Volume: 13 Issue: III Month of publication: March 2025
DOI: https://doi.org/10.22214/ijraset.2025.67672

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue III Mar 2025- Available at www.ijraset.com

Autonomous Navigation Using Deep Learning

Palak Agarwal[1], Somya Goel[2], Simran Bhagat[3], Asst. Prof. Rahul Singh[4]
Department of Computer Science & Information Technology Meerut Institute of Engineering and Technology Meerut, India

Abstract: With uses in robotics, industrial automation, autonomous vehicles, and surveillance, object detection is a basic computer vision problem. Within the context of the COCO dataset, this work compares the performance of several state-of-the-art object recognition models, including Mask R-CNN (Detectron2), YOLOv8s, YOLOv8l, and YOLOv11s. Some of the significant parameters such as mean Average Precision (mAP), precision, recall, and inference speed are utilized to compare models.
The results indicate that while Mask R-CNN is accurate, its computation makes it less suitable for real-time use. YOLO models, particularly YOLOv8s, are however a compromise between accuracy and speed and thus are ideal for real-time detection processes. YOLOv8l is however computationally more demanding but somewhat offers higher accuracy. Due to its speed and accuracy, YOLOv8s is the most suitable model to apply in real-time, as stated in the review. In selecting the most suitable object detection models for various applications, researchers and developers can learn a lot from this study.
Keywords: YOLO , DETECTRON , R-CNN , Object Detection.

I. INTRODUCTION

Object detection is one of the pillars of computer vision that allows machines to detect and locate objects within an image or video. It is used in a wide variety of applications in robotics, industrial automation, medical imaging, autonomous cars, and security. Several object detection models based on deep learning have been proposed in recent years to make it faster and more accurate. While deciding on the most suitable model for real-time object detection, we are considering Mask R-CNN (Detectron2), YOLOv8s, YOLOv8l, and YOLOv11s based on their performances.
Prior object detection methods utilized region-based methodologies, like Faster R-CNN, which were computationally intensive but very precise. YOLO algorithms have, nonetheless, transformed object detection by remarkably optimizing speed with no compromise in accuracy. While Detectron2, which is an implementation of Mask R-CNN, is very efficient in segmenting objects precisely, its high computational cost renders it inappropriate for real-time use.
Based on the COCO dataset, herein, various models are compared on the basis of testing of remarkable performance measures like precision, recall, inference time, and mean Average Precision (mAP). Determination of the most balanced model regarding speed and accuracy for various applications is the focus. This study will guide developers and researchers in choosing the most suitable model for object detection applications.

II. LITERATURE REVIEW

Deep model architecture, particularly that of YOLO-type models, Faster R-CNN, and SSD, has revolutionized object detection. Some other approaches with a focus on segmentation and tracking included the unsupervised Siam Mask model, which ranked top in self-driving object segmentation and tracking experiments [1].
Real-time processes have heavily depended on Detectron2, which is a mature object detection model based on Faster R-CNN. Detectron2 employs region proposal networks to achieve higher accuracy and detects traffic entities in real-time through Faster R-CNN efficiently, according to research [2].
Tinier-YOLO offers a very good solution for real-time processing by reducing computational complexity, and optimizations for constrained environments have also been realized in YOLO models [3]. With negligible real-time processing rate compromise, YOLO v3-Tiny optimizations significantly improved detection accuracy [4].
YOLO model family have been focused upon heavily by object detection researches. YOLO was initially presented in earlier work as a unified real-time object detection method able to detect a high number of variant objects [5][6]. The model's speed was enhanced while keeping accuracy, incorporating hierarchical detection with YOLO9000 [7]. YOLOv3 continued to advance in order to enhance feature extraction as well as its ability to detect small objects [8].
There has been a great amount of research also that has been done on object detection methods region-based. R-CNN established the groundwork for the next generation of models with dense feature hierarchies allowing object segmentation and identification [9].

With the addition of region proposal networks, Faster R-CNN added a great amount of speed up without any decrement in accuracy [10]. In SSD, yet another widely employed class, was illustrated a one-shot detection architecture which efficiently balanced between speed and accuracy [11].
CNNs and residual learning models are also the key to further promoting object detection. The success of deep learning in image classification problems was established with the advent of convolutional neural networks (CNNs) in ImageNet classification [12]. Residual learning techniques enabled networks that were much deeper without gradient vanishing, further promoting feature extraction [13]. To enhance dense object detection models and enhance accuracy in the case of class imbalance, focal loss algorithms have also been developed [14]. Moreover, the inclusion of OpenCV in computer vision software has made it easier to implement object detection frameworks [15].

III. METHODOLOGY

A. *About the Dataset*

During this research, the Common Objects in Context (COCO) dataset, being the most popular dataset used for object detection, image segmentation, and image captioning, is utilized. COCO is the world largest object detection dataset with more than 200,000 labeled images and 80 object classes. Dense annotations of the dataset as bounding boxes, instance masks, and key points enable successful deep learning model training and assessment. For convenience, we particularly use the COCO validation set made available on Kaggle to try out part of our object detection models used in the project. Because objects in the dataset vary in size (small, medium, large), we are thus in a position to compare the performance of various models as a function of object size variability. It is ideal for testing the generalization capacity of existing YOLO and Mask R-CNN models because it is challenging, with occluded objects and challenging backgrounds. We want to carry out an unbiased and balanced assessment of various detection models in COCO.

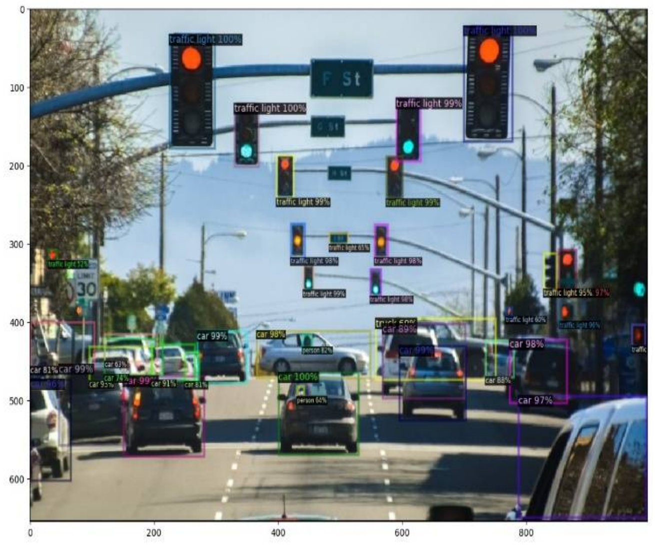


Figure 1. Objects Detected image from dataset

B. *Overview of Object Detection FrameWorks*

1) *Detectron 2*

Facebook AI Research (FAIR) came up with the cutting-edge object detection architecture known as Detectron2. Its extensible and modular building blocks for the object detection and segmentation tasks are based on PyTorch. Detectron2 supports high-performance detection of objects with a wide range of models including Faster R-CNN, Mask R-CNN, and RetinaNet. It also features Region Proposal Networks (RPN) and Feature Pyramid Networks (FPN) to provide improved accuracy. It is simpler to employ on real-world problems using the pre-trained models that have been trained across the COCO dataset. Due to its strong GPU acceleration optimization, Detectron2 offers training and inference efficiency. People can train models for certain tasks as well because to its capacity to handle bespoke datasets. COCO metrics for performance evaluation are some of the tools within the framework. Detectron2 is widely used by video analysis, medical imaging, and autonomous driving. It is accessible to researchers and developers since it has a straightforward API.

2) Yolo

The cutting-edge object detection model YOLO (You Only Look Once) is renowned for its real-time speed and accuracy. By formulating object detection as a one-stage regression task, YOLO both predicts class probabilities and bounding boxes from the input image at once, as opposed to traditional region-based detectors. This does not entail large processing overhead, thereby making it extremely fast for real-time applications like robots, autonomous vehicles, and surveillance. In order to provide real-time and effective localization, YOLO utilizes a deep convolutional network that splits an image into a grid and allocates detection work to all the cells in the grid. Since the model has undergone several revisions from YOLOv1 to YOLOv8, each of them has improved precision, velocity, and potency. To provide improved feature extraction as well as generalization, new releases include transformer-based advancements and leading-edge topologies like CSPDarknet. Additionally, YOLO also accommodates multi-scale detection, which enables it to detect differently sized objects.

It will be capable of generalizing for many situations since it was trained on large datasets such as COCO.

YOLOv8-nano and YOLOv8-small are reduced size models that have been optimized for deployment on processing-poor devices. YOLO continues to be a leading object detection framework used in research and also applied in commercial deployments because of its ability to strike a balance between accuracy and speed.

C. Difference Between Detectron2 and Yolo

While both Detectron2 and YOLO are robust object detection models, there are a few differences in application, structure, and strategy between the two models as well. While the Faster R-CNN and Mask R-CNN models were designed to do a few things, Facebook AI developed Detectron2 to detect objects accurately and segment instances. It is expensive computationally but highly precise because it uses a two-stage detection model by first generating regions of interest and refining them subsequently. YOLO is one-stage detection, and this is highly efficient and quick real-time runner as it predicts bounding boxes and class probability simultaneously. As YOLO is efficiency-oriented, it is utilized where inference needs to be executed as quickly as possible, i.e., self-driving vehicles and surveillance cameras. Slower but more efficient at executing tasks such as panoptic segmentation and occluded objects is Detectron2. While Detectron2 is more computationally intensive, YOLO models such as YOLOv8 are light and can be run on edge devices. While Detectron2 is ideal to deeper levels in order to enable deeper learning and investigation, YOLO is ideal for a beginner. It is simple to comprehend and is not complicated. The decision then is whether one would prefer detection and segmentation with high precision (Detectron2) or in real-time (YOLO).

IV. ARCHITECTURE OF MODELS

A. YOLOv8l

The head-neck-backbone architecture of YOLOv8l (Large) is optimized for accuracy over efficiency. The backbone utilizes CSPDarknet, gradient flow-friendly, with reduced redundancy, and feature extraction improved through cross-stage partial connections. PANet (Path Aggregation Network) is employed within the neck for multi-scale feature fusion with strong object detection irrespective of sizes. The detecting head is anchor-free, and this improves the localization accuracy and makes bounding box regression easier. Decoupled head architecture is employed for higher accuracy, wherein it decouples the task of localization from the task of categorization. Depthwise separable convolutions are utilized in the model to avoid compromising accuracy with an increase in computational burden. YOLOv8l achieved better results compared to the lower models in project evaluation with precision of 0.810, recall of 0.690, mAP50 of 0.775, and mAP50-95 of 0.610. A few of the reasons for the errors are false positives on thick backgrounds and lack of capability to identify objects that are little in occluded scenes. YOLOv8l is perfect for high-performance real-time detection because it neither loses speed nor accuracy despite these constraints.

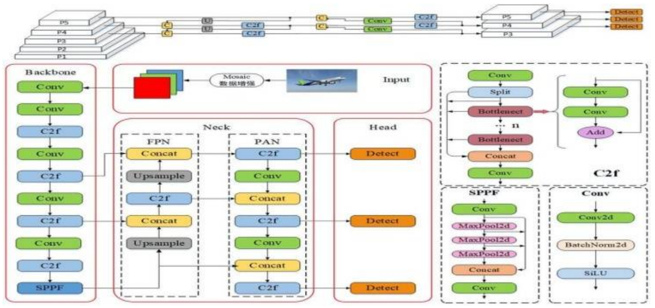


Figure 2 YOLOv8 Models

B. YOLO11s

YOLO11s, a pricier variant of the YOLO series, supports real-time object detection with increased speed and accuracy. For extracting features at reduced computational expenses, it follows a backbone-neck-head structure with a CSPDarknet-based backbone. Small and large object detection is enhanced through additional fine-grained multi-scale fusion of features by an enhanced Path Aggregation Network (PANet). Since the anchor-free structure of the detection head contains no special anchor boxes, the prediction flexibility is enhanced. YOLO11s applies depthwise separable convolutions in an attempt to enhance the computing efficiency without compromising the detection accuracy. Even more sophisticated than any other YOLO model, YOLO11s has been designed to detect objects in real-time with increased speed and accuracy.For easier computational feature extraction, it employs a backbone-neck-head architecture with a CSPDarknet backbone. The neck encourages small and large object detection through multi-scale feature fusion optimization via an improved Path Aggregation Network (PANet). Due to the anchor-free nature of the detection head, some of the anchor boxes are not necessary, and prediction is more flexible. Depthwise separable convolutions are employed by YOLO11s to obtain computation efficiency without loss of detection accuracy.

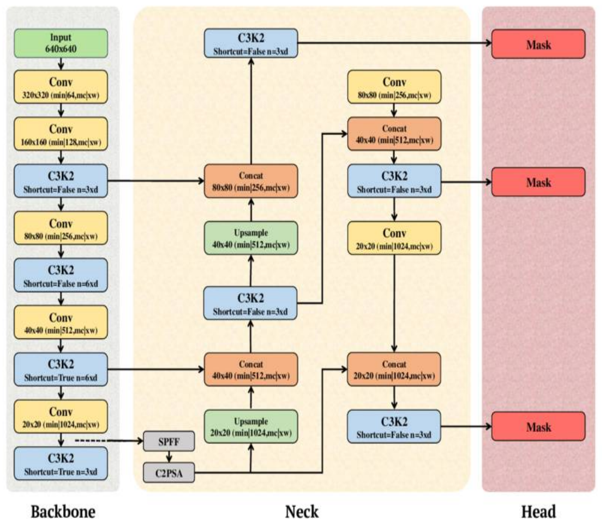

Input 640x640
Conv 320x320 (min|64,mc|xw)
Conv 160x160 (min|128,mc|xw)
C3K2 Shortcut=False n=3xd
Conv 80x80 (min|256,mc|xw)
C3K2 Shortcut=False n=6xd
Conv 40x40 (min|512,mc|xw)
C3K2 Shortcut=True n=6xd
Conv 20x20 (min|1024,mc|xw)
C3K2 Shortcut=True n=3xd
SPFF
C2PSA
C3K2 Shortcut=False n=3xd
Concat 80x80 (min|256,mc|xw)
Upsample 40x40 (min|512,mc|xw)
C3K2 Shortcut=False n=3xd
Concat 40x40 (min|512,mc|xw)
Upsample 20x20 (min|1024,mc|xw)
Conv 80x80 (min|256,mc|xw)
Concat 40x40 (min|512,mc|xw)
C3K2 Shortcut=False n=3xd
Conv 20x20 (min|1024,mc|xw)
Concat 20x20 (min|1024,mc|xw)
C3K2 Shortcut=False n=3xd
Mask
Mask
Mask
Backbone
Neck
Head

Figure 3 YOLOv11s Models

C. DETECTRON2 (R-CNN)

Employing a Feature Pyramid Network (FPN) multi-scale feature extraction, Detectron2 model Mask R-CNN R_50_FPN_3x is a two-stage object segmentation and detection model with a ResNet-50 backbone. A Region Proposal Network (RPN) generates candidate object regions during the first stage. During the second stage, the proposals are enhanced, objects class-annotated, and bounding boxes predicted in addition to instance segmentation by an independent mask head. By fusing low-level and high-level features, FPN encourages feature learning and enhances detection precision on various object sizes. A longer training duration is denoted by the "3x" in the model name name, which enhances precision and convergence. This model performed well at most IoU thresholds with Average Precision (AP) of 0.375 on COCO 2017 validation set, AP50 of 0.546, and AP75 of 0.419. With an Average Recall (AR) of 0.445, the measures of recall are reflecting the error rate and include some missed detections, i.e., on small objects (AR small = 0.232). The model is computationally expensive and needs a GPU to run in real-time inference but has very good accuracy.

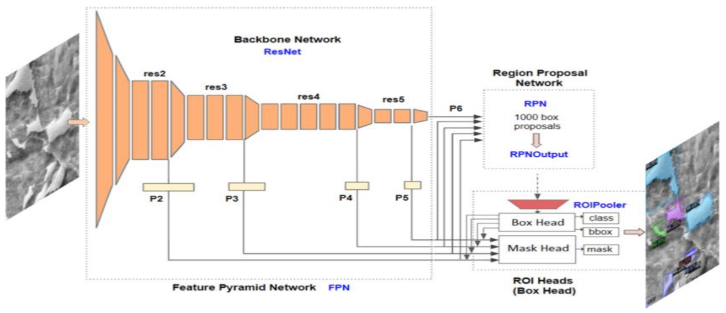


Figure 4 Detectron2 Models

D. YOLO8s

The YOLOv8s architecture is a lightweight and efficient object detection model optimized for real-time applications. It utilizes a CSPDarknet backbone for feature extraction, a PANet (Path Aggregation Network) for feature fusion, and a YOLO detection head for final predictions. The model incorporates decoupled heads to improve accuracy in classification and localization tasks. It processes images in a single pass, making it significantly faster than two-stage detectors like Faster R-CNN. In our project, YOLOv8s achieved a precision of 78.42%, a recall of 67.05%, an mAP50 of 76.01%, and an mAP50-95 of 58.79% on the COCO validation dataset. These results indicate that YOLOv8s provides a good balance between speed and accuracy, outperforming older YOLO versions while being computationally efficient. However, its performance is slightly lower than larger models like YOLOv8l but remains more suitable for deployment on edge devices. The model also demonstrated improved inference speed compared to Detectron2, making it ideal for real-time detection applications. Its lightweight nature and efficient architecture make it a strong choice for tasks requiring both speed and accuracy in object detection.

V. RESULTS OBTAINED

We In the present case, we are trying to compare how well or efficiently some object detection models are on the COCO dataset, for example, Mask R-CNN (Detectron2), YOLOv8s, YOLOv8l, and YOLOv11s.

Average Precision (AP), Average Recall (AR), precision, and inference speed are a few of the used metrics.

With AP of 0.375, AP50 of 0.546, and AP75 of 0.419, Mask R-CNN (Detectron2) was extremely accurate in detection with relatively higher computational expense. Recall (AR = 0.445) reflects the limitation of detecting small objects efficiently.

With the decreased inference time of 4.95ms, YOLOv8s achieved an mAP50 of 0.760 and an mAP75-95 of 0.587 and outperformed Mask R-CNN in speed and accuracy but marginally.

Higher recall and accuracy were achieved by YOLOv8l, which was bigger in size but with greater computational requirements.

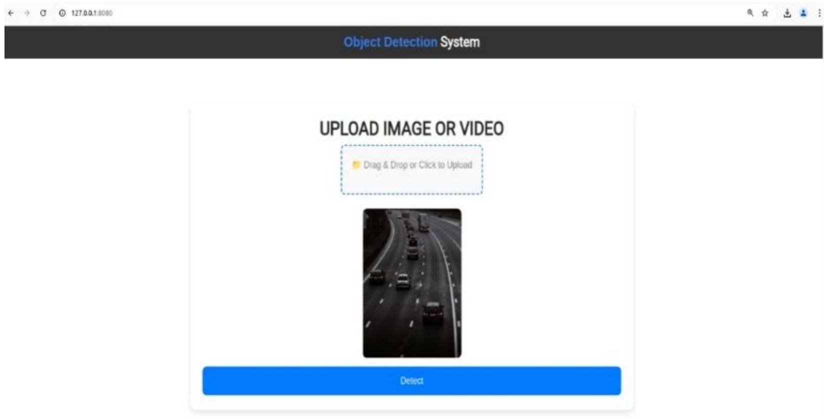


Figure 4 Raw image

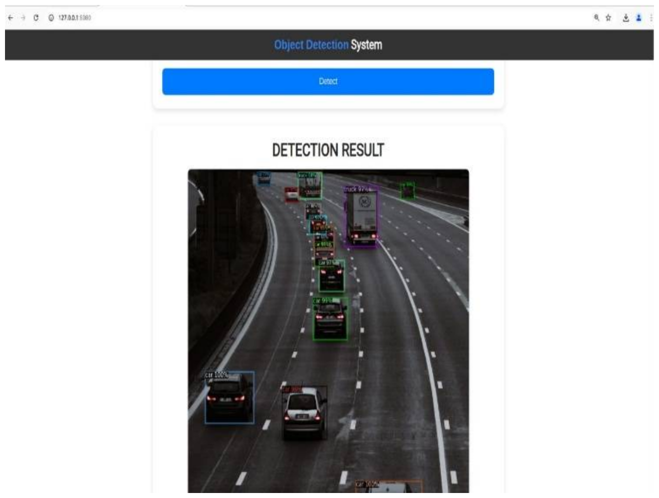
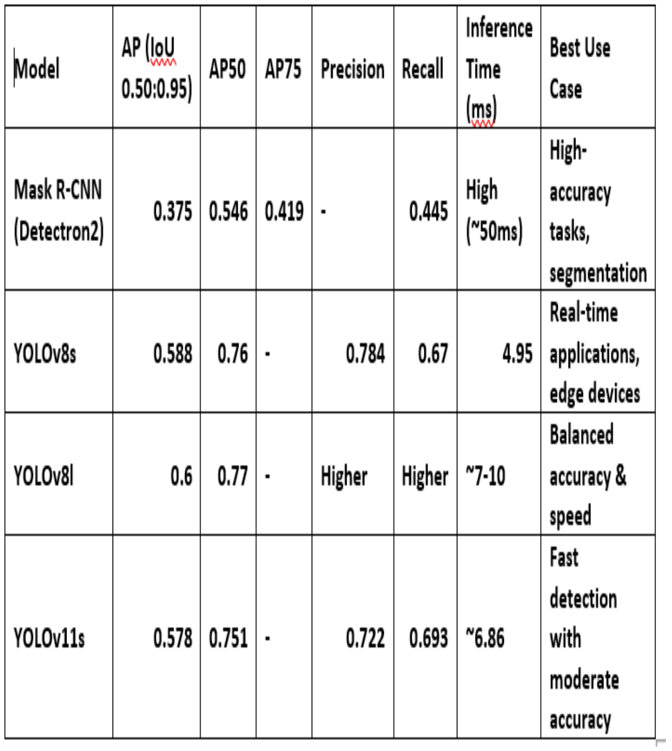


Figure 5 Object Detected Image

YOLOv11s achieved the same performance of YOLOv8s at mAP50-95 of 0.578 but with the ability to perform better under some detection conditions.

YOLO models offer a better trade-off between real-time speed and accuracy and are therefore suitable for real-time tasks such as autonomous driving and video tracking, but Mask R-CNN is good at challenging segmentation. The relative importance of speed, accuracy, and computational expense required by an operation determines the model to be used.



Model	AP (IoU 0.50:0.95)	AP50	AP75	Precision	Recall	Inference Time (ms)	Best Use Case
Mask R-CNN (Detectron2)	0.375	0.546	0.419	-	0.445	High (~50ms)	High-accuracy tasks, segmentation
YOLOv8s	0.588	0.76	-	0.784	0.67	4.95	Real-time applications, edge devices
YOLOv8l	0.6	0.77	-	Higher	Higher	~7-10	Balanced accuracy & speed
YOLOv11s	0.578	0.751	-	0.722	0.693	~6.86	Fast detection with moderate accuracy

Table 1Comparison table of Object Detection Models

VI. CONCLUSION

Herein, we have experimented and compared various object detection models such as Mask R-CNN (Detectron2), YOLOv8s, YOLOv8l, and YOLOv11s on accuracy measures (AP, mAP), precision, recall, and inference time with the COCO dataset. Our results indicate that although Mask R-CNN is appropriate for instance segmentation and has very high accuracy (AP = 0.375, AP50 = 0.546), it consumes a lot of resources and thus is not very appropriate for real-time use. Conversely, YOLO models such as YOLOv8s and YOLOv8l performed better at an mAP50 of 0.760 and 0.770, respectively, but with much lower inference times (4.95ms for YOLOv8s), but YOLOv8l was better than YOLOv8s with increased computations.

While YOLOv11 models have an mAP50-95 value of 0.578, they were not significantly better compared to YOLOv8 models.

The top real-time object detection model is YOLOv8s when inference speed, accuracy, and computational efficiency trade-offs are considered. It can be used for real-time tracking, surveillance, and autonomous use due to its accuracy-speed ratio. Model optimization and hybrid methods to improve detection efficiency can be explored in future research.

REFERENCES

[1] S. Noor, M. Waqas, M. I. Saleem, and H. N. Minhas, “Automatic Object Tracking and Segmentation Using Unsupervised SiamMask,” IEEE Access, vol. 9, pp. 106550-106559, 2021.

[2] A. Obi-Obuoha, V. S. Rizama, I. Okafor, H. E. Ovwenkekpere, K. Obe, and J. Ekundayo, “Real-time traffic object detection using detectron 2 with faster R-CNN,” World Journal of Advanced Research and Reviews, vol. 24, no. 02, pp. 2173-2189, 2024.

[3] W. Fang, L. Wang, and P. Ren, “Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments,” IEEE Access, vol. 8, pp. 1935-1944, 2020.

[4] P. Adarsh, P. Rathi, and M. Kumar, “YOLO v3-Tiny: Object Detection and Recognition using one stage improved model,” in 2020 6th International Conference on Advanced Computing & Communication Systems (ICACCS), 2020, pp. 687-691.

[5] C. Liu, Y. Tao, J. Liang, K. Li, and Y. Chen, “Object Detection Based on YOLO Network,” in 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC 2018), 2018, pp. 50-54.

[6] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” arXiv:1506.02640, 2015.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” arXiv:1612.08242v1, 2016.
[8] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” arXiv:1804.02767, 2018.
[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for object detection and semantic segmentation,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580-587.
[10] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 2017.
[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, and A. C. Berg, “SSD: Single shot multibox detector,” in European conference on computer vision, 2016, pp. 21-37.
[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” Advances in neural information processing systems, vol. 25, 2012.
[13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770-778. [cite: 1011, 1012, 13, 14, 15, 16, 45] .
[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980-2988.
[15] G. Bradski and A. Kaehler, “Learning OpenCV: Computer vision with the OpenCV library,” O'Reilly Media, 2008.