Object Detection in Autonomous Cyber-Physical Vehicle Platforms: Status and Open Challenges

Abhishek Balasubramaniam and Sudeep Pasricha

1 Introduction

Autonomous vehicles (AVs) have received immense attention in recent years, in large part due to their potential to improve driving comfort and reduce injuries from vehicle crashes. It has been reported that more than 36,000 people died in 2019 due to fatal accidents on U.S. roadways [1]. AVs can eliminate human error and distracted driving that is responsible for 94% of these accidents [2]. By using sensors such as cameras, lidars, and radars to perceive their surroundings, AVs can detect objects in their vicinity and make real-time decisions to avoid collisions and ensure safe driving behavior.

AVs are generally categorized into six levels by the SAE J3016 standard [3] based on their extent of supported automation (see Table 1). While level 0–2 vehicles provide increasingly sophisticated support for steering and acceleration, they heavily rely on the human driver to make decisions. Level 3 vehicles are equipped with Advanced Driver Assistance Systems (ADAS) to operate the vehicle in various conditions, but human intervention may be requested to safely steer, brake, or accelerate as needed. Level 4 vehicles are capable of full self-driving mode in specific conditions but will not operate if these conditions are not met. Level 5 vehicles can drive without human interaction under all conditions.

Automotive manufactures have been experimenting with AVs since the 1920s. The first modern AV was designed as part of CMU NavLab's autonomous land

A. Balasubramaniam (✉)
Department of Electrical and Computer Engineering, Colorado State University, Fort Collins, CO, USA
e-mail: abhishek.balasubramaniam@colostate.edu

S. Pasricha
Colorado State University, Fort Collins, CO, USA
e-mail: sudeep@colostate.edu

V. K. Kukkala, S. Pasricha (eds.), *Machine Learning and Optimization Techniques for Automotive Cyber-Physical Systems*, https://doi.org/10.1007/978-3-031-28016-0_17

Table 1 SAE J3016 levels of automation

SAE level	Name	Driving environment monitor
0	No automation	Human driver
1	Driver assistance	
2	Partial driving automation	
3	Conditional driving automation	ADAS system
4	High driving automation	
5	Full driving automation	

vehicle project in 1984 with level 1 autonomy that was able to steer the vehicle while the acceleration was controlled by a human driver [4]. This was followed by an AV designed by Mercedes-Benz in 1987 with level 2 autonomy that was able to control steering and acceleration with limited human supervision [5]. Subsequently, most major auto manufacturers such as General Motors, Bosch, Nissan, and Audi started to work on AVs.

Tesla was the first company to commercialize AVs with their Autopilot system in 2014 that offered level 2 autonomy [6]. Tesla AVs were able to travel from New York to San Francisco in 2015 by covering 99% of the distance autonomously. In 2017, Volvo launched their Drive Me feature with level 2 autonomy, with their vehicles traveling autonomously around the city of Gothenburg in Sweden under specific weather conditions [7]. Waymo has been testing its AVs since 2009 and has completed 200 million miles of AV testing. They also launched their driverless taxi service with level 4 autonomy in 2018 in the metro Phoenix area in USA with 1000–2000 riders per week, among which 5–10% of the rides were fully autonomous without any drivers [8]. Cruise Automation started testing a fleet of 30 vehicles in San Francisco with level 4 autonomy in 2017, launched their self-driving Robotaxi service in 2021 [9]. Even though Waymo and Cruise support level 5 autonomy, their AVs are classified as level 4 because there is still no guarantee that they can operate safely in all weather and environmental conditions.

AVs rely heavily on sensors such as cameras, lidars, and radars for autonomous navigation and decision making. For example, Tesla AVs rely on camera data with six forward facing cameras and ultrasonic sensors. In contrast, Cruise AVs use a sensor cluster that consists of a radar in the front while camera and lidar sensors are mounted on the top of the AV to provide a 360-degree view of the vehicle surroundings [9]. One of the main tasks involved in achieving robust environmental perception in AVs is to detect objects in the AV vicinity using software-based object detection algorithms. Object detection is a computer vision task that is critical for recognizing and localizing objects such as pedestrians, traffic lights/signs, other vehicles, and barriers in the AV vicinity. It is the foundation for high-level tasks during AV operation, such as object tracking, event detection, motion control, and path planning.

The modern evolution of object detectors began 20 years ago with the Viola Jones detector [10] used for human face detection in real-time. A few years later, Histogram of Oriented Gradient (HOG) [11] detectors became popular for pedestrian detection. HOG detectors were then extended to Deformable Part-based

Models (DPMs), which were the first models to focus on multiple object detection [12]. With growing interest in deep neural networks around 2014, the Regions with Convolutional Neural Network (R-CNN) deep neural network model led to a breakthrough for multiple object detection, with a 95.84% improvement in Mean Average Precision (mAP) over the state-of-the-art. This development helped redefine the efficiency of object detectors and made them attractive for entirely new application domains, such as for AVs. Since 2014, the evolution in deep neural networks and advances in GPU technology have paved the way for faster and more efficient object detection on real-time images and videos [10]. AVs today rely heavily on these improved object detectors for perception, pathfinding, and other decision making.

This chapter discusses contemporary deep learning-based object detectors, their usage, optimization, and limitations for AVs. We also discuss open challenges and future directions.

2 Overview of Object Detectors

Object detection consists of two sub-tasks: localization, which involves determining the location of an object in an image (or video frame), and classification, which involves assigning a class (e.g., 'pedestrian', 'vehicle', 'traffic light') to that object. Figure 1 illustrates a taxonomy of state-of-the-art deep learning-based object detectors. We discuss the taxonomy of these object detectors in this section.

2.1 Two-Stage vs Single Stage Object Detectors

Two-stage deep learning based object detectors involve a two-stage process consisting of (1) region proposals and (2) object classification. In the region proposal stage, the object detector proposes several Regions of Interest (ROIs) in an input image that have a high likelihood of containing objects of interest. In the second stage, the most promising ROIs are selected (with other ROIs being discarded)

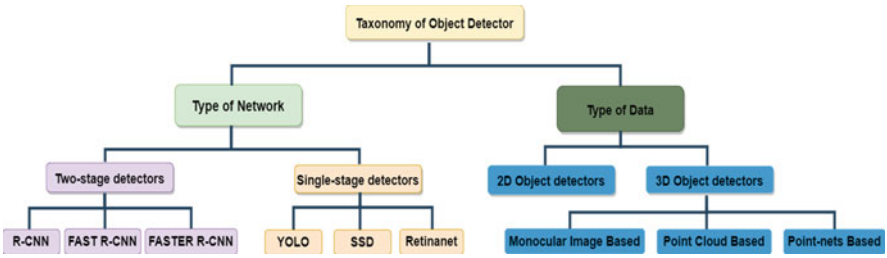

Fig. 1 Taxonomy of object detectors

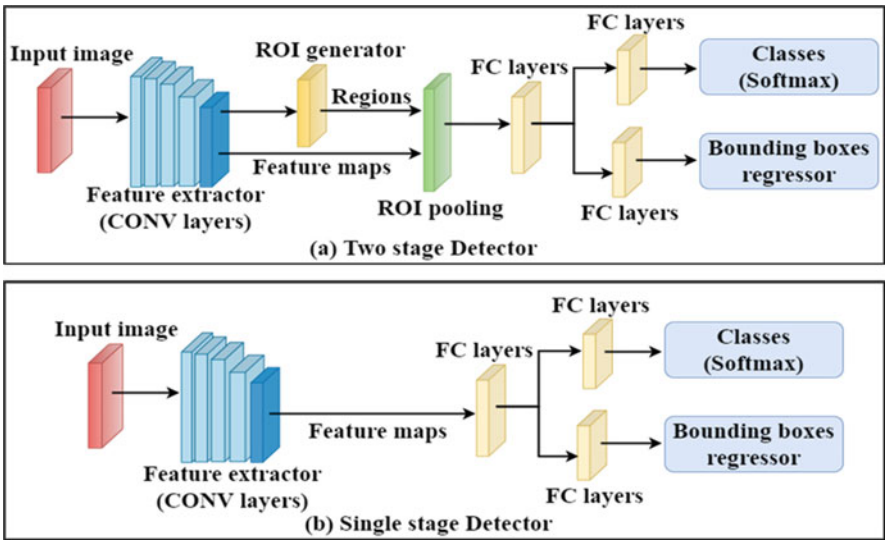

Fig. 2 Two-stage vs Single stage detector network diagram

and objects within them are classified [13]. Popular two-stage detectors include R-CNN, Fast R-CNN, and Faster R-CNN. In contrast, single-stage object detectors use a single feed-forward neural network that creates bounding boxes and classifies objects in the same stage. These detectors are faster than two-stage detectors but are also typically less accurate. Popular single-stage detectors include YOLO, SSD, EfficientNet, and RetinaNet.

Figure 2 illustrates the difference between the two types of object detectors. Both types of object detectors are typically evaluated using the mAP and Intersection over Union (IoU) accuracy metrics. mAP is the mean of the ratio of precision to recall for individual object classes, with a higher value indicating a more accurate object detector. IoU measures the overlap between the predicted bounding box and the ground truth bounding box. Formally, IoU is the ratio of the area of overlap between the (bounding and ground truth) boxes and the area of union between the boxes. Figure 3 illustrates the IoU of an object detector prediction and the ground truth. Figure 3a shows a highly accurate IoU and 3b shows a less accurate IoU.

R-CNN was one of the first deep learning-based object detectors and used an efficient selective search algorithm for ROI proposals as part of a two-stage detection [13]. Fast R-CNN solved some of the problems in the R-CNN model, such as low inference speed and accuracy. In the Fast R-CNN model, the input image is fed to a Convolutional Neural Network (CNN), generating a feature map and ROI projection. These ROIs are then mapped to the feature map for prediction using ROI pooling. Unlike R-CNN, instead of feeding the ROI as input to the CNN layers, Fast R-CNN uses the entire image directly to process the feature maps to detect objects [14]. Faster R-CNN used a similar approach to Fast R-CNN, but instead of using a

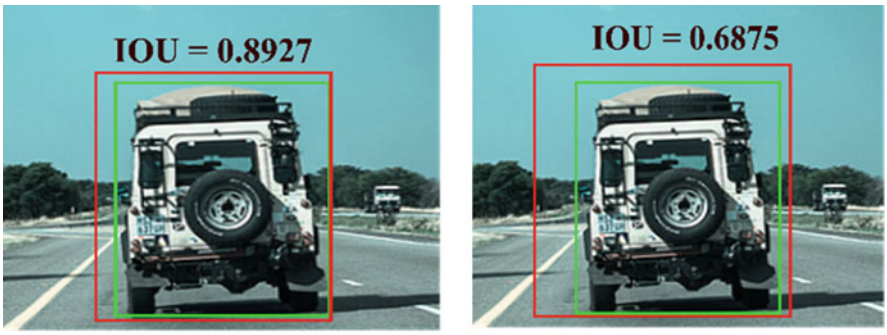

Fig. 3 Example of an IOU; green box: ground truth; red box: prediction

selective search algorithm for the ROI proposal, it employed a separate network that fed the ROI to the ROI pooling layer and the feature map, which were then reshaped and used for prediction [15].

Single-stage object detectors such as YOLO (You only look once) are faster than two-stage detectors as they can predict objects on an input with a single pass. The first YOLO variant, YOLOv1, learned generalizable representations of objects to detect them faster [16]. In 2016, YOLOv2 improved upon YOLOv1 by adding batch normalization, a high-resolution classifier, and use of anchor boxes to create bounding boxes instead of using a fully connected layer like YOLOv1 [17]. In 2018, YOLOv3 was proposed with a 53 layered backbone-based network that used an independent logistic classifier and binary cross-entropy loss to predict overlapping bounding boxes and smaller objects [18]. Single-Shot Detector (SSD) models were proposed as a better option to run inference on videos and real-time applications as they share features between the classification and localization task on the whole image, unlike YOLO models that generate feature maps by creating grids within an image. While the YOLO models are faster than SSD, they trail behind SSD models in accuracy [19]. Even though YOLO and SSD models provide good inference speed, they have a class imbalance problem when detecting small objects. This issue was addressed in the RetinaNet detector that used a focal loss function during training and a separate network for classification and bounding box regression [20].

In 2020, YOLOv4 introduced two important techniques: 'bag of freebies' which involves improved methods for data augmentation and regularization during training and 'bag of specials' which is a post processing module that allows for better mAP and faster inference [21]. YOLOv5, which was also introduced in 2020, proposed further data augmentation and loss calculation improvements. It also used auto-learning bounding box anchors to adapt to a given dataset [22]. Another variant called YOLOR (You Only Learn One Representation) was proposed in 2021 and used a unified network that encoded implicit and explicit knowledge to predict the output. YOLOR can perform multitask learning such as object detection, muti-label image classification, and feature embedding using a single model [23]. The YOLOX

Table 2 2D and 3D object detector models and their performance

Name	Year	Type	Dataset	mAP	Inference rate (fps)
R-CNN [13]	2014	2D	Pascal VOC	66%	0.02
Fast R-CNN [14]	2015		Pascal VOC	68.8%	0.5
Faster R-CNN [15]	2016		COCO	78.9%	7
YOLOv1 [16]	2016		Pascal VOC	63.4%	45
YOLOv2 [17]	2016		Pascal VOC	78.6%	67
SSD [19]	2016		Pascal VOC	74.3%	59
RetinaNet [20]	2018		COCO	61.1%	90
YOLOv3 [18]	2018		COCO	44.3%	95.2
YOLOv4 [21]	2020		COCO	65.7%	62
YOLOv5 [22]	2021		COCO	56.4%	140
YOLOR [23]	2021		COCO	74.3%	30
YOLOX [24]	2021		COCO	51.2%	57.8
Complex-YOLO [27]	2018	3D	KITTI	64.00%	50.4
Complexer-YOLO [28]	2019		KITTI	49.44%	100
Wen et al. [29]	2021		KITTI	73.76%	17.8
RAANet [30]	2021		NuScenes	62.0%	–

model, also proposed in 2021, uses an anchor-free, decoupled head technique that allows the network to process classification and regression using separate networks. Unlike the YOLOv4 and YOLOv5 models, YOLOX has reduced number of parameters and increased inference speed [24]. The performance of each model in terms of mAP and inference speed is summarized in Table 2.

2.2 2D vs 3D Object Detectors

2D object detectors typically use 2D image data for detection, but recent work has also proposed a sensor-fusion based 2D object detection approach that combines data from a camera and radar [25]. 2D object detectors provide bounding boxes with four Degrees of Freedom (DOF). Figure 4 shows the most common approach for encoding bounding boxes 4a: [x, y, height, width] and 4b: [xmin, ymin, xmax, ymax] [26]. Unfortunately, 2D object detection can only provide the position of the object on a 2D plane but does not provide information about the depth of the object. Depth of the object is important to predict the shape, size, and position of the object to enable improved performance in various self-driving tasks such as path planning, collision avoidance, etc.

Figure 5 shows the difference between a 2D and 3D object detector output on real images. 3D object detectors use data from a camera, lidar, or radar to detect objects and generator 3D bounding boxes. These detectors provide bounding boxes with (x, y, z) and (height, width, length) along with yaw information [26]. These object detectors use several approaches, such as point clouds and frustum pointnets,

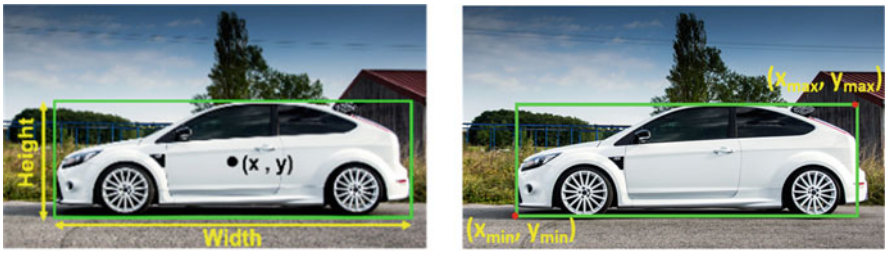

Fig. 4 Commonly used bounding box encoding methods

Fig. 5 Object detection modalities: (**a**) 2D vs. (**b**) 3D

for predicting objects in real-time. Point cloud networks can directly use 3D data, but the complexity and cost of computing are very high, so some networks use 2D to 3D lifting while compensating for the loss of information. Pointnets are used along with RGB images, where 2D bounding boxes are obtained using RGB images. Then these boxes are used as ROIs for 3D object detection which reduces the search effort [26]. Monocular image-based methods have also been proposed that use an RGB image to predict objects on the 2D plane and then perform 2D to 3D lifting to create 3D object detection results.

Recent years have seen growing interest in 3D object detection with deep learning. Complex-YOLO, an extension of YOLOv2, used a Euler Region Proposal Network (E-RPN), based on an RGB Birds-Eye-View (BEV) map from point cloud data to get 3D proposals. The network exploits the YOLOv2 network followed by E-RPN to get the 3D proposal [27]. Later in 2019, Complexer-YOLO achieved semantic segmentation and 3D object detection using Random Finite Set (RFS) [28]. The more recent work on 3D object detection by Wen et al. [29] in 2021 proposed a lightweight 3D object detection model that consists of three submodules: (1) point transform module, which extracts point features from the RGB image based on the raw point cloud, (2) voxelization, which divides the features into equally spaced voxel grids and then generates a many-to-one mapping between the voxel grids and the 3D point clouds, and (3) point-wise fusion module, which fuses the features using two fully connected layers. The output of the point-wise fusion module is

encoded and used as input for the model. Another 3D detector proposed in 2021 called RAANet used only lidar data to achieve 3D object detection [30]. It used the BEV lidar data as input for a region proposal network which was then used to create shared features. These shared features were used as the input for an anchor free network to detect 3D objects. The performance of these models is summarized in Table 2.

3 Deploying Object Detectors in AVs

Deploying deep learning-based object detector models in AVs has its own challenges, mainly due to the resource-constrained nature of the onboard embedded computers used in vehicles. These computing platforms have limited memory availability and reduced processing capabilities due to stringent power caps, and high susceptibility to faults due to thermal hotspots and gradients, especially during operation in the extreme conditions found in vehicles. As the complexity of the object detector model increases, the memory and computational requirements, and energy overheads also increase. In this section, we discuss techniques to improve object detector model deployment efficiency. The performance of some of the latest works on this topic is summarized in Table 3.

3.1 Pruning

Pruning a neural network model is a widely used method for reducing the model's memory footprint and computational complexity. Pruning was first used in the 1990s to reduce neural network sizes for deploying them on embedded platforms [31]. Pruning involves removing redundant weights and creating sparsity in the model by training the model with various regularization techniques (L1, L2, unstructured, and structured regularization). Sparse models are easier to compress, and the zero weights created during pruning can be skipped during inference, reducing inference time, and increasing efficiency. While most pruning approaches target deep learning models for the simpler image classification problem, relatively fewer works have attempted to prune the more complex object detector models. Wang et al. [32] proposed using a channel pruning strategy on SSD models in which they start by creating a sparse normalization and then prune the channels with a small scaling factor followed by fine-tuning the network. Zhao et al. [33] propose a compiler aware neural pruning search on YOLOv4 which uses an automatic neural pruning search algorithm that uses a controller and evaluator. The controller is used to select the search space, pre-layer pruning schemes, and prune the model whereas the evaluator evaluates the model accuracy after every pruning step.

Table 3 Different object detector model optimization techniques and their performance

Name	Technique used	Model compression achieved	Object detector	Latency improvement	Hardware used
Wang et al. [32]	Pruning	32.40%	SSD	33.61%	GTX 1080Ti
Zhao et al. [33]	Pruning	93.27%	YOLOv4	80.70%	Qualcomm Adreno 64
Fan et al. [34]	Quantization	75%	SSDLite-MobileNetV2	85.67%	Zynq ZC706
LCDet [35]	Quantization	77.79%	YOLOv2	13.66%	Snapdragon 835
Kang et al. [36]	Knowledge distillation	37.11%	RetinaNet	32.98%	–
Chen et al. [37]	Knowledge distillation	83.82%	RCNN	7.93%	–

3.2 Quantization

Quantization is the process of approximating a continuous signal by a set of discrete symbols or integer values. The discrete set is selected as per the type of quantization such as integer, floating-point, and fixed-point quantization. Quantizing deep learning-based object detector models involves converting the baseline 32-bit parameters (weights, activations, biases) to fewer (e.g., 16 or 8) bits, to achieve lower memory footprint, without significantly reducing model accuracy. Fan et al. [34] proposed an 8-bit integer quantization of all the bias, batch normalization, and activation parameters on SSDLite-MobileNetV2. LCDet [35] proposed a fully quantized 8-bit model in which parameters of each layer of a YOLOv2 object detector were quantized to 8-bit fixed point values. To achieve this, they first stored the minimum and maximum value at each layer and then used relative valued to linearly distribute the closest integer value to all the reduced bitwidth weights.

3.3 Knowledge Distillation

Knowledge Distillation involves transferring learned knowledge from a larger model to a smaller, more compact model. A teacher model is first trained for object detection, followed by a smaller student model being trained to emulate the prediction behavior of the teacher model. The goal is to make the student model learn important features to arrive at the predictions that are very close to that of the original model. The resulting student model reduces the computational power and memory footprint compared to the original teacher model. Kang et al. [36] proposed an instance-conditional knowledge decoding module to retrieve knowledge from the teacher network (RetinaNet with a ResNet-101 classifier model as backbone) via query-based attention. They also used a subtask that optimized the decoding module and feature maps to update the student network (RetinaNet with a simpler ResNet-50 model as backbone). Chen et al. [37] proposed a three-step knowledge distillation process on R-CNN with a Resnet-50 model as backbone. The first step used a feature pyramid distillation process to extract the output features that can mimic the teacher network features. They then used these features to remove the output proposal to perform Regional Distillation (RD), enabling the student (RCNN with a much simpler ResNet-18 model as backbone) to focus on the positive regions. Lastly, Logit Distillation (LD) on the output was used to mimic the final output of the teacher network.

4 Open Challenges and Opportunities

While there has been significant work on effective object detection for AVs, there are significant outstanding challenges that remain to be solved. Here we discuss some of the key challenges and opportunities for future research in the field.

Neural Architecture Search (NAS) In recent years, NAS based efforts have gained much attention to automatically determine the best backbone architecture for a given object detection task. Recent works such as NAS-FCOS [38], MobileDets [39], and AutoDets [40] have shown promising results on image classification tasks. Using automated NAS methods can help identify better anchors boxes and backbone networks to improve object detector performance. The one drawback of these efforts is that they take significantly longer to discover the final architecture. More research is needed to devise efficient NAS approaches targeting object detectors.

Real-Time Processing Object detectors deployed in AV's use video inputs from AV cameras, but the object detectors are typically trained to detect objects on image datasets. Detecting an object on every frame in a video can increase latency of the detection task. Correlations between consecutive frames can help identify the frames that can be used for detecting new objects (while discarding others) and reduce the latency of the model. Creating models that can correlate spatial and temporal relationships between consecutive frames is an open problem. Recent work on real-time object detection [41, 42], has begun to address this problem, but much more work is needed.

Sensor Fusion Sensor fusion is one of the most widely used methods for increasing accuracy of 2D and 3D object detection. Many efforts fuse lidar and RGB images to perform object detection for autonomous driving. But there are very few works that consider fusion data from ultrasonic sensors, radar, or V2X communication. The fusion of data from more diverse sensors is vital to increasing the reliability of the perception system in AVs. Fusing additional sensor data can also increase stability and ensure that the perception system does not fail when one of the sensors fails due to environmental conditions. Recent efforts [43, 44] are beginning to design object detectors that work with data from various sensors, which is a step in the right direction for reliable perception in AVs.

Time Series Information Most conventional object detection models rely on a CNN-based network for object detection that does not consider time series information. Only a few works, such as [45, 46], consider multi-frame perception that uses data from the previous and current time instances. Correlating time series information about vehicle dynamics can increase the reliability of the model. Some works such as Sauer et al. [47] and Chen et al. [48] have used time-series data such as steering angle, vehicle velocity, etc. with object detector output to create a closed loop autonomous driving system. Research on combining these efforts with time-series object detector outputs can enable us to make direct driving decisions from these multi-modal models for safer and more reliable driving.

Semi-supervised Object Detection Supervised machine learning methods which are used in all object detectors today require an annotated dataset to train the detector models. The major challenge in supervised object detection is to annotate data for different scenarios such as, but not limited to, weather conditions, terrain, variable traffic, and location, which is a time-consuming task to ensure improved safety and adaptability of these models in real-world AV driving scenarios. Due to the evolving changes in driving environments, the use of semi-supervised learning for object detection can reduce training time of these models. Some recent efforts, e.g., [49–51] advocate for performing object detection using semi-supervised transformer models. Due to the high accuracy of transformer-based models, they can yield better performance when detecting object for autonomous driving tasks. Even though transformer-based models yield higher accuracy, deploying them on embedded onboard computers is still a challenge due to their large memory footprint, which requires further investigation.

Open Datasets Object detector model performance can vary due to changing lighting, weather, and other environmental conditions. Data from different weather conditions during training can help fit all the environmental needs to address this problem. Adding new data to accommodate these weather conditions changes when training and testing these models can help overcome this issue. The Waymo open dataset [52] has a wide variety of data that focus on different lighting and weather conditions to overcome this issue. More such open datasets are needed to train reliable object detectors for AVs to ensure robust performance in a variety of environmental conditions.

Resource Constraints Most object detectors have high computational and power overheads when deployed on real hardware platforms. To address this challenge, prior efforts have adapted pruning, quantization, and knowledge distillation techniques (see Sect. 3) to reduce model footprint and decrease the model's computational needs. New approaches for hardware-friendly pruning and quantization, such as recent efforts [53, 54], can be very useful. Techniques to reduce matrix multiplication operations, such as [55–57] can also speed up object detector execution time. Hardware and software co-design, by combining pruning, quantization, knowledge distillation etc. along with hardware optimization such as parallel factors adjustment, resource allocation etc. also represents an approach to improve object detector efficiency. Results from recent work [58–60] have been promising, but much more research is needed on these topics.

5 Conclusion

In this chapter, we discussed the landscape of various object detectors being considered and deployed in emerging AVs, the challenges involved in using these object detectors in AVs, and how the object detectors can be optimized for lower computational complexity and faster inference during real-time perception. We also

presented a multitude of open challenges and opportunities to advance the state-of-the-art with object detection for AVs. As AVs are clearly the transportation industry's future, research to overcome these challenges will be crucial to creating a safe and reliable transportation model.

References

1. Automated Vehicles for Safety, NHTSA Report (2021)
2. Kukkala, V.K., Tunnell, J., Pasricha, S., Bradley, T.: Advanced driver-assistance systems: a path toward autonomous vehicles. IEEE Consum. Electron. Mag. **7**(5), 18–25 (2018)
3. J3016B: Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles – SAE international., https://www.sae.org/standards/content/j3016_201806
4. Staff, R.D.: Navlab: The self-driving car of the '80s. Rediscover the '80s (2016)
5. Dickmanns, E.D.: Dynamic Vision for perception and control of Motion (2010)
6. Lawler, R.: Riding shotgun in Tesla's fastest car ever (2014)
7. Drive Me, the World's most ambitious and advanced public autonomous driving experiment, starts today. Volvo Cars Global Media Newsroom, 2016
8. Safety report and Whitepapers, Waymo., https://waymo.com/safety/
9. McEachern, S.: Cruise Founder Takes Company's First Driverless Ride on SF Streets: Video. GM Authority (2021)
10. Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., Qu, R.: A survey of deep learning-based object detection. IEEE Access. **7**, 128837–128868 (2019)
11. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05) (Vol. 1, pp. 886–893). IEEE (2005)
12. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1627–1645 (2009)
13. Girshick, R., Donahue, J., Darrell, T., Malik, J.: 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 (2014)
14. Girshick, R.: Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448 (2015)
15. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: towards real-time object detection with region proposal networks. Adv. Neural Inf. Proces. Syst. **28** (2015)
16. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788 (2016)
17. Redmon, J., Farhadi, A.: YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7263–7271 (2017)
18. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint, arXiv:1804.02767 (2018)
19. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single shot multibox detector. In European conference on computer vision, pp. 21–37. Springer, Cham (2016)
20. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp. 2980–2988 (2017)
21. Bochkovskiy, A., Wang, C.Y., Liao, H.Y.M.: Yolov4: optimal speed and accuracy of object detection. arXiv preprint, arXiv:2004.10934 (2020)

22. Ultralytics; Ultralytics/yolov5: Yolov5 in PyTorch & ONNX & CoreML & TFLite, GitHub. https://github.com/ultralytics/yolov5
23. Wang, C.Y., Yeh, I.H., Liao, H.Y.M.: You only learn one representation: unified network for multiple tasks. arXiv preprint, arXiv:2105.04206 (2021)
24. Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J.: Yolox: Exceeding yolo series in 2021. arXiv preprint, arXiv:2107.08430 (2021)
25. Nobis, F., Geisslinger, M., Weber, M., Betz, J., Lienkamp, M.: A deep learning-based radar and camera sensor fusion architecture for object detection. In: 2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF), pp. 1–7, IEEE (2019)
26. Fang, J., Zhou, L., Liu, G.: 3d bounding box estimation for autonomous vehicles by cascaded geometric constraints and depurated 2d detections using 3d results. arXiv preprint, arXiv:1909.01867 (2019)
27. Simony, M., Milzy, S., Amendey, K., Gross, H.M.: Complex-yolo: an euler-region-proposal for real-time 3d object detection on point clouds. In: Proceedings of the European Conference on Computer Vision (ECCV) Workshops pp. 0–0 (2018)
28. Simon, M., Amende, K., Kraus, A., Honer, J., Samann, T., Kaulbersch, H., Michael Gross, H.: Complexer-yolo: Real-time 3d object detection and tracking on semantic point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 0–0 (2019)
29. Wen, L.H., Jo, K.H.: Fast and accurate 3D object detection for lidar-camera-based autonomous vehicles using one shared voxel-based backbone. IEEE Access. **9**, 22080–22089 (2021)
30. Lu, Y., Hao, X., Sun, S., Chai, W., Tong, M., Velipasalar, S.: RAANet: range-aware attention network for LiDAR-based 3D object detection with auxiliary density level estimation. arXiv preprint, arXiv:2111.09515 (2021)
31. Liang, T., Glossner, J., Wang, L., Shi, S., Zhang, X.: Pruning and quantization for deep neural network acceleration: a survey. Neurocomputing. **461**, 370–403 (2021)
32. Wang, Q., Zhang, H., Hong, X., Zhou, Q.: Small object detection based on modified FSSD and model compression. arXiv preprint, arXiv:2108.10503 (2021)
33. Zhao, P., Yuan, G., Cai, Y., Niu, W., Liu, Q., Wen, W., Ren, B., Wang, Y., Lin, X.: Neural pruning search for real-time object detection of autonomous vehicles. ACM/IEEE DAC. (2021)
34. Fan, H., Liu, S., Ferianc, M., Ng, H.C., Que, Z., Liu, S., Niu, X., Luk, W.: A real-time object detection accelerator with compressed SSDLite on FPGA. FPT. (2018)
35. Tripathi, S., Dane, G., Kang, B., Bhaskaran, V., Nguyen, T.: LCDet: low-complexity fully-convolutional neural networks for object detection in embedded systems. IEEE CVPRW. (2017)
36. Kang, Z., Zhang, P., Zhang, X., Sun, J., Zheng, N.: Instance-conditional knowledge distillation for object detection. Adv. Neural Info. Proces. Syst. (2021)
37. Chen, R., Ai, H., Shang, C., Chen, L., Zhuang, Z.: Learning lightweight pedestrian detector with hierarchical knowledge distillation. IEEE ICIP. (2019)
38. Wang, N., Gao, Y., Chen, H., Wang, P., Tian, Z., Shen, C., Zhang, Y.: NAS-FCOS: "efficient search for object detection architectures". Int. J. Comput. Vis. **129**, 3299–3312 (2021)
39. Xiong, Y., Liu, H., Gupta, S., Akin, B., Bender, G., Wang, Y., Kindermans, P.J., Tan, M., Singh, V., Chen, B.: MobileDets: searching for object detection architectures for mobile accelerators. IEEE/CVF CVPR. (2021)
40. Li, Z., Xi, T., Zhang, G., Liu, J., He, R.: AutoDet: pyramid network architecture search for object detection. Int. J. Comput. Vis. **129**, 1087–1105 (2021)
41. Zhu, H., Wei, H., Li, B., Yuan, X., Kehtarnavaz, N.: Real-time moving object detection in high-resolution video sensing, vol. 20. Sensors (2020)
42. Dai, X., Yuan, X., Wei, X.: TIRNet: object detection in thermal infrared images for autonomous driving. Appl. Intel. (2021)
43. Chavez-Garcia, R.O., Aycard, O.: Multiple sensor fusion and classification for moving object detection and tracking. IEEE TITS. **17**(2) (2016, Feb)

44. Cho, H., Seo, Y.W., Kumar, B.V., Rajkumar, R.R.: A multi-sensor fusion system for moving object detection and tracking in urban driving environments. IEEE ICRA. (2014)
45. Casas, S., Luo, W., Urtasun, R.: IntentNet: learning to predict intention from raw sensor data. Proc. 2nd Annu. Conf. Robot Learn. (2018)
46. Luo, W., Yang, B., Urtasun, R.: Fast and furious: real time endto-end 3D detection, tracking and motion forecasting with a single convolutional net. Proc. IEEE/CVF CVPR. (2018, Jun)
47. Sauer, A., Savinov, N., Geiger, A.: Conditional affordance learning for driving in urban environments. Proc. 2nd Annu. Conf. Robot Learn. (2018)
48. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: DeepDriving: learning affordance for direct perception in autonomous driving. Proc. ICCV. (2015, Dec)
49. Xie, E., Ding, J., Wang, W., Zhan, X., Xu, H., Sun, P., Li, Z., Luo, P.: DetCo: unsupervised contrastive learning for object detection. Proc. IEEE/CVF Int. Conf. Comp. Vision. (2021)
50. Dai, Z., Cai, B., Lin, Y., Chen, J.: UP-DETR: unsupervised pre-training for object detection with transformers. IEEE/CVF CVPR. (2021)
51. Bar, A., Wang, X., Kantorov, V., Reed, C.J., Herzig, R., Chechik, G., Rohrbach, A., Darrell, T., Globerson, A.: DETReg: unsupervised pre-training with region priors for object detection. arXiv preprint, arXiv:2106.04550 (2021)
52. Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V.: Scalability in perception for autonomous driving: Waymo open dataset. IEEE/CVF CVPR. (2020)
53. Chen, P., Liu, J., Zhuang, B., Tan, M., Shen, C.: AQD: towards accurate quantized object detection. IEEE/CVF CVPR. (2021)
54. Kim, S., Kim, H.: Zero-centered fixed-point quantization with iterative retraining for deep convolutional neural network-based object detectors. IEEE Access. **9**, 20828–20839 (2021)
55. Pilipović, R., Risojević, V., Božič, J., Bulić, P., Lotrič, U.: An approximate GEMM unit for energy-efficient object detection. Sensors. **21**, 4195 (2021)
56. Winograd, S.: Arithmetic complexity of computations, vol. 33. SIAM (1980)
57. Kala, S., Mathew, J., Jose, B.R., Nalesh, S.: UniWiG: unified Winograd-GEMM architecture for accelerating CNN on FPGAs. IEEE VLSID, 209–214 (2019)
58. Zhang, X., Lu, H., Hao, C., Li, J., Cheng, B., Li, Y., Rupnow, K., Xiong, J., Huang, T., Shi, H., Hwu, W.M.: SkyNet: a hardware-efficient method for object detection and tracking on embedded systems. Proc. Mach. Learn. Syst. **2**, 216–229 (2019)
59. Zhu, Y., Liu, Y., Zhang, D., Li, S., Zhang, P., Hadley, T.: Acceleration of pedestrian detection algorithm on novel C2RTL HW/SW co-design platform. IEEE ICGCS. (2010)
60. Ma, Y., Zheng, T., Cao, Y., Vrudhula, S., Seo, J.S.: Algorithm-hardware co-design of single shot detector for fast object detection on FPGAs. IEEE/ACM ICCAD. (2018)