28. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inform Process Syst 25:1097–1105
29. Kuutti S, Bowden R, Jin Y, Barber P, Fallah S (2020) A survey of deep learning applications to autonomous vehicle control. IEEE Transactions on Intelligent Transportation Systems
30. Lee Y, Hwang Jw, Lee S, Bae Y, Park J (2019) An energy and gpu-computation efficient backbone network for real-time object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp 0–0
31. Li H, Wang J, Xu L, Zhang S, Tao Y (2021) Efficient and accurate object detection for 3d point clouds in intelligent visual internet of things. Multimed Tools Appl, pp –38
32. Lienhart R, Maydt J (2002) An extended set of haar-like features for rapid object detection. In: Proceedings international conference on image processing. IEEE, vol 1, pp I–I
33. Lin TY, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2117–2125
34. Lin TY, Goyal P, Girshick R, He K, Dollár P (2017) Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp 2980–2988
35. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016) Ssd: single shot multibox detector. In: European conference on computer vision. Springer, pp 21–37
36. Liu L, Ouyang W, Wang X, Fieguth P, Chen J, Liu X, Pietikäinen M (2020) Deep learning for generic object detection: a survey. Int J Comput Vis 128(2):261–318
37. Majid Azimi S (2018) Shuffledet: real-time vehicle detection network in on-board embedded uav imagery. In: Proceedings of the European Conference on Computer Vision (ECCV) workshops, pp 0–0
38. Minaee S, Boykov YY, Porikli F, Plaza AJ, Kehtarnavaz N, Terzopoulos D (2021) Image segmentation using deep learning. A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence
39. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
40. Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271
41. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv:180402767
42. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: towards real-time object detection with region proposal networks. arXiv:150601497
43. Rybski PE, Huber D, Morris DD, Hoffman R (2010) Visual classification of coarse vehicle orientation using histogram of oriented gradients features. In: IEEE Intelligent vehicles symposium. IEEE, pp 921–928
44. Silva PB, Andrade M, Ferreira S (2020) Machine learning applied to road safety modeling: a systematic literature review. Journal of traffic and transportation engineering (English edition)
45. Simhambhatla R, Okiah K, Kuchkula S, Slater R (2019) Self-driving cars: evaluation of deep learning techniques for object detection in different driving conditions. SMU Data Sci Rev 2(1):23
46. Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arXiv:14091556

47. Sivaraman S, Trivedi MM (2013) Looking at vehicles on the road: a survey of vision-based vehicle detection, tracking, and behavior analysis. IEEE Trans Intell Transp Syst 14(4):1773–1795
48. Srivastava S, Narayan S, Mittal S (2021) A survey of deep learning techniques for vehicle detection from images. Journal of Systems Architecture, 102152
49. Sutskever I, Martens J, Dahl G, Hinton G (2013) On the importance of initialization and momentum in deep learning. In: International conference on machine learning. PMLR, pp 1139–1147
50. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
51. Tan M, Pang R, Le QV (2020) Efficientdet: scalable and efficient object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10781–10790
52. Tang Y, Zhang C, Gu R, Li P, Yang B (2017) Vehicle detection and recognition for intelligent traffic surveillance system. Multimed Tools Applic 76(4):5817–5832
53. Wang H, Yu Y, Cai Y, Chen X, Chen L, Liu Q (2019) A comparative study of state-of-the-art deep learning algorithms for vehicle detection. IEEE Intell Transp Syst Mag 11(2):82–95
54. Wu B, Iandola F, Jin PH, Keutzer K (2017) Squeezedet: unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 129–137
55. Xiao Y, Tian Z, Yu J, Zhang Y, Liu S, Du S, Lan X (2020) A review of object detection based on deep learning. Multimed Tools Appl 79(33):23729–23791
56. Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1492–1500
57. Yu F, Chen H, Wang X, Xian W, Chen Y, Liu F, Madhavan V, Darrell T (2020) Bdd100k: a diverse driving dataset for heterogeneous multitask learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2636–2645
58. Zaidi SSA, Ansari MS, Aslam A, Kanwal N, Asghar M, Lee B (2021) A survey of modern deep learning based object detection models. arXiv:210411892
59. Zhang S, Wen L, Bian X, Lei Z, Li SZ (2018) Single-shot refinement neural network for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4203–4212
60. Zou Z, Shi Z, Guo Y, Ye J (2019) Object detection in 20 years: a survey. arXiv:190505055

Affiliations

Bharat Mahaur[1] · Navjot Singh[2] · K. K. Mishra[1]

Bharat Mahaur
bharatmahaur@gmail.com

kkm@mnnit.ac.in

[1] Department of Computer Science and Engineering, Motilal Nehru National Institute of Technology Allahabad, Allahabad, UP, India

[2] Department of Information Technology, Indian Institute of Information Technology Allahabad, Allahabad, UP, India