Road Object Detection using Yolov3 and Kitti Dataset

Ghaith Al-refai[1]
School of Electrical and
Computer Engineering
Oakland University
Rochester, Michigan 48309

Mohammed Al-refai[2]
School of Computer Science and Information Technology
Jordan University of Science and Technology
Irbid, Jordan 22110

***Abstract*—Road objects (such as pedestrians and vehicles) detection is a very important step to enhance road safety and achieve autonomous driving. Many on-vehicle sensors, such as radars, lidars and ultrasonic sensors, are used to detect surrounding objects. However, cameras are widely used sensors for road objects detection for the rich information they provide and their inexpensive prices with compared to other sensors. Machine learning and computer vision algorithms are utilized to classify objects in the collected images and videos. There are many computer vision algorithms proposed for image and video object detection, e.g. logistic regression and SVM with feature extraction. However, Convolutional Neural Network (CNN) algorithms showed a high detection accuracy compared to other approaches. This research implements You Only Look Once (YOLO) algorithm that uses Draknet-53 CNN to detect four classes: pedestrians, vehicles, trucks and cyclists. The model is trained using Kitti images dataset which is collected from public roads using vehicle's front looking camera. The algorithm is tested, and detection results are presented.**

***Keywords*—Pedestrian detection; computer vision; CNN; machine learning; artificial intelligence; vehicle safety**

I. Introduction

Due to the increase in road accidents, the necessity of improving vehicle safety has increased in the past few years. According to world health organization, more than 1.35 million people die every year because of vehicle accidents [1]. Vehicle safety features started with passive safety approaches such as the three points seatbelts [2]. After that, active safety features introduced in vehicles, such as airbags, Anti-lock Braking System (ABS). At the beginning of the 20th century, Advanced Driver Assistance Systems (ADAS) introduced in vehicles. ADAS utilizes on-vehicle sensors to detect surroundings and notify drivers to avoid accidents. ADAS can provide the assistance to drivers as visualization, warning and control. Back up camera is an example of a visualization assist, where a rear camera sends video information to the driver display. Lane Departure Warning (LDW) is an example of warning feature, where an alarm is sent to the driver when vehicle is leaving the ego lane without turning the blinker on. Automatic Emergency Braking (AEB) is an example of a control feature, where an automatic braking command is sent automatically by the ADAS module if an object is detected in the front of the vehicle. Ziebinski et al. provided a review for the recent ADAS features in vehicles [3].

In order to implement the ADAS features, road sensing to detect surrounding objects is required. Pedestrians, cars, trucks and cyclist are examples of common road elements that needs to be detected. Ultrasonic sensors are commonly used to detect objects within short range distance from the vehicle. Radars, Lidar and cameras are used for long distance detection. Machine learning algorithms are used to analyze and understand the collected information by the sensors and classify objects in a scene. The improvement achieved in machine learning and computer vision algorithms in the past few years, and the low cost of cameras compared to radars and Lidars, made cameras a widely used sensor in vehicles for object detection and tracking to implement ADAS features.

Many computer vision algorithms were developed for road object detection. Template matching is the basic approach for object detection. In such approach, a template is used to describe an object, then objects in captured images are compared to the template to check if it matches. Shen and Steng introduced an algorithm for vehicle detection using template matching [4]. James et al. [5] presented a two-stage template-based method to detect people in thermal images. A review for template matching algorithms for object detection was provided in [6].

Feature extraction and logistic regression classification algorithms are other approaches for objects classification. Feature extraction step describes objects by their unique features, such as edges, textures and contours. Then a trained classifier is used to classify the objects. Dalal and Triggs used Histogram Oriented Gradient (HOG) and Support Vector Machine (SVM) for human detection [7]. The algorithm showed a near perfect result in MIT pedestrian database. Tsai et al. [8] proposed a vehicle detection algorithm that uses three features, including corners, edge maps, and coefficients of wavelet transforms, then a cascade multichannel classifier is used to classify the features. Li et al. [9] proposed a method based on adaboost classifier for Haar-like features and linear Discriminant Analysis (LDA) to detect the traffic signs.

A detailed review for pedestrian detection system was done by Gerónimo et al [10]. Sivaraman et al. [11] provided an overview review of the past decade's literature in on-road vision-based vehicle detection.

Convolutional Neural Network (CNN) have recently shown outstanding results in objects detection and classification. The huge computations and the large memory requirement for the CNN made it difficult to implement for real time detection

applications. However, the hardware improvement of the processors, GPUs and memory, in addition to parallel processing, made it possible to implement real-time CNN algorithms with low cost.

In this work, Darknet-53 CNN with YOLO algorithm is used for road object detection. The Kitti dataset is adopted to train and test the algorithm and its dataset. The algorithm possibly detects four objects: cars, trucks, pedestrians and cyclists. This paper provides a brief review for related works. Section 3 presents the algorithm implementation and presents detection results. Finally, Section 4 summarizes the conclusion of this research work.

II. Related Work

CNN is one of the most widely used machine learning technique in vision related application. Generally, CNN includes the following processes: convolution, pooling, activation functions and fully connected layers. The convolution works as a feature extractor, while pooling is used for down sampling. The activation functions add nonlinearity to the algorithm to handle complex features. The fully connected layers are used to classify the features. A training dataset is required to train the CNN. Back propagation technique is used to train the algorithm to reduce the classification error. Shridhar provided a guidance for deep learning algorithms and applications [12].

CNN architecture can vary by varying the number of layers, neurons, activation functions and the order of computation elements. The architecture varies based on the application and the available training data. There are many architectures have been introduced in the past few years. Alex et al. introduced Alexnet [13]. This neural network has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully connected layers with a final 1000-way softmax. VGG-16 network introduced with 138 million parameters and more layer depth than Alexnet [14]. Googlenet and Resnet are also examples of CNN architectures [15], [16]. Khan et al. [17] provided a survey of the recent architectures of deep convolutional neural networks.

The captured images by vehicle's front camera includes many objects. In autonomous driving and ADAS applications, Objects like vehicles and pedestrians are the objects of interest. Other objects are considered as background and shouldn't be detected, such as buildings and trees. There are many regional CNN techniques developed to classify the objects of interest in images.

The basic approach for selective search CNN is to scan the whole image by bounding boxes and pass it to the CNN to classify them. The disadvantage of this approach is the large number of bounding boxes to be classified, which makes it a challenge for real time implementation and increase the chances for false positives. Ross et al. [18] proposed a method that uses selective search to extract just 2000 regions from the image and they are called region proposals. Therefore, instead of trying to classify a huge number of regions, the CNN just work with 2000 regions. These 2000 region proposals are generated using the selective search algorithm. This algorithm is called regional-CNN. Fig. 1 shows the system overview of R-CNN as proposed in [18]. 2000 candidates are still a huge number of region proposals for real time application.

Ross [19] proposed a newer version of R-CNN to improve the algorithm processing time, he called the algorithm Fast R-CNN. This approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, the whole image is fed to the CNN to generate a convolutional feature map. From the convolutional feature map, it identifies the region of proposals and warp them into squares. Fast R-CNN improved algorithm runtime compared to R-CNN. Even though, the algorithm is still time consuming and slow.

Shaoqing Ren et al. [20] came up with an object detection algorithm that eliminates the selective search algorithm and lets the network learn the region proposals In this approach Regional Proposal Network (RPN) is used to generate region proposals, then, these proposals are fed to CNN for object classification. Fig. 2 shows the block diagram of the Faster RCNN block diagram. Mask R-CNN is an improved version of Faster R-CNN that reduces the processing time of the algorithm [21].

All the previous mentioned algorithms use regions to locate objects in images, the algorithm doesn't look to the complete image. You Only Look Once (YOLO) algorithm looks to the complete image one time by passing the whole image to the CNN. YOLO showed a massive improvement in reducing the algorithm computation and it is faster than the other regional CNN. The next section explains YOLO architecture and our detection system implementation.

III. YOLO for Object Detection

Yolo algorithm was proposed for the first time in 2016 by Redmon et al. [22]. Then a second version was proposed by the same author with better, faster and stronger performance [23]. Redmon and Ali proposed the third and the latest version of Yolo, and they called Yolov3 [24]. In this research, Yolov3 was used to implement road object detection system.

Yolov3 network proposed a CNN with 53 layers, so the network is called Dearknet-53. Fig. 3 shows the architecture of Darknet-53. The convolutional layers are followed by batch normalization layer and Leaky ReLU activation. No form of pooling is used, and a convolutional layer with stride 2 is used to down sample the feature maps.

Yolo algorithm works as following:

- The input image is divided to SxS cells

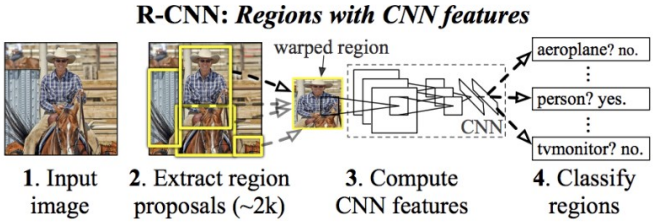

Fig. 1. R-CNN System Overview [18].

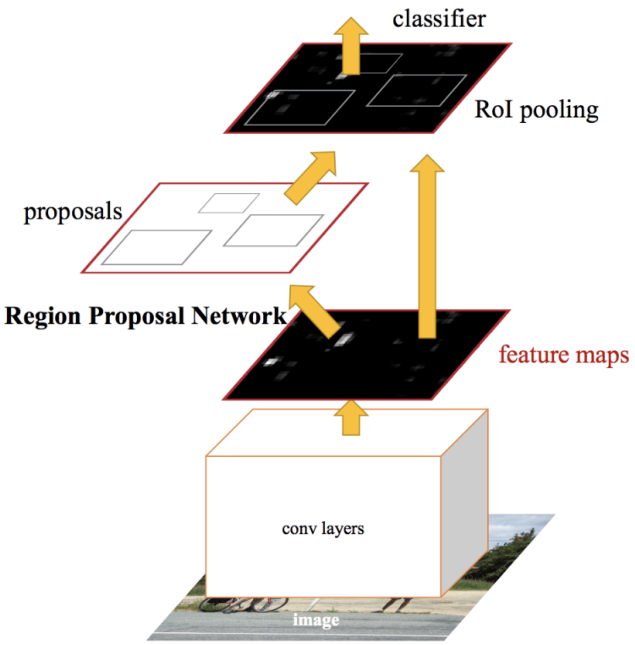

Fig. 2. Faster R-CNN System Block Diagram [20]

- B bounding boxes are generated from each cell
- Each bounding box is described by a vector, the
- vector includes Tx, Ty, Th, Tw, Pc, C

Where:
Tx, Ty: the top left corner of the bounding box coordinates
Th, Tw: the height and the width of the bounding box
Pc: Object score, between 0 and 1
C: class scores vector

In our implementation, the input image is resized to 416x416x3. And divided to 13x13 cells, each cell is 32x32 pixels. 9 bounding boxes are generated from each box. The scoring of each box takes a value from 0 to 1. It represents the probability that the bounding box includes an object. C is 4x1 vector, and it represents the prediction of a bounding box as one of the following classes: pedestrian, truck, car or cyclist. The sum of the prediction should add to 1.

Each cell produces a vector with size of B*(5+C). In our implementation the size of the vector is equal to 9*(5+4) = 81. And the output of the CNN per image equals to 13*13*81 = 13689.

For example, if a bounding is given by the following vector: [5, 6, 50, 70, 0.5, 0.05, 0.7, 0.05, 0.2], it means the top left corner of the bounding box is located at (5,6) with height of 50 pixels and width of 70 pixels. The scoring of the object is 0.5, which means the bounding box includes an object with probability of 0.5. The last four elements represent the object class, it shows that the object classified as pedestrian, truck, car and cyclist with probabilities of 0.05, 0.7, 0.05 and 0.2, respectively.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Fig. 3. Darknet-53 Network Architecture [24]

TABLE I. YOLOV3 IMPLEMENTATION PARAMETERS

Parameters	Value
Input image size	0.005
Input image size	416X416X3
Number of cells per image	13x13
Number of bounding boxes per cell	9
Classes	[Pedestrian, Truck, Car, Cyclist]
Classification threshold	0.5
Non-Maximum suppression overlapping threshold	0.5

The class confidence is given by the product of the object score and the maximum element of vector C. In the previous example, the class confidence equals 0.5*0.7 = 0.35. The classification of object is done based on a fixed threshold value. The non-maximum suppression algorithm is used to eliminate the overlapping detection boxes for the same object. Table I summarizes the parameters of Yolo implementation in our algorithm.

IV. Kitti Dataset and Algorithm Training

A labeled dataset is needed to train the Yolov3 algorithm. There are many datasets available online that can be used for training and testing such as COCO dataset [25], Pascal VOC dataset [26] and google open images V5 dataset [27]. However, Kitti dataset is collected by a vehicle equipped with dash camera and other sensors for autonomous driving testing and benchmarking [28]. This makes it a perfect fit for our detection system as we aim to detect road objects. Kitti dataset consists of 7481 training images with seven labeled classes: cars, van, tram, trucks, pedestrian, person sitting and cyclist. Fig. 4 shows some samples of Kitti dataset.

In our implementation, the dataset labels were reorganized to fit in our four classes. Van, tram and cars are considered as one class, it is called cars. Sitting person and pedestrian are merged, the class named as pedestrian. Trucks and cyclists taken from the dataset without any change.

Kitti dataset labeling format is incompatible with Yolov3. Labels were formatted to match Yolov3 algorithm. The label should be constructed as following: [P1, P2, P3, P4, Class], where
(P1, P2) the top left corner of the bounding box
(P3, P4) the bottom right corner of the bounding box
Class: the bounding box class, it takes value between 0 to 3. It means Pedestrian, Truck, Car, Cyclist, respectively.

The pre-trained weights of Yolov3 with COCO dataset were uploaded as initial weights for the model. The model was trained in two steps. The first step is the frozen training, where the initial layer weights were frozen, and the end layer weights were trained. The epoch is set to 25 and the batch to 32. The second step was the unfrozen training for fine tuning. The epoch is set to 25 and the batch to 4. Adam optimizer is used in the training model [29]. Fig. 5 shows the training loss with epoch value.

V. Detection Results

The algorithm is tested using 300 images from Kitti test dataset. The detection was analyzed by counting the true positives, true negatives and false positives. The scoring threshold for classification was set 0.5, and the non-maximum suppression for the detection boxes overlapping is set 0.5. Fig. 6 shows some samples of the detection in our implementation.

Fig. 4. Image Samples from Kitti Training Dataset

Two concerns can be made about Kitti test dataset. The first concern is the test images are not labeled, so detection results can't be automated and have to be done manually. The second concern is cars are the dominant object in the dataset and the dataset doesn't have enough samples for pedestrians, cyclists and trucks.

Table II summarizes the detection results for all the classes. Results show that 615 objects are classified correctly, while 111 objects were miss classified and 88 objects were false positives. It is noticed that false negative detection was high for pedestrians and cyclists. For better understanding of the results, the precision and recall were calculated for each class. Precision and recall are given in the following equations:

$$Precision = Truepositives/(Truepositives+Falsepositives) \quad (1)$$

$$Recall = Truepositives/(Truepositives+Falsenegatives) \quad (2)$$

Table III shows the precision and recall for each class. The precision is higher than 98% for all the classes except for cyclists, it is 88.23%. So, we can conclude that detection accuracy is very high for all the classes. There is a significant drop in the recall values for pedestrians and cyclists because of the high values for the false negatives. Due to the small

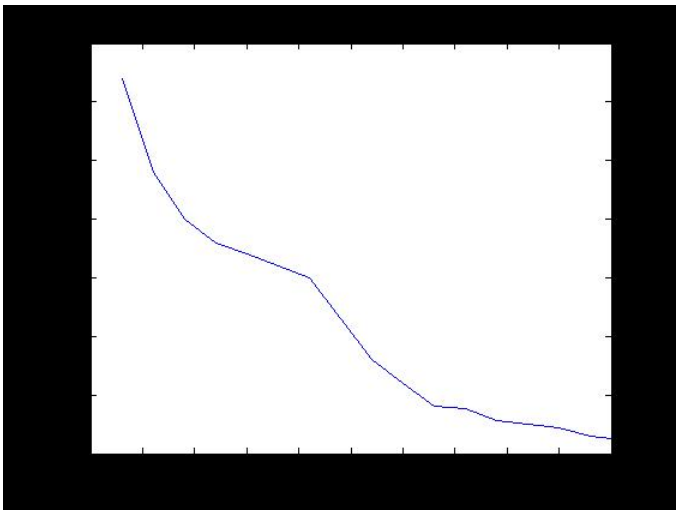

Fig. 5. Darknet-53 Training Loss with Epoch using Kitti Dataset

TABLE II. Detection results for Kitti dataset using Yolov3

Objects	True Positives	False Positives	False Negatives
Cars	559	6	72
Trucks	3	0	3
Pedestrians	38	0	32
Cyclists	15	2	4
Total	615	8	111

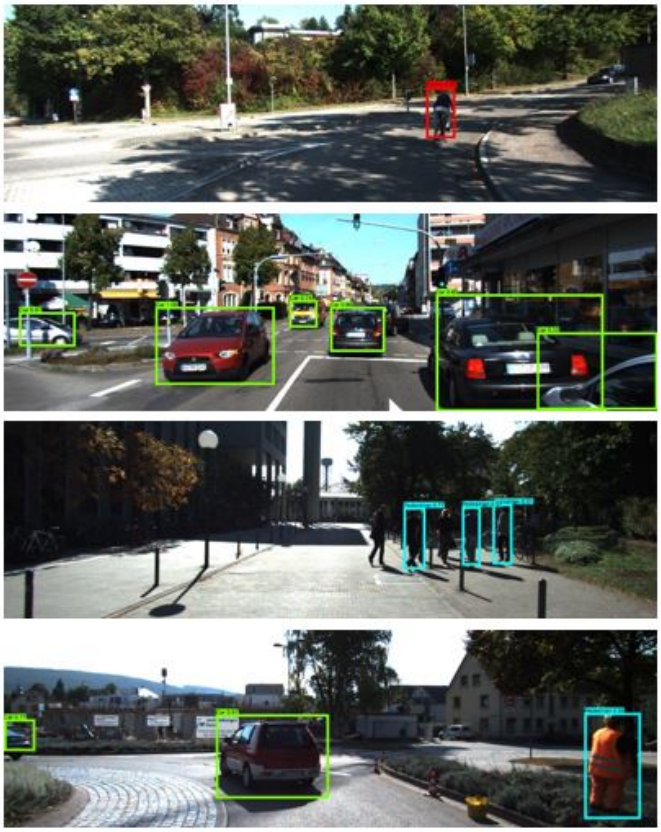

Fig. 6. Detection Results Image Samples

TABLE III. PRECISION AND RECALL VALUES FOR THE CLASSES

Objects	Precision	Recall
Cars	98.9%	88.59%
Trucks	100%	50%
Pedestrians	100%	54.28%
Cyclists	88.23%	78.9%
Total	98.7%	84.7%

samples of trucks available in the test dataset, we can't judge the algorithm performance in trucks detection.

The summary we can make from the above results is that the algorithm showed very excellent detection accuracy for cars. It also showed a high precision for pedestrians and cyclists but with low recall values due to the high number of false negatives. That means the algorithm has high miss detection rate for small objects like pedestrians and cyclists compared to larger objects like cars.

VI. CONCLUSION

Convolutional neural networks are a widely used algorithm for object detection and classifications. This research implemented Yolov3 algorithm which uses Darknet-53 CNN for road object detection. Cars, trucks, pedestrians and cyclists are the algorithm detection classes. Kitti dataset is used for the algorithm training and testing.

The paper provided a quick review for the different approaches for road object detection and presented the recent CNN architectures and methods for object detection. The Darknet-53 architecture of Yolov3 is explained and the algorithm hyper-parameters were discussed. Algorithm training steps and parameters like epoch and batch were presented.

Detection results are displayed by counting true positives, false positives and false negatives per each class. Precision and recall also calculated for each class. The algorithm showed a very good detection accuracy for cars. Pedestrian and cyclist detection showed more false negatives than cars. The test dataset includes only six trucks; therefore, no solid conclusion can be made about truck detection. A conclusion we can make from the detection results is that the algorithm produces more false negatives in small objects detection.

A suggested solution to improve detection accuracy for pedestrians and cyclists is to reduce the cell size. In our implementation the cell size is 32x32 pixels. Smaller cells should improve detection for the small objects, in the other hand, it increases the processing time of the algorithm. Another solution to reduce the false negatives is to increase the epoch from 25 to a higher number in the training.

Kitti dataset have enough trained labeled images to achieve good performance. However, cars are the dominant object in the dataset, and the test images are not labeled. Which makes detection result analysis more difficult. There are other open source datasets that can be used in road object detection such as Waymo open dataset and Bekerly deep drive.

REFERENCES

[1] World Health Organization (WHO). "Global status report on road safety 2018 (2018)." Geneva (Switzerland): WHO (2019).

[2] Ivar, Bohlin Nils. "Safety belt." U.S. Patent 3,043,625, issued July 10, 1962.

[3] Ziebinski, Adam, Rafal Cupek, Damian Grzechca, and Lukas Chruszczyk. "Review of advanced driver assistance systems (ADAS)." In AIP Conference Proceedings, vol. 1906, no. 1, p. 120002. AIP Publishing LLC, 2017.

[4] Shen, X. L., and D. C. Tseng. "Vision detection of lanes and vehicles for advanced safety vehicles." National Central University, Chung-li (2007).

[5] Davis, James W., and Mark A. Keck. "A two-stage template approach to person detection in thermal imagery." In 2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05)-Volume 1, vol. 1, pp. 364-369. IEEE, 2005.

[6] Nath, Rajiv Kumar, and Swapan Kumar Deb. "On road vehicle/object detection and tracking using template." Indian Journal of Computer Science and Engineering 1, no. 2 (2010): 98-107.

[7] Dalal, Navneet, and Bill Triggs. "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.

[8] Tsai, Luo-Wei, Jun-Wei Hsieh, and Kuo-Chin Fan. "Vehicle detection using normalized color and edge map." IEEE transactions on Image Processing 16, no. 3 (2007): 850-864.

[9] Li, Zhijiang, Chuan Dong, Ling Zheng, and Long Liu. "Traffic signs detection based on haar-like features and adaboost classifier." In ICTIS 2013: Improving Multimodal Transportation Systems-Information, Safety, and Integration, pp. 1128-1135. 2013.

[10] Gerónimo, D. and López, A.M., 2014. Vision-based pedestrian protection systems for intelligent vehicles (pp. 1-114). New York, NY, USA: springer.

[11] Sivaraman, Sayanan, and Mohan M. Trivedi. "A review of recent developments in vision-based vehicle detection." In 2013 IEEE Intelligent Vehicles Symposium (IV), pp. 310-315. IEEE, 2013.

[12] Shridhar, A. "A beginner's guide to deep learning." (2017).

[13] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." In Advances in neural information processing systems, pp. 1097-1105. 2012.

[14] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[15] Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9. 2015.

[16] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.

[17] Khan, Asifullah, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. "A survey of the recent architectures of deep convolutional neural networks." arXiv preprint arXiv:1901.06032 (2019).

[18] Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "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.

[19] Girshick, Ross. "Fast r-cnn." In Proceedings of the IEEE international conference on computer vision, pp. 1440-1448. 2015.

[20] Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks." In Advances in neural information processing systems, pp. 91-99. 2015.

[21] He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask r-cnn." In Proceedings of the IEEE international conference on computer vision, pp. 2961-2969. 2017.

[22] Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "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.

[23] Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7263-7271. 2017.

[24] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[25] Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft coco: Common objects in context." In European conference on computer vision, pp. 740-755. Springer, Cham, 2014.

[26] Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." International journal of computer vision 88, no. 2 (2010): 303-338.

[27] Open images dataset V5, www.storage.googleapis.com/ openimages/web/visualizer/index.html?set=train&type=detection&c=%2Fm%2F01j51

[28] Geiger, Andreas, Philip Lenz, Christoph Stiller, and Raquel Urtasun. "Vision meets robotics: The kitti dataset." The International Journal of Robotics Research 32, no. 11 (2013): 1231-1237.

[29] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).