Measurement 257 (2026) 118676

Contents lists available at ScienceDirect

Measurement

journal homepage: www.elsevier.com/locate/measurement

Optimizing object detection for autonomous robots: a comparative analysis of YOLO models

Rohan Vaghela [a], Darsh Vaishnani [a], Jigar Sarda [a,*], Amit Thakkar [a], Yash Nasit [a], Biswajit Brahma [b], Akash Kumar Bhoi [c]

[a] Department of Computer Science & Engineering, Chandubhai S. Patel Institute of Technology, Charotar University of Science & Technology, Anand, Gujarat, India
[b] McKesson Corporation, 32559 Lake Bridgeport St, Fremont, CA 94555, USA
[c] eSupport for Research, Burla 768018, Sambalpur, India

ARTICLE INFO

Keywords:
You Only Look Once (YOLO)
Object detection
Autonomous robots
Deep learning
Computer vison

ABSTRACT

Real-time and precise object detection is crucial for safety and efficient operation of autonomous robots. These systems highly rely on robust object detection to avoid obstacles, navigate in environments and interact safely with objects and people. Hence, this study aims to evaluate and compare the performance of various versions of the You Only Look Once (YOLO) model for real-time object detection in autonomous robotics applications. The study specifically analyses YOLOv5, YOLOv8, YOLOv9, and YOLOv10 performance and measures key evaluation metrics such as mean Average Precision (mAP), Recall and Precision. The models were trained on a custom dataset of campus movements, collected using a depth camera mounted on a semi-autonomous robot. Experimental results illustrates that the YOLOv5 achieves a mAP50 score of 72.6 %. YOLOv8 variants (nano, small, and medium) achieves mAP50 scores of 81.00%, 81.50 %, and 81.90 %, respectively. YOLOv10 models (nano, small, and medium) achieves mAP50 scores of 80.00 %, 71.70 %, and 72.50 %, respectively while YOLOv9c model demonstrated the highest accuracy, achieving an mAP50 score of 82.20 %, outperforming other YOLO versions. These findings provide perceptions into suitability of various YOLO models for real-time robotic applications, guiding the selection of optimal architectures for autonomous navigation and interaction.

1. Introduction

Since 1980s, robotics has undertaken notable advancements, particularly with the emergence of autonomous robotics technology, where object detection stands as an essential component of intelligent systems [1]. Object detection allows robots to understand and analyse their surroundings by leveraging complex machine learning algorithms, advanced sensor technologies, and real-time data processing to identify and classify objects within a vehicle's environment [2,3]. The evolution of computer vision and artificial intelligence has significantly pushed the development of object detection technologies, driven by the growing dependence on autonomous systems such as self-driving cars, delivery robots, warehouse automation solutions, and search-and-rescue operatives. These systems demand accurate, real-time object detection and tracking to operate effectively in dynamic and unpredictable settings. However, deploying real-time autonomous applications introduces significant challenges, particularly the scarcity of comprehensive custom-made real-world datasets complicates the autonomous robotics. Though, current datasets often face significant challenges that limit their effectiveness in real-world applications. One crucial issue is the incorrect camera angles, as images are typically captured at human eye level, whereas autonomous robots operate from a lower viewpoint, making object detection less effective in real-world scenarios. Additionally, there is a resolution mismatch, as many datasets recorded in high resolution, which simplifies object detection, while robots in real-world environments often use lower-resolution cameras, making recognition more difficult. Furthermore, inconsistent object labelling in some datasets, where objects are only labelled after a certain distance prevents early detection, weakening robot's capability to identify and avoid obstacles on time. These challenges collectively weaken the reliability and performance of autonomous systems in dynamic environments.

Additionally, many autonomous systems operate on resource-constrained embedded platforms such as Jetson's Xavier/Nano, or Raspberry Pi, where traditional large-scale models fail due to extreme

* Corresponding author.
E-mail addresses: 22cs088@charusat.edu.in (R. Vaghela), 22cs089@charusat.edu.in (D. Vaishnani), jigarsarda.ee@charusat.ac.in (J. Sarda), amitthakkar.it@charusat.ac.in (A. Thakkar), 22cs043@charusat.edu.in (Y. Nasit), Biswajit.Brahma@gmail.com (B. Brahma), akashkrbhoi@gmail.com (A.K. Bhoi).

https://doi.org/10.1016/j.measurement.2025.118676
Received 9 November 2024; Received in revised form 2 August 2025; Accepted 7 August 2025
Available online 9 August 2025

Table 1
Background study of different YOLO models for Object Detection.

Reference	Model Name	Contribution	Limitations	Methodology	Accuracy
[37]	BP-YOLO	Combines YOLOv7 with SimAM attention and DCNv2 for product detection in complex environments; uses BlazePose for shopping behavior recognition in unmanned vending machines.	High reliance on BlazePose for behavior recognition may limit accuracy in occluded hand scenarios; model complexity may affect edge deployment.	YOLOv7 backbone enhanced with SimAM attention, DCNv2 for deformable convolution, BlazePose for hand keypoint tracking.	96.17 % mAP for product detection; 92–98 % shopping behavior recognition under varied conditions.
[38]	CSP Partial-YOLO	Introduces PHDC and CSP Partial Stage for lightweight remote sensing detection; uses coordinate attention for small object localization.	May not generalize well to non-remote sensing tasks; trade-off between extreme lightness and detection robustness not thoroughly explored.	Combines PHDC block (partial + hybrid dilated convolution), CSP Partial Stage, and Coordinate Attention within PPYOLOE-R framework.	Outperforms YOLOv8, YOLOX, and RTMDet on DOTA and SODA-A datasets with lower latency and higher accuracy.
[39]	YOLOv5/6/7/8	Evaluates YOLOv5–v8 on wrist abnormality detection using pediatric X-rays; YOLOv8m achieves 0.92 sensitivity and 0.95 mAP, outperforming Faster R-CNN.	Dataset is limited to pediatric X-rays; generalization to adult or other fracture types is not addressed.	Comparative evaluation of YOLOv5-v8 variants on GRAZPEDWRI-DX dataset using single-stage detection pipeline.	YOLOv8m achieves 0.92 sensitivity and 0.95 mAP; YOLOv6m shows best overall class sensitivity (0.83); YOLOv8x highest mAP (0.77) across all classes.
[40]	YOLO-PL	Improved YOLOv4 for helmet detection with E-PAN, DCSPX, L-VoVN modules; lightweight and robust under occlusion and small object scenarios.	Limited to helmet detection tasks; scalability and performance on general-purpose detection not validated.	Enhances YOLOv4 using E-PAN for feature fusion, DCSPX instead of SPP, L-VoVN for lightweight backbone, and Swish activation.	Achieves comparable AP to YOLO-P with 36.8 % fewer parameters, 33.1 % lower computation, and improved inference speed.
[41]	YOLO-NL	Proposes global dynamic label assignment, Rep-CSPNet and SSPP structure for fast and robust detection in dense/occluded scenes.	Complexity from multiple custom components may hinder generalizability and inference optimization on lightweight hardware.	YOLOX-based detector using global dynamic label assignment, Rep-CSPNet, shortest-longest gradient, PANet with self-attention.	Achieves 52.9 % mAP on COCO 2017, 98.8 % accuracy on FMD dataset with 130 FPS for real-time face mask detection.
[42]	YOLOv8-MNC	Introduced for pediatric melanoma diagnosis with enhanced detection of small targets and better global feature learning.	Focused on a specific medical dataset; generalizability to other medical imaging domains or pediatric conditions needs further validation.	Built on YOLOv8, with a small-target detection layer, NWD Loss for accuracy, multi-head self-attention for global features, and CARAFE up sampling to reduce information loss.	Achieved mAP of 81.89 %, improving prior methods by 5.65 % on pediatric melanoma dataset.
[43]	YOLOv8-Cyclone	Enhances early tropical cyclone detection using satellite imagery for improved disaster preparedness and response.	Performance and generalizability in non-tropical or low-resolution satellite imagery scenarios need further validation.	Customized YOLOv8 framework to capture circular cloud structures and atmospheric patterns; trained on labeled satellite image datasets for cyclone detection.	Reported high detection accuracy and computational efficiency, outperforming previous cyclone detection methods.
[44]	YOLOv8-SuperCyclone	Enhances real-time identification of super cyclones using satellite imagery, bypassing limitations of traditional NWP forecasting.	Model performance under rare, evolving storm structures and unseen atmospheric conditions needs further validation.	Fine-tuned YOLOv8 on labeled cyclone satellite images to detect cyclonic patterns without strict physical assumptions; evaluated using precision-recall and validation loss.	Reported high accuracy and robustness in identifying cyclone structures; validated with precision-recall and loss metrics.
[45]	YOLOv5/v7/v8/v9c/v10 (Blood Cell Detection)	Evaluates YOLO model family for detecting RBCs, WBCs, and platelets; YOLOv5 found most effective overall with strong precision and mAP.	Evaluation limited to detection only; segmentation and classification of abnormal cells not addressed; real-time deployment in clinical settings untested.	Used CNN-based YOLO variants (v5, v7, v8-m/S/N, v9c, v10-m/S/N) on medical images to detect RBCs, WBCs, and platelets; assessed precision and mAP50.	YOLOv5 achieved highest mAP50 of 93.5 %; YOLOv10n had 98.6 % precision for WBCs; YOLOv10m led RBC detection (85.1 %) and YOLOv5 for platelets (88.8 %).
[46]	YOLOv7/v8 (Brain Tumor Detection)	Compares YOLOv7 and YOLOv8 for automated brain tumor detection on MRI scans; YOLOv8 shows superior precision and recall.	Limited to detection tasks; segmentation and classification of tumor types not explored; potential overfitting risk due to dataset size variability.	Applied YOLOv7 and YOLOv8 models on annotated MRI images for detecting brain tumors of various sizes, shapes, and locations; evaluated precision, recall, and validation accuracy.	YOLOv8: 96.5 % validation accuracy, 95.9 % precision, 93.3 % recall; YOLOv7: 87.6 % validation accuracy, 81.9 % precision, 85.4 % recall.
[47]	YOLOv3/5	Real-time physical distancing monitoring system. Person detection and distance estimation.	Generic dataset (COCO) may not be optimal for all indoor scenes. Accuracy vs FPS trade-off across models. 2D estimation may cause distance errors due to perspective.	Pre-trained on MS COCO– Real-time estimation of distance between detected persons.	YOLO v3: – Accuracy: 87.07 %
[48]	YOLOv3/5	Real-time system for measuring indoor area and people count-Calculates max allowed capacity based on area. Person detection in predefined video region	Manual ROI selection required-No depth or 3D estimation considered. Potential inaccuracy in area computation. Not tailored for dynamic/multiple ROIs	YOLO models with MS COCO weights. Manual selection of ROI in video. Estimated area size and computed max occupancy.	YOLO v3 Accuracy: 88.39 %

computational demands, long inference times, and high-power consumption. To avoid these limitations, lightweight object detection frameworks like YOLO offer a feasible solution, characterized by reducing parameter counts, minimizing latency, and optimize for real-time inference on low-power devices [1]. This seamless integration of YOLO with embedded systems emphasizes its suitability for autonomous applications, enabling efficient and reliable performance in real-world deployments. These advancements jointly address the technical barriers to achieving robust object detection in modern robotics.

Currently, the domain of AI offers numerous methods for object

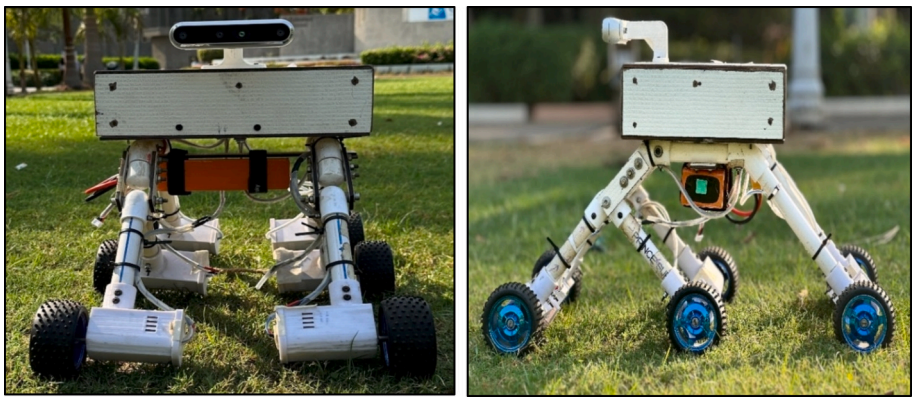

Fig. 1. Six-Wheeled Rover.

Fig. 2. Some annotated and labelled images from dataset.

Table 2
System Configuration for the Implementation Server.

CPU	Intel Xeon 16 Core (x2)
RAM	64 GB DDR4 ECC (x2)
HDDs	2 TB SATA Enterprise
SSD	240 GB SATA
Graphics Card	NVIDIA RTX A5000
Operating System (OS)	Unix OS

identification, including Two-stage and One-stage object detection [1]. Two-stage algorithms comprise of generating potential regional boxes and regressively classifying these boxes to determine the target's size, category, and location. In contrast, one-stage algorithms directly compute the whole image and predict the objects category, location, and size at the same time using regression methods, thereby enhancing detection efficiency. YOLO is a one-stage algorithms renowned for their quickness and precision. SSD [4] and RetinaNet [5] are frequently used one-stage object identification methods.

YOLO algorithm was first introduced in study [6] in computer vision field in the year 2015. The YOLO family of algorithms are among the first single-stage object identification techniques, which were developed after the introduction of Spatial Pyramid Pooling Networks [7], Region-based CNN [8], and Faster R-CNN [9]. YOLO presents an innovative framework specifically developed to increase the real-time efficiency of object detection. This algorithmic series enables rapid recognition by separating the picture into grids and making predictions for classification and bounding boxes for each grid segment [10]. It is characterized by its fast-computing speed and small model size [11]. The YOLO architecture is characterized by its exquisite simplicity, consisting of 3 primary components: the head, backbone, and neck. The backbone network consolidates and enhances characteristics extracted from the input pictures. In the neck, these characteristics blend and merge before being sent to the head. The neural network's brain uses these characteristics to predict the placements and classifications of bounding boxes, which are then immediately outputted [12]. Consequently, YOLO can

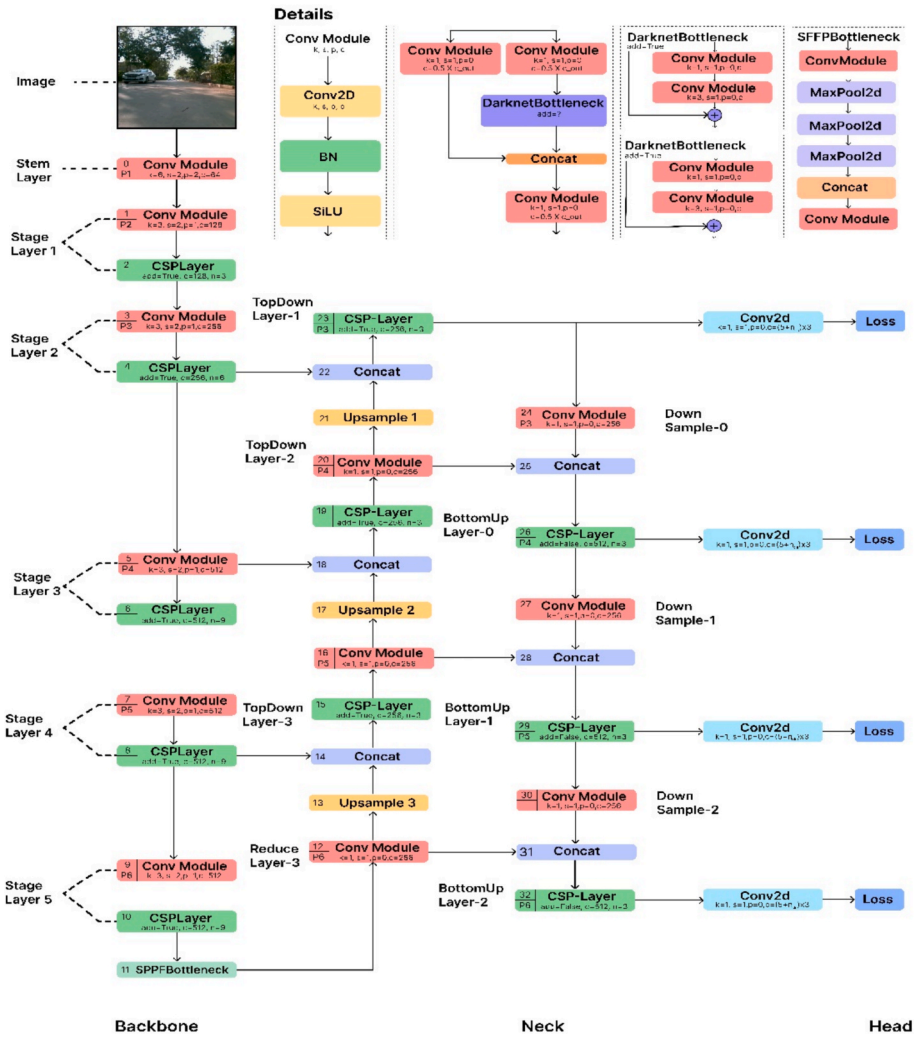

Fig. 3. Detailed Architecture of YOLO v5 model.

acquire widely applicable qualities that may be applied to various domains. Despite its commendable degree of accuracy, its pace often falls short of expectations. Due to its ability to attain a detection rate of 45 fps and much better mAP index compared to previous real-time detection systems, it is evident that this system stands out [13] in real-world applications. However, despite its advantages, several drawbacks must be considered for the applications. One prominent limitation is that the YOLO model prioritizes real-time detection over accuracy, which often results in reduced precision compared to more computationally intensive models such as FR-CNN or Mask R-CNN. This trade-off can manifest as lower precision scores and missed detections, particularly when dealing with small objects or those in densely packed environments. Additionally, YOLO's grid-based approach poses challenges in detecting small or low-resolution objects, as it struggles to capture fine details effectively, limiting its performance in scenarios requiring high granularity. Another significant issue arises in occluded object detection, where YOLO frequently fails to accurately identify partially hidden objects or mistakenly splits a single object into multiple detections—a problem especially prevalent in crowded settings. Furthermore, the algorithm's single-stage detection approach, which processes the image in one pass, can lead to misclassification in complex scenes featuring overlapping or occluded elements, declining its reliability. However, modern architectures, have solved these issues to some extent.

Also, the application of YOLO with smart technology are across various domains, enhancing efficiency, safety, and automation. In Industrial automation, YOLO algorithms enable rapid and precise object recognition, streamlining production workflows and ensuring quality control. However, integrating YOLO with smart technology presents challenges. The computational complexity of YOLO models necessitates optimization techniques like model pruning and quantization for deployment on low-power devices [14,15].

The novelty of this research lies in the following key contributions:

1. Our study indeed presents a unique focus by leveraging a depth camera mounted on a semi-autonomous mobile robot to capture real-time navigation data. This approach enables the integration of depth information with object detection, offering a more context-aware and safer navigation system, which is crucial for autonomous robotic applications.
2. Unlike many existing studies that rely on synthetic or static datasets, our work utilizes a custom-collected dataset from a mobile robot navigating a real campus environment.
3. Our research provides a comprehensive performance comparison of object detection using four recent YOLO architectures YOLOv5, YOLOv8, YOLOv9c, and YOLOv10 along with their nano, small, and medium variants where applicable. This extensive benchmarking offers valuable insights into the trade-offs between model size, accuracy, and suitability for real-time robotic applications.

The rest of the study is organized as follows: Section 2 provides literature work on the object detection task. Section 3 presents information about dataset formation and a brief overview of various YOLO

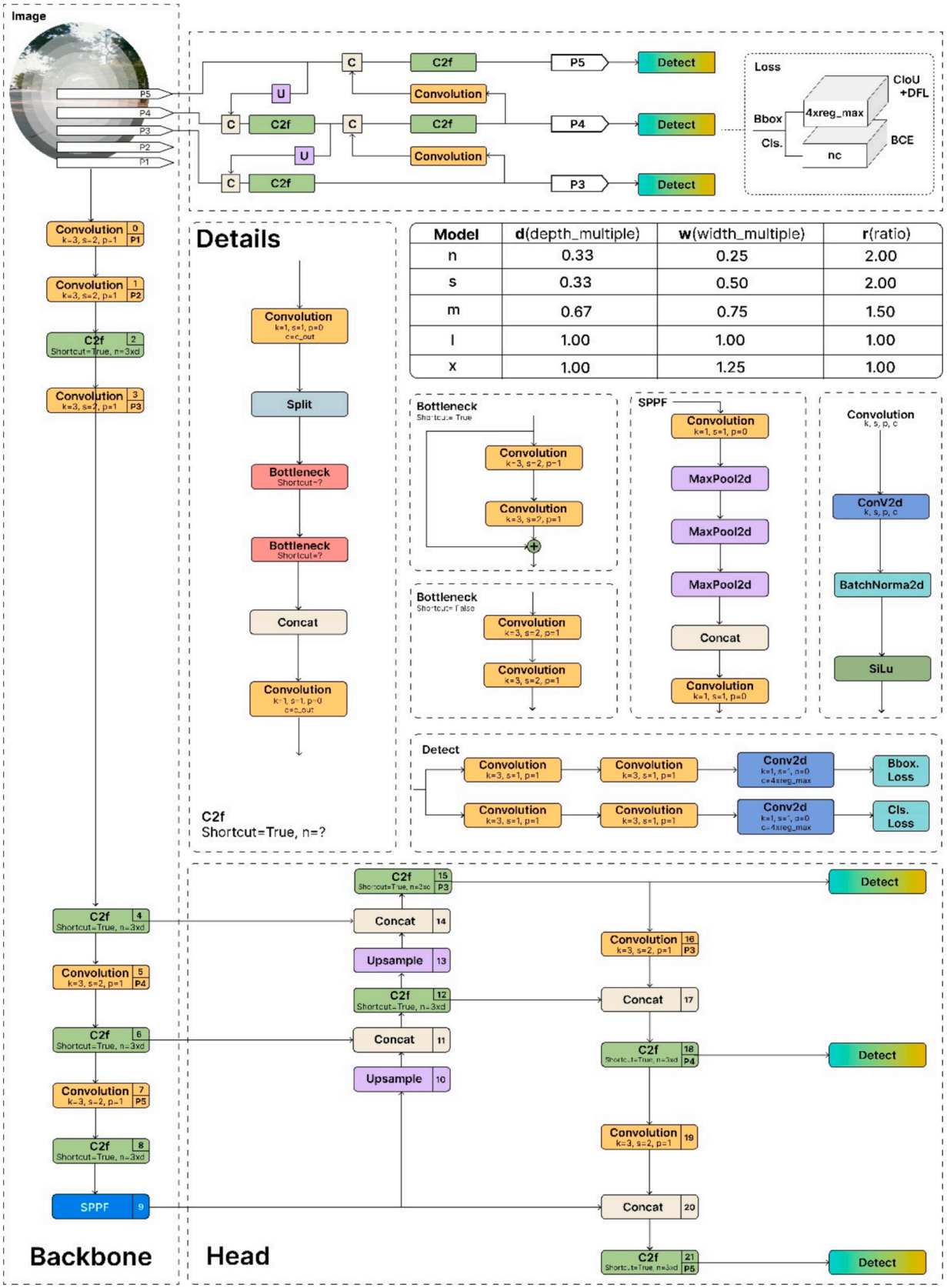

Fig. 4. Detailed Architecture of YOLO v8 model.

Table 3
Shows the variants of YOLO v8 models.

Name	Layers	Parameters	Gradients	GFLOPs
YOLO v8 nano	225	3,011,043	3,011,027	8.2
YOLO v8 small	225	11,135,987	11,135,971	28.6
YOLO v8 medium	295	25,856,899	25,856,883	79.1

models. Section 4 discusses the evaluation metrics used in the study. Section 5 presents the results of the implemented work, followed by the conclusion in Section 6.

2. Literature Review

Li et al. [16] introduced an enhanced version of the YOLOv8 method that incorporates novel components from the real-time detection transformer (RT-DETR). The system, was trained on a dataset that includes occlusion situations and utilizes an exclusion error function tailored for obstruction, greatly enhances the detection correctness. Wang et al. [17] created SM-YOLO v5, a tomato detection model that enhances the accuracy of identifying tiny target tomatoes. This was accomplished by adding a dedicated small-target identification layer and utilising MobileNetV3-Large as backbone. The developed model was applied to the tomato dataset from the plant factory, resulting in a 1.4 % increase in mAP score compared to the baseline YOLOv5.

Zeng et al. [18] used MobileNetV3 to create a simplified backbone network by replacing the attention layer in YOLOv5s with a down sampling convolutional layer. The improved network achieves a mean accuracy of 96.90 % for tomatoes at different ripening phases, while reducing the inference speed by 64.88 % compared to the novel YOLO v5s. Faruq Aziz et al. [19] introduced a very effective method for detecting skin lesions using the YOLO v9 network. This method accurately recognizes different kinds of skin lesions in about 3000 photos. Their overall accuracy and mAP were 60.5 % and 81.4 % respectively. Separate research [20] examines enhancements in the detection of road potholes and introduces an improved YOLO v5s model that utilizes a SIoU loss function and an attention mechanism. The recommended

Table 4
Shows the variants of YOLO v10 models.

Name	Layers	Parameters	Gradients	GFLOPs
YOLO v10 nano	385	2,707,430	2,707,414	8.4
YOLO v10 small	402	8,067,126	8,067,110	24.8
YOLO v10 medium	498	16,485,286	16,485,270	64.0

Table 5
Various Hyperparameters values used for training YOLO variants.

Hyperparameters	Values
Optimizer	AdamW
Learning Rate	0.002
Momentum	0.9
Weight decay	0.0005
Image Size	640
Batch size	16
Patience	100
Workers	8
Close mosaic	10
IoU threshold	0.7

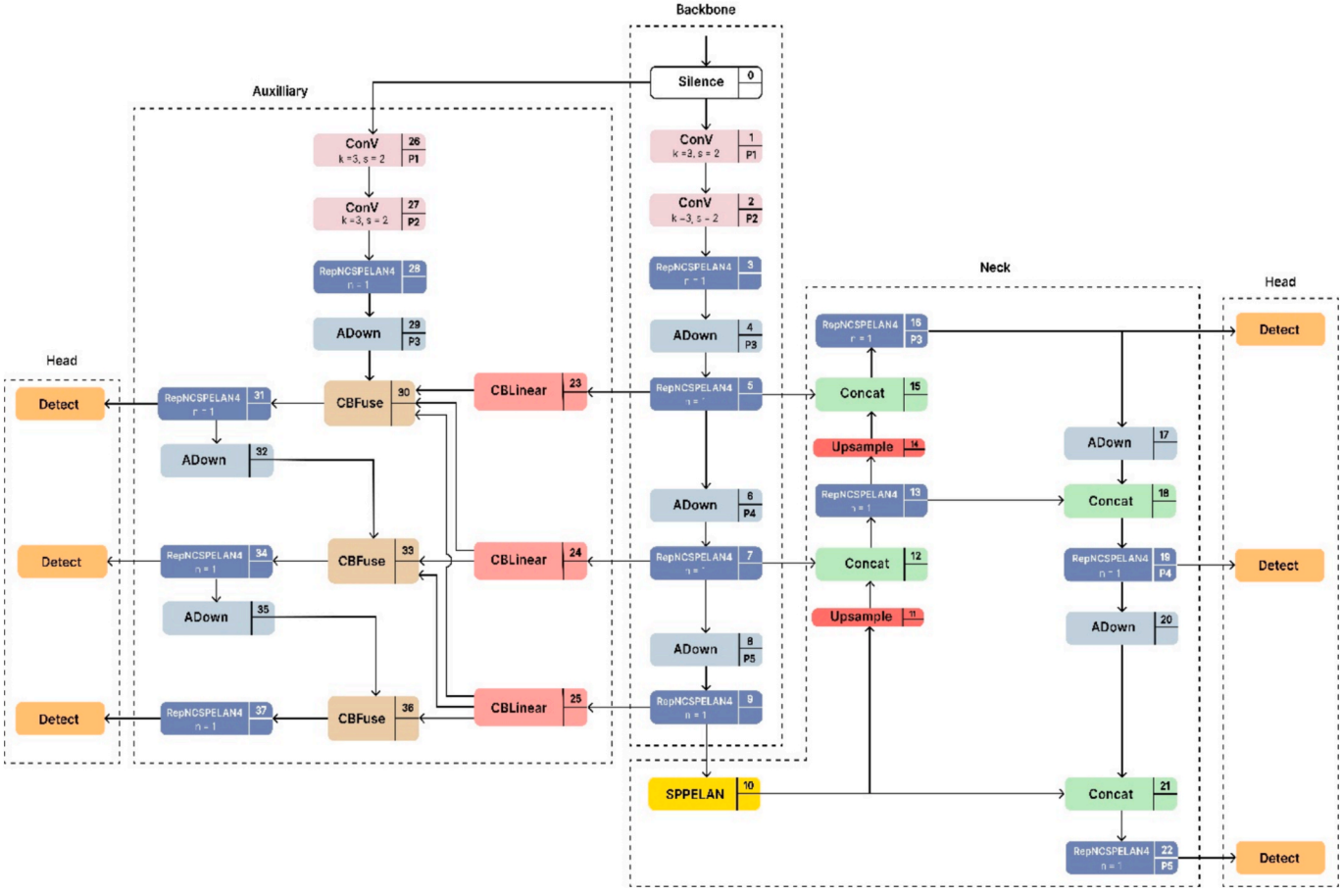

Fig. 5. Detailed Architecture of YOLO v9 model.

Table 6
Simulation Results for YOLO v5 model.

Name	Epochs	Precision (in%)	Recall (in%)	mAP50 (in%)	mAP50-95 (in%)
YOLO v5	25	45.45	56.81	49.56	26.02
	50	57.72	90.90	67.07	40.48
	75	61.64	78.52	70.73	41.29
	100	**62.00**	**79.50**	**72.60**	**46.20**

YOLO v5-based model meets real-time detection needs and ensures safety. It demonstrates enhanced detection accuracy and a reduced model size.

Xiang Yue et al. [21] developed a detecting technique for monitoring the health of fruits and facilitating their harvesting in areas where tomatoes are grown. The researchers enhanced YOLO v8 and rebranded it as RSR-YOLO by including an innovative FasterNet (IFN) module and a partial group convolution (PgConv) for more effective feature mining. In addition, researchers integrate the Gather and Distributed technique (GD) and revamp the feature fusion unit. The RSR-YOLO model attains precision, F1 score, recall, and mAP values of 91.6 %, 88.7 %, 85.9 %, and 90.7 %, correspondingly. Madarapu Sathvik et al. [22] introduced a method for detecting potholes using YOLO v7. The model's effectiveness was verified by analyzing diverse images specifically related to potholes. The model obtained an F1 score of 0.51 %. Hoang-Tu Vo [23] presents an innovative way to address the persistent problem of monitoring and counting ripe (or manual) tomatoes, which often relies on eye examination. The solution employs the YOLO v9 model for detection, which attains results with a mAP score of 0.88, precision score of 0.85, and recall of 0.83.

Riddhi et al. [24] used a YOLO v8 model to identify potholes, which had a compact model size of 4.9 MB. Their mean average precision (mAP) reached a value of 78.27 %. Tian Yong Wu et al. [25] enhanced the YOLO v8 model for remote sensing object recognition by introducing a lightweight convolution SEConv and an innovative Efficient Multi-scale Attention (EMA) mechanism. These components are merged in the network to create the SPPFE module. Their performance on the optical remote sensing dataset SIMD resulted in an accuracy of 86.5 %. Xiang Jia et al. [26] introduced an efficient and precise object detection system for autonomous driving that relies on an enhanced version of YOLO v5. The structure was enhanced via the process of structure reparameterization (Rep), and an extra approach of neural architecture search was also added. Following the improvisation, the experimental findings indicate a detection accuracy of 96.1 %.

Appe et al. [27] enhanced the network's capability to combine features by including the attention module in the convolutional block into the neck component of YOLO v5. Nevertheless, rather than improving network performance via the augmentation of computational and parametric capabilities, a growing body of research is pushing for a streamlined model design to enhance responsiveness in real-time detection [28,29]. The research [30] developed an intelligent street pothole detection method by altering the RetinaNet. This system is capable of detecting potholes and conducting metrological investigations utilizing 3D vision. The photogrammetric approach was used to construct a three-dimensional point cloud representation of potholes, which was then combined with a convolutional neural network (CNN) based system for detecting potholes. Consequently, it attained a commendable F1-score of 0.98 on the benchmark dataset, along with a mean error rate of less than 5 %. Chan Gao and colleagues [31] developed an enhanced Jiangnan private garden detection algorithm using YOLO v8. This architecture combines the Dynamic Head modules (DyHead), Diverse Branch Block (DBB) and Bidirectional Feature Pyramid Network (BiFPN) to enhance model accuracy, object detecting representational capabilities, and feature fusion. The improvements increased the accuracy by 8.7 %, resulting in a mAP@0.5 of 57.1 %.

In direction to overcome the difficulties of lightweight and real-time

Table 7
Simulation Results for YOLO v8 model.

Name	Epochs	Precision (in %)	Recall (in%)	mAP50 (in%)	mAP50-95 (in%)
YOLO v8 nano	25	57.64	75.76	63.52	37.21
	50	60.71	89.77	73.14	45.97
	75	64.58	73.73	77.40	46.99
	100	**74.20**	**80.70**	**81.00**	**52.00**
YOLO v8 small	25	60.63	67.04	57.71	34.51
	50	52.02	79.54	54.14	42.03
	75	66.00	83.82	74.42	46.97
	100	**71.90**	**90.10**	**81.50**	**51.7**
YOLO v8 medium	25	60.513	77.27	72.63	47.18
	50	62.00	77.29	75.20	51.90
	75	66.30	90.90	76.33	48.83
	100	**73.20**	**78.40**	**81.90**	**51.20**

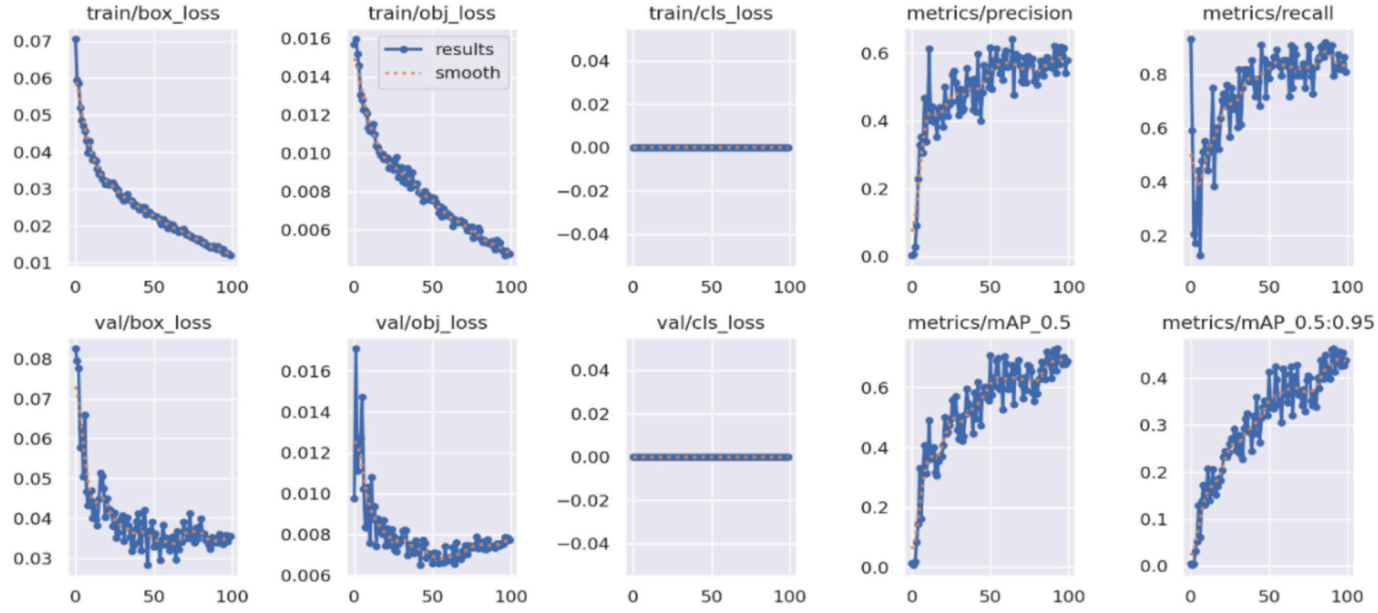

Fig. 6. Results YOLO v5 model.

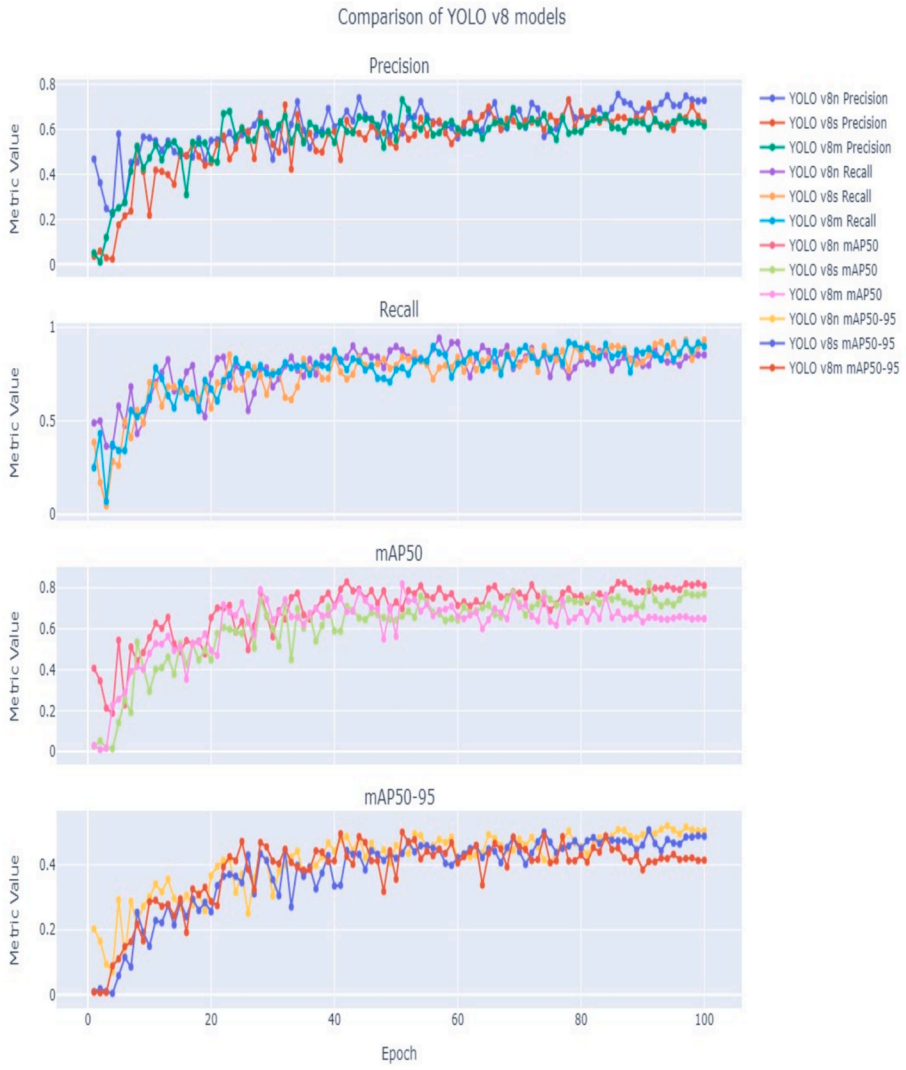

Fig. 7. Various parameter comparison of YOLO v8 models.

coal sorting detection, Qingliang Zeng et al. [32] introduced an enhanced version of YOLO v8. This was achieved by using the Laplacian image enhancement method and the CBAM attention mechanism. In addition, the EIOU loss function was used, resulting in attaining outstanding performance with a mAP of 99.5 %. Another study [33] enhances YOLOv8 by integrating the C2f-FocalNextBlock and ResNet-EMA modules for improved feature extraction and cross-dimensional interactions, along with a slim-neck structure for better multi-scale feature integration. These modifications achieve a high mAP of 79.9 %, demonstrating the effectiveness of deep learning in wind turbine damage detection. Xing Jiang et al. [34] enhance YOLO v8 by integrating the huge kernel block from UniRepLKNet into the backbone module to optimize it. In addition, they include C2fSTR, a fusion of the C2f module and Swin transformer, which combines Cross-Stage Partial Fast Spatial Pyramid Pooling (SPPFCSPC) with attention methods. Experimental findings on the URPC2019 dataset demonstrate that YOLO v8-MU achieves a mAP of 78.4 % at a confidence threshold value of 0.5. Furthermore, it attains an accuracy of 75.5 % on the Aquarium dataset and 80.9 % on the URPC2020 dataset. The study [35] enhances YOLOv8 for underwater object detection by integrating a weighted multi-error information entropy approach, modelling illumination as structured noise, and optimizing the loss function with a minimum weighted error entropy criterion. Tests on underwater datasets show it achieves a mAP of 96.7 % and 116.6 FPS, surpassing baseline YOLOv8 and other advanced models in accuracy and speed. Further the study [36] proposes an enhanced method for detecting tropical cyclones using the YOLOv8 object identification algorithm, improving its architecture to identify key cyclone features in satellite data. Trained on a diverse dataset, the model outperforms existing methods in accuracy and efficiency, offering significant benefits for early warning systems and disaster preparedness in coastal areas. Table 1 summarizes the performance and review findings of different YOLO models applied to object detection tasks.

3. Methodology

3.1. Dataset

In today's era of automation and self-driving car, object detection plays a crucial role as without object detection, object cannot be avoided. So, a good dataset is extremely noteworthy objective for the detection. So, considering the importance of creating a dataset, we initiated it by developing a custom dataset for object detection of our surrounding campus environment by mounting an "Intel RealSense depth camera D400" on a six-wheeled rover as shown in Fig. 1. Mounting the camera at the rover level helps model to understand more and make it easier for it to predict. A video was a recorded using this camera with an external support of laptop at discrete time when traffic of the campus was at its peak such that a greater number of human

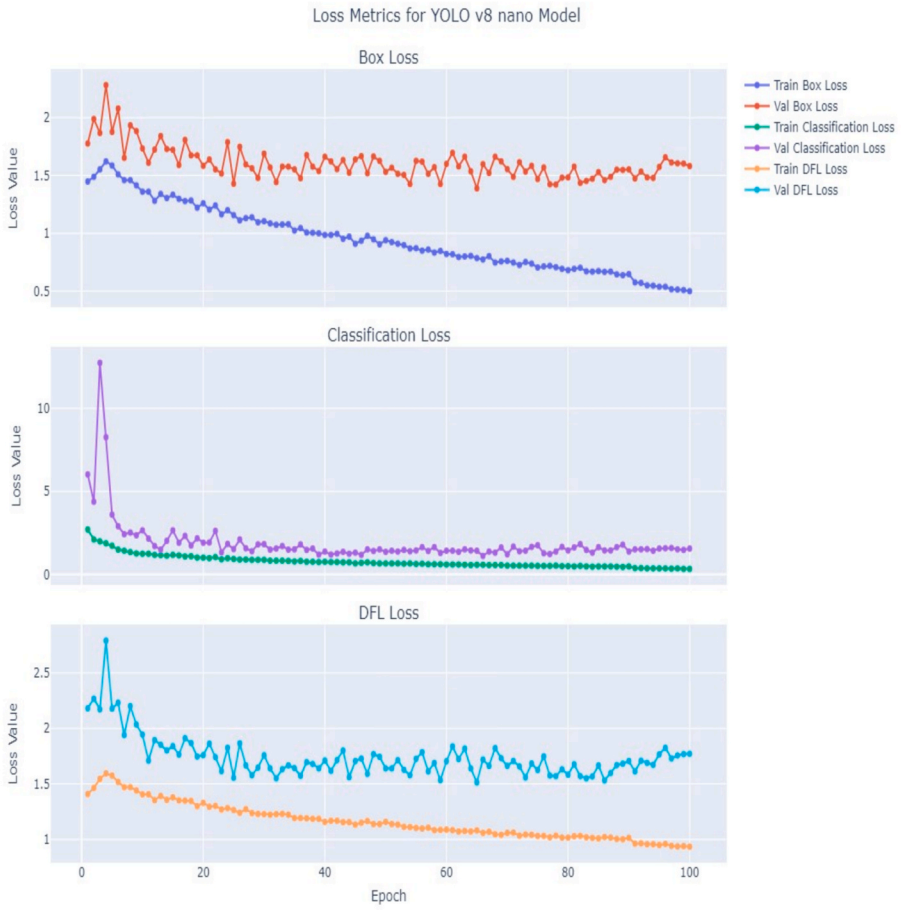

Fig. 8. Various Loss values comparison for YOLO v8 nano model.

objects and vehicle appears. For the collection of videos, a python library 'pyrealsense2' was used.

Initially, video was recorded at a frame 0f 480 × 480 pixels. As moving further, the video was then converted into discrete frames with same size for the creation of dataset. The video was recorded at 30 frames per second and every 30th frame is extracted from the video. Overall, 6 videos have been recorded with an average time span of 20 min each. At the end a total of 5646 frames was extracted from all the videos.

For the annotation and labelling of the objects within these frames, an open-source platform named Roboflow [49] was used. Annotation was done manually by the labellers while annotating many empty frames were encountered which were discarded as there was no object found or the object was too far from the frame. Our main aim, was to only annotate object which were closer to the frame or those who occupy at least half of the frame height from the rover's distance because detecting a far object can make model to take a particular action at an early stage which is of no use. After annotating, the images were resized to 640 × 640 pixels as a data pre-processing step using a Roboflow built-in function. Further for data augmentation part, three different augmentations process were applied which includes horizontal flip, blurring (up to 1.5 pixels), and brightness adjustment (between −20 % to 20 %). Fig. 2 shows the annotation and labelling of the objects in images using Roboflow platform.

After the preparation of dataset there was an imbalance in dataset between the object and background class. To overcome this problem, total 531 images were added in the dataset. In total 1483 images were used for dataset formation. Before the implementation, the dataset was split into 3 parts namely training, validation and testing with a split ratio of 80:10:10.

3.2. System Configuration

Table 2 shows the configuration for the implementation sever used for the study.

3.3. YOLO v5

It was first introduced by Glen Jocher, CEO and founder of Ultralytics in 2020 [50], which signifies an advancement in the YOLO series. It is the foremost native implementation of the YOLO family architectures to be written in PyTorch, rather than Darknet53 [51]. YOLOv5 integrates an Ultralytics algorithm known as AutoAnchor. This pre-training tool evaluates and refines bounding boxes to ensure they are optimally suited for the training parameters, such as image dimensions and dataset. Given below, Fig. 3 shows the detailed architecture of YOLO v5 model. The architecture uses an improved CSPDarknet53 that begins with a Stem which is a large window sized stride convolution layer and used to reduce computational costs and memory. This is tailed by a series of convolutional layers that extracts pertinent features from image. YOLO v5 offers five scaled versions: YOLO v5n, YOLO v5s, YOLO v5m, YOLO v5l, and YOLO v5x. These variants differ in the width and depth of their convolutional modules to cater to specific hardware and application requirements. For example, YOLO v5s and YOLO v5n are lightweight models designed for small resource devices, whereas YOLO v5x is enhanced for high performance, though with a trade-off in speed.

3.4. YOLO v8

YOLO v8, was released by Ultralytics in 2022 [52]. As a single-stage detection algorithm, YOLO v8 can recognize and categorize objects in an

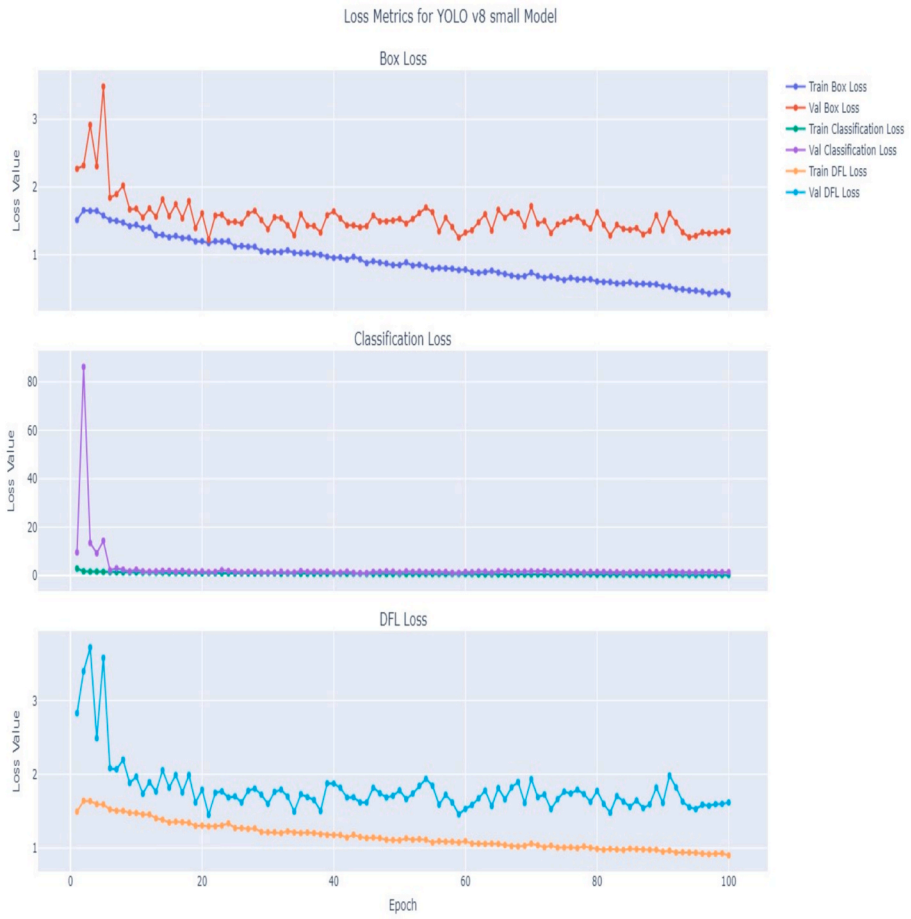

Fig. 9. Various Loss values comparison for YOLO v8 small model.

image with just one pass, making it extremely fast and suitable for real-time applications. A detailed architecture of YOLO v8 is shown in Fig. 4. YOLO v8 modifies the YOLO v5 C3 unit by replacing it with the C2f unit. Additionally, it retains the PAN and feature pyramid network (FPN) structure and eliminates the convolution assembly in the up-sampling stage to decrease computational intricacy. YOLO v8 employs a revised version of the Efficient-Net backbone network, which balances accuracy and computing efficiency. Specifically, the backbone network integrates a FPN to combined features at diverse scales, enhancing the model's capability to detect objects of various dimensions.

The head of YOLO v8 consists of several convolutional layers, ending in an output layer that produces detection results. This head is composed of three predictive branches: class probabilities, forecast bounding boxes and objectness scores. The number of object classes and anchor boxes regulates the quantity of output channels in each division. YOLO v8 offers five variations. YOLO v8n, the smallest, has a parameter size of 3.2 million and the lowest FLOPS at 8.7 billion, making it the most lightweight among the YOLO v8 models. YOLO v8s, another lightweight model, is suitable for low-resource devices, while the larger and extra-large models are designed for high-performance applications. The choice of a specific YOLO v8 model depends on the dataset size and the particular application requirements. Table 3 shows the comparison between various YOLO v8 model variants with parameters, gradients and GFLOPs value. From the table, medium variant has highest number of layers and GFLOPs values.

3.5. YOLO v9

Introducing YOLO v9, a ground breaking advancement in object detection, developed by Chien-Yao Wang in 2024 [53]. It introduces different techniques such as the Programmable Gradient Information (PGI) and Generalized Efficient Layer Aggregation Network (GELAN) to effectively address issues related to computational competence and data loss. The phenomenon of diluted or loss data through the forward pass in neural network due to alterations within the layers is acknowledged as information bottleneck. Due to this accuracy gradient is a not achieved for effective learning. So, to overcome this problem, YOLO v9 introduces PGI that preserved the gradient information through the network and help the model to learn and improve more effectively to recognize objects more accurately. The architecture of the YOLO v9 PGI is shown in Fig. 5 which contains mainly three components: The main branch, an auxiliary reversible branch, and multi-level auxiliary information. The core branch is the original architecture used for inference.

The working of the auxiliary reversible branch is to produce consistent gradients and apprise network parameters. The loss function can suggest direction and prevent the chance of determining misleading correlations from insufficient feed forward features that are not as crucial to the target by giving details that maps from data to targets. The concept of multi-level auxiliary information involves integrating a network between the FP hierarchy layers of the main branch and auxiliary supervision. This integration network amalgamates the gradients returned from various prediction heads, as illustrated in the Fig. 5. The GELAN is new network architecture formed by combination of both CSPNet and ELAN. Gradient path planning served as the foundation for the creation of this lightweight network architecture. It resolves the information bottleneck problem which improve efficiency and accuracy. The YOLO v9s model surpasses the YOLO MS-S in parameter efficiency and computational load, achieving an enhanced AP

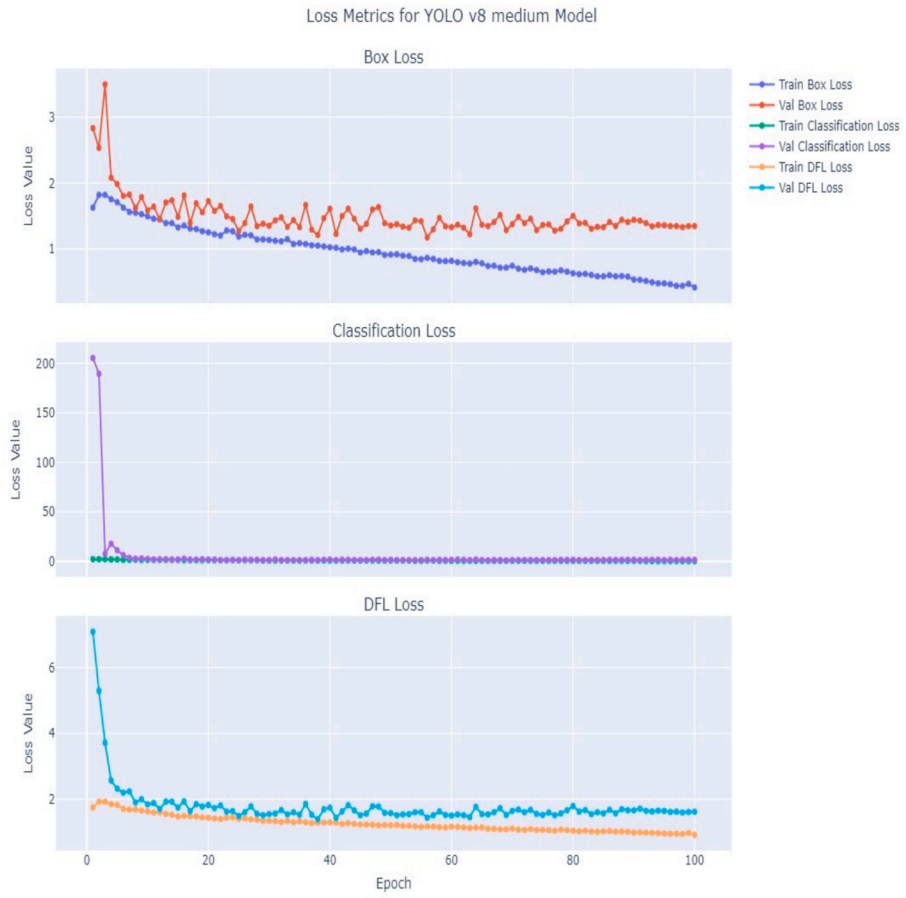

Fig. 10. Various Loss values comparison for YOLO v8 medium model.

Table 8
Simulation Results for YOLO v9c model.

Name	Epochs	Precision (in %)	Recall (in%)	mAP50 (in%)	mAP50-**95** (in %)
YOLO v9c	25	60.63	67.04	57.74	34.54
	50	52.00	79.54	64.14	42.03
	75	66.00	83.82	74.42	46.97
	100	**71.30**	**85.20**	**82.20**	**50.80**

improvement of approximately 0.4 to 0.6 %. Meanwhile, the YOLO v9m and YOLO v9e models demonstrate remarkable advancements in optimizing the balance between model complexity and detection performance, offering substantial reductions in parameters and computational requirements while concurrently enhancing accuracy.

3.6. YOLO v10

Researcher, et al. Ao Wang [54] introduces a novel real-time end-to-end object detector YOLO v10, the latest model of YOLO family introduced on 25 May 2024. YOLO v10 is developed by taking YOLO v8 as a base model, due to its admirable latency-accuracy stability and its availability in numerous model sizes. It is designed based on the reliable dual assignments for NMS-free training and achieve a complete efficiency-accuracy driven model. Consistent dual assignments for NMS-free training significantly improves the effectiveness of end-to-end detection. This results in the avoidance of traditional Non-Maximum Suppression (NMS) steps and streamlining the detection process. YOLO v10 model architecture comprises of Compact Inverted Block (CIB), Large-Kernel Convolution, Partial Self-Attention (PSA), and Spatial-Channel Decoupled Down sampling. YOLO v10 has outperform previous YOLO version in both accuracy and efficiency across all the variants. YOLO v10 has 6 other versions such as YOLO v10n, YOLO v10s, YOLO v10m, YOLO v10b, YOLO v10l, YOLO v10x in which YOLO v10b achieves 25 % fewer parameters and 46 % less latency as compared to YOLO v9c with the same parameters. Table 4 shows the comparison between various YOLO v10 variants. From the table, it can be seen that medium variants have highest parameters values while nano has lowest.

4. Evaluation Metrics

Precision is a key metric in evaluating the performance of a predictive model, specifically concentrating on the accuracy of true predictions. It measures the amount of TP outcomes among all instances that the model has predicted as positive. Mathematically, precision is evaluated by dividing the number of TP to the sum of true positives and false positives (FP), as shown in Eq. (1).

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

Further, recall or Sensitivity, is a vital metric for assessing the effectiveness of a predictive model, particularly in its ability to identify all relevant instances. As described in Eq. (2), recall is calculated by dividing the number of TP to the sum of FN and TP.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

Average Precision is represented as the area under the precision-recall curve for each individual category. Eq. (3) is used to compute the subsequent average precision for each category. To accurately represent the

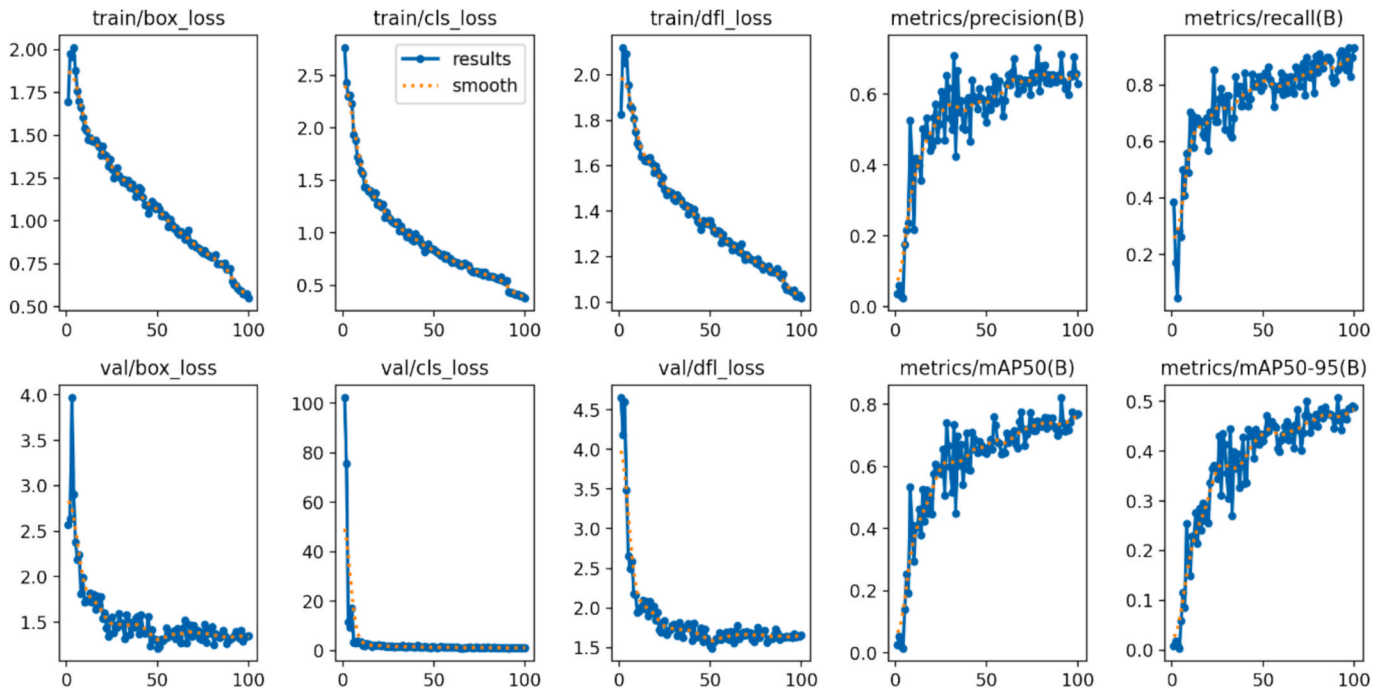

Fig. 11. Various parameters comparison for YOLO v9c model.

Table 9
Simulation Results for YOLO v10 model.

Name	Epochs	Precision (in %)	Recall (in%)	mAP50 (in%)	mAP50-95 (in%)
YOLO v10 nano	25	58.55	73.86	55.91	34.42
	50	56.95	77.27	65.90	43.50
	75	53.83	81.81	68.96	44.86
	100	**74.50**	**75.00**	**80.00**	**46.70**
YOLO v10 small	25	57.70	63.63	56.82	32.74
	50	54.35	81.81	58.23	38.43
	75	61.80	85.22	69.52	43.88
	100	**62.90**	**88.60**	**71.70**	**45.35**
YOLO v10 medium	25	31.21	59.09	36.88	19.60
	50	53.22	73.86	58.91	31.50
	75	55.48	76.13	65.89	40.20
	100	**59.40**	**84.90**	**72.50**	**48.00**

model's performance across the entire dataset, mAP for any model can be computed using Eq. (4), where n is the total number of average precisions.

$$AP = \int_0^1 Precision.Recall \; dR \tag{3}$$

$$mAP = \frac{1}{N}\sum_{i=1}^{n} AP_i \tag{4}$$

IoU is a measurement used in detection to evaluate the correctness of bounding boxes. By utilizing bounding boxes to localize objects and identify localization errors, IoU is calculated as the ratio of the area of intersection between the ground truth bounding box and the predicted bounding box to the combined area of both boxes, as shown in Eq. (5).

$$IoU = \frac{Overlapping \;\; Area}{union \;\; Area} \tag{5}$$

Another key performance metric for object detection is the F1-score as shown in Eq. (6), which combines the Recall and Precision metrics to provide an additional comprehensive measure of the model's accuracy.

$$F1 - score = \frac{2{*}Precision{*}Recall}{Precision + Recall} \tag{6}$$

5. Results

This results section provides an inclusive analysis of the performance of numerous YOLO family models trained on the generated dataset. Table 5 shows the hypermeters values used for training all the YOLO variants. All the hyperparameters value for each variant and its sub variants has been kept same so that the final performance can be made on the same ground basis. Further, initially, the widely recognized object detection algorithm YOLO v5 was employed for the task. Further, Table 6 shows the results of the YOLO v5 model at various epoch values, detailing precision, recall, and mAP values at different thresholds. From Table 6, an increase in the mAP50 value is observed with the rise in epoch values, alongside a similar pattern in other metrics. Notably, the highest mAP50 of 72.60 % is achieved at 100 epochs, with average recall and precision values of 79.50 % and 62 %, respectively.

Fig. 6 shows the various graphs plotted between various parameters and epochs values for the highest mAP value of 72.60 %. There are three different loss function used for the error calculation. The box loss represents the error in the prediction of the bounding box while object loss shows the error of predicting object in the images. The classification loss represents error in classification of object class. In Fig. 6 both training and validation graphs are plotted with various epochs values. All the loss values are decreasing with increase in epochs values. The training curve for box and object loss seems to be a smooth decreasing curve while the validation curve for the same have a large oscillation in the values. The classification loss is zero because a single class is being predicted indicating no need for classification of class. Various other evaluation parameter like precision, recall and mAP graphs are also plotted in Fig. 6. The precision and recall graph standout in an increasing trend which is a good sign for training of model. There are also several disturbances in the graphs which indicates the continuous training of the model. Other two mAP graphs also show an increasing trend in metric value. There also several fluctuations in graphs which results in low mAP value.

Second, YOLO v8 model was used for training, with subsequent

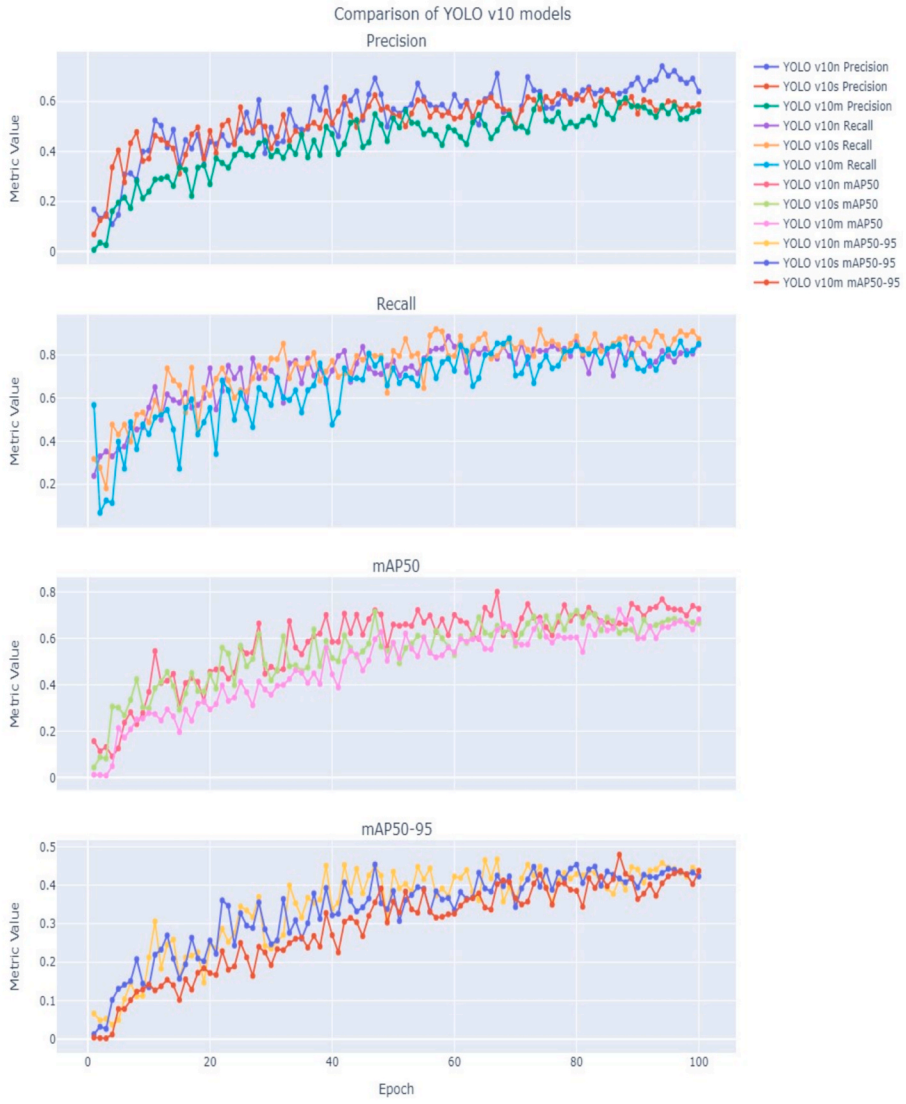

Fig. 12. Various parameters comparison for YOLO v10 models.

iterations across its nano, small, and medium variants using varying epoch values. Table 7 represents the accuracy and other evaluation metrics for the nano, small, and medium variants of the YOLO v8 model. The YOLO v8 nano model achieved its highest mAP50 of 81 % at 100 epochs, with precision and recall values of 74.20 % and 80.70 %, respectively. YOLO v8 small model reached its highest mAP50 of 81.50 % at 100 epochs, with precision and recall values of 71.90 % and 90.10 % respectively. Similarly, the YOLO v8 medium model attained its highest mAP50 of 81.90 % at 100 epochs. Overall, the YOLO v8 medium model achieved the highest mAP50, while the YOLO v8 nano model recorded the highest precision and recall values.

Fig. 7 shows the graph plotted between various evaluation parameters and epochs values for all three YOLO v8 model variants with 100 epochs values. The first graph is plotted between various epochs and mAP50 values. From the graph one can see that there are many disturbances which shows the learning of very model variants. The values in the graph shows increment in the values of mAP which shows convergence in the model training. The second graph is plotted between various precision and epochs values for all three variants. One can see that the precision values are incrementing at every epoch value which shows how the precision the model increases with increases in epochs value. The increment in the value shows how well the model has been trained on the train and validation set. The third graph is plotted between recall and epochs value. Similarly to precision and mAP50 values recall values increases with increase in epochs value as the model trains. The last graph represents the various mAP50-95 values with respect to epochs. The graph depicts the performance metrics of various YOLO v8 models across different epochs. However, based solely on this graph, it cannot be definitively concluded whether any particular YOLO v8 model is superior to others because it depends on average values.

Fig. 8 illustrates various graphs plotting between different loss values and epochs for the YOLO v8 nano model for the highest mAP50 value. Three distinct loss functions are represented: bounding box loss, classification loss, and distributed focal loss (DFL) loss. The bounding box loss indicates the error in predicting the bounding boxes in the images. The classification loss shows the error in classifying the objects within the bounding boxes, while the DFL loss measures the error in the distance between the true and predicted boxes in the images. The graphs reveal that the loss values for training are consistently lower compared to validation loss values. However, both training and validation loss values decrease as the number of epochs increases, demonstrating good convergence. From Fig. 8, it is evident that the classification loss values are the lowest among the three loss functions, likely due to the model being implemented for a single class. Moreover, it can be pragmatic that the box loss values for validation data are slowly diverging than training values due to which there is a gap formation between both curves. Also,

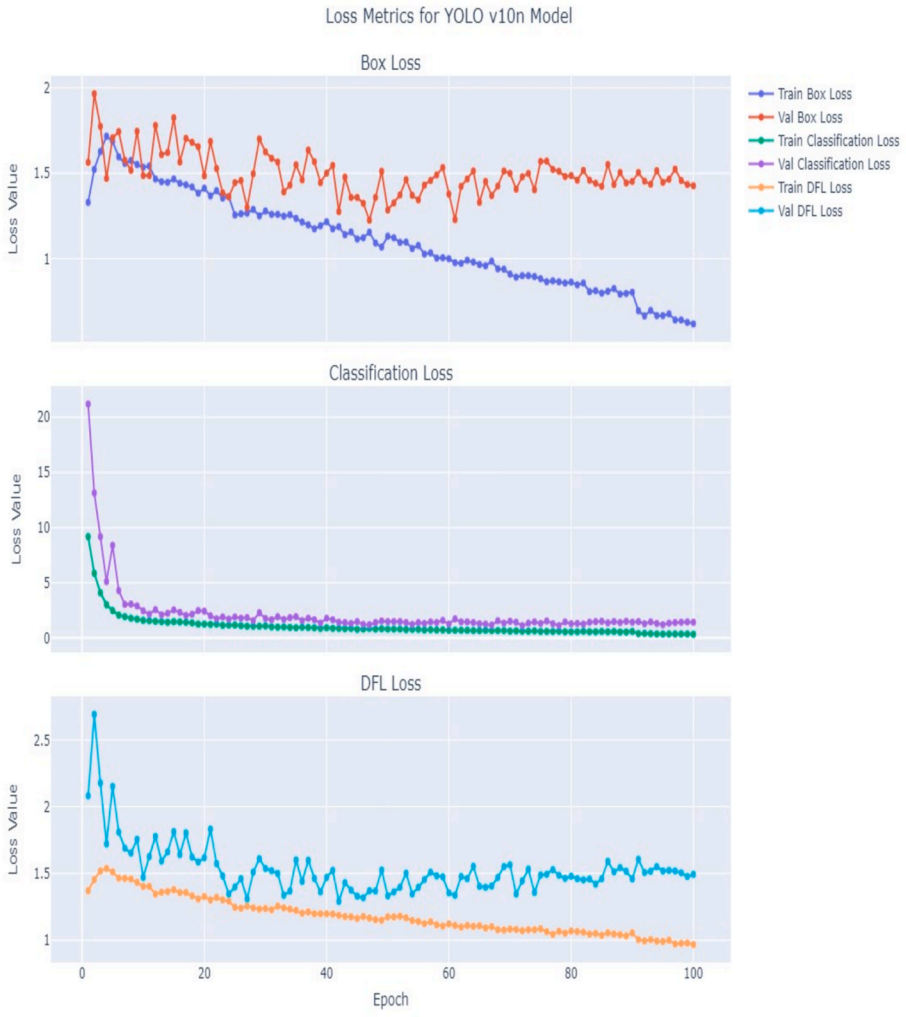

Fig. 13. Various Loss values comparison for YOLO v10n models.

some disturbances are observed in box and DFL loss in validation curve which shows that how model is learning at the time of learning or training.

Fig. 9 illustrates various graphs plotting for loss values and epoch values for YOLO v8 small variant. Like the YOLO v8 nano model, three different loss functions are utilized for evaluation. Observing the box loss graph, it's noticeable that the gap between validation and training loss values is similar to the nano variants and consistently decreases with increasing epoch values. From the plotting between DFL loss and epochs value it can be seen that there is a breach between training and validation loss values. Initially the validation loss value was very high which decreases with the increment in epochs values. Also, there are some disturbances in the validation curve but same is not observed in the training curve.

Fig. 10 illustrates the various graph plotted between various loss values and epochs values for YOLO v8 medium variant. From the graph plotted between box loss and epochs values it can be noticed that both validation and training loss are decreasing with the rise in epochs values. At the initial values of epochs both training and validation values have very small gap between them but as epochs values increases, they are diverging slowly. Similar to all the classification loss values for above both models' medium model has lowest value.

But on comparison to DFL loss values to small variant medium variants loss values seems to be more convergent as the gap between training and validation values are minimal. Hence, on comparison it can stated that the medium variant has good performance than above both model on basis of DFL loss convergence.

For the comparison purpose, YOLO v9c category model has been at various epochs value with auto optimizer mode for the detection task. Table 8 shows the results of YOLO v9 c model at various epochs values with their precision, recall and mAP values. From the table, it can be stated that the highest mAP50 value of 82.20 % has been achieved at the epoch value of 100 with the mean precision and recall value of 71.30 % and 85.20 % respectively. From the table, an increasing pattern in the mAP values can be observed which shows the convergence and good training of the model as with increase in epochs values all parameters are increasing slowly.

Fig. 11 shows the graphs plotted between various parameters like loss values, evaluation metrics and different epochs values for the highest mAP value combination. In Fig. 11 various loss functions trainings and validations graphs are plotted with epochs values. From the figure, the training curves of all loss functions is observed to be a smooth declining curve with some initial rise in values of box and DFL loss while the validation curve of box and DFL loss is observed to be a declining curve with some moderate variations in values at higher epochs values while classification loss curve appears to be a convergent decreasing curve with minimal disturbances amongst all. Other various evaluation parameters graphs are also been plotted as seen in figure. The recall and precision curve for the model appears to be inclining curve with increase in epochs values. There are several moderate fluctuations in both values however it manages out to achieve a high mAP value. From the values of mAP and precision it surpasses the above both

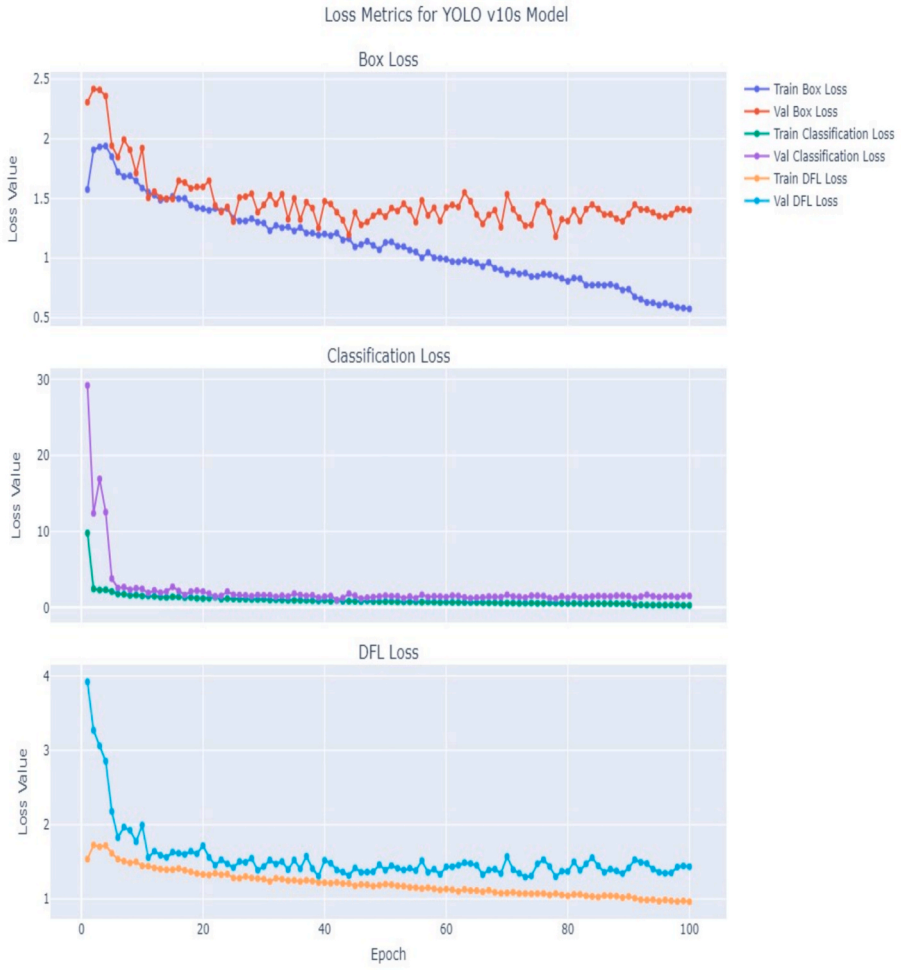

Fig. 14. Various Loss values comparison for YOLO v10s models.

models. There are also some fluctuations in mAP graphs but with high value it can be ignored.

Lastly, YOLO v10 model variants have been trained on the custom dataset. Three different variants namely nano, small and medium variants have been used for the comparison purpose. Table 6 shows results of various evaluations parameters like precision, recall and mAP with epochs values for all variants. From Table 9 it can be noticed that YOLO v10 nano variants has the highest mAP50 value of 80 % at 100 epochs with the precision and recall value of 74.50 % and 75 % respectively while medium and small variants have mAP50 value of 72.50 % and 71.70 % respectively at 100 epochs.

Fig. 12 shows graphs that compare the performance metrics of various YOLO v10 models over 100 epochs, illustrating their progression in key evaluation metrics. The Precision graph shows how accurately each model's positive predictions align with actual instances, revealing a general upward trend with occasional fluctuations, indicating that the models become more precise over time. The Recall graph highlights the models' ability to identify all relevant instances, with all models displaying an increasing trend, though with some variability, and stabilizing towards the end. The mAP50 graph, which evaluates the models' performance at a 50 % Intersection over Union (IoU) threshold, shows a similar positive trend, with the values increasing and then levelling off, signifying that the models achieve higher average precision as training progresses. The mAP50-95 graph provides a comprehensive measure, averaging precision across multiple IoU thresholds from 50 % to 95 %, and mirrors the trends seen in the other metrics, with improvements and eventual stabilization. Overall, these graphs collectively illustrate that the YOLO v10 models learn and adapt over the epochs, resulting in enhanced precision, recall, and mAP across different IoU thresholds, with performance gains stabilizing towards the later stages of training.

Fig. 13 shows the different graphs plotted between different loss functions and epochs values for the YOLO v10n model. In the Box loss graph the training curve seems to decreasing with the increase in epochs values while validation curve does not seem too convergent highly. The validation curve in the graph diverges way with the increase in epochs values with major disturbances yet it achieves the highest mAP50 value amongst all variants. Similarly, the DFL training loss values decreases with increase in epochs values also validation curve in the graph decreases highly but disturbances are more as compared to training curve. Also, some divergence is observed between both the curves but the gap is less than Box loss and also divergence is less. Moreover, a good convergence is seen in the classification loss because only one class is predicted.

Fig. 14 shows various graph plotted between different loss values and epochs for YOLO v10s model. The Box loss training curve increases initially but with the increase in number of epochs it decreases tremendously. A same pattern can be observed initially in the validation curve but it diverges at the end of training with a huge difference in both values. Hence, an open area can be seen at the end of the graph. Similar to all other models and variants discussed above classification loss decreases heavily and converges with some initial high oscillations in the initial phase of epochs. The DFL training curve also follows the same pattern with some increment in initial phase with decrement in the validation loss values. The validation curve shows a constant decrement

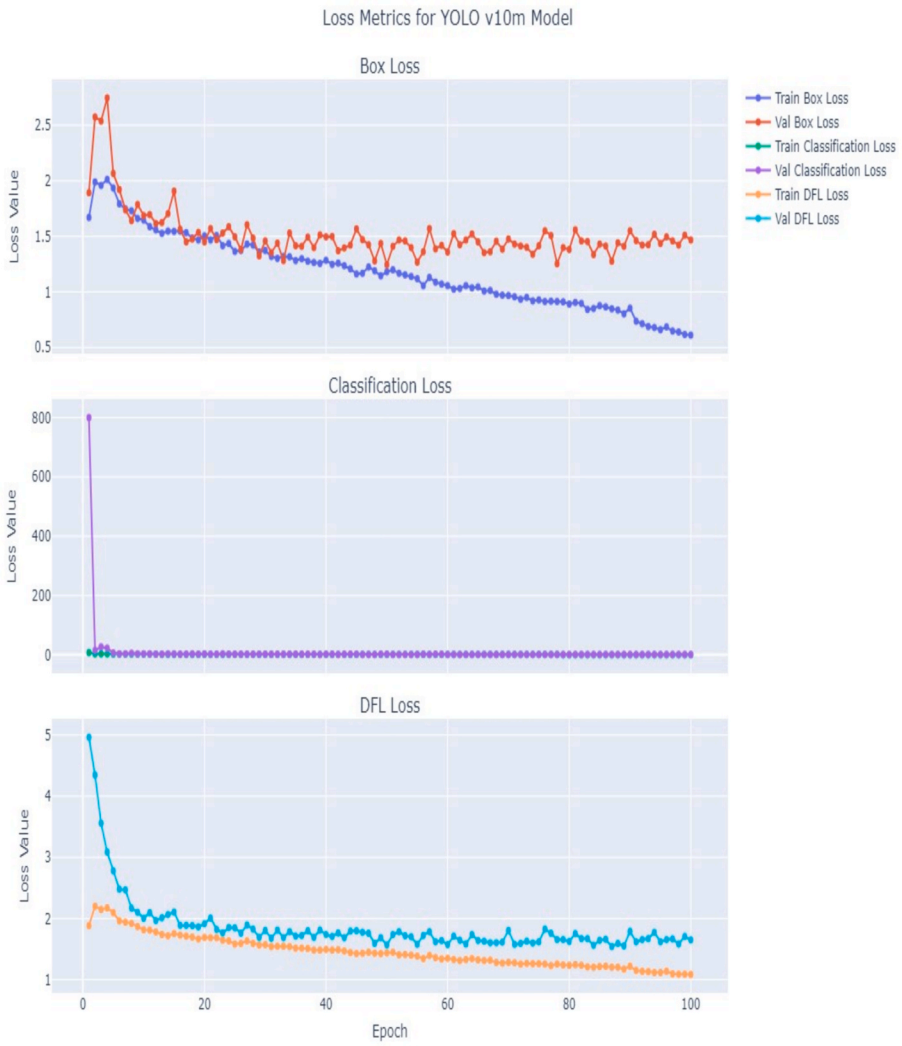

Fig. 15. Various Loss values comparison for YOLO v10m models.

Table 10
Comparison Table for all YOLO models.

Name	Epochs	Precision (in %)	Recall (in%)	mAP50 (in%)	mAP50-95 (in%)
YOLO v5	100	62.00	79.50	72.60	46.20
YOLO v8 medium	100	73.20	78.40	81.90	51.20
YOLO v9c	**100**	**71.30**	**82.50**	**82.20**	**50.80**
YOLO v10 nano	100	74.50	75.00	80.00	46.70

in values with some disturbances. Also, the difference between both the values is minor which shows good training and convergence sign for predictions.

Fig. 15 shows the graph plotted between various loss values and epochs for the YOLO v10m variant. Similar to above YOLO v10 variants training Box loss curve increases at initial phase and then starts to decreases with increase in epochs values. A huge difference between validation and training loss values can be observed with increase in epochs values but compared to other variants DFL loss curves shows more convergence and also the small disturbances are observed as compared to other variants. Also, a constant value can be observed in the classification loss curves.

Table 10 show the comparison of all the models with mAP and other metrics used for the comparison. The highest mAP values from each model have been chosen for the table. As a result, from YOLO v8 and v10 model the variant with highest mAP has been chosen for the comparison purpose. Based on the table and results from various YOLO models, it can be stated that YOLO v9c model achieved the maximum mAP50 of 82.20 % compared to other variants and models. Moreover, YOLO v8 medium variant model demonstrated strong performance on both validation and training datasets, achieving an mAP50 of 81.90 %. Despite being a newer and faster algorithm, YOLO v10 did not attain a high mAP value on the training set. However, the overall performance of any model cannot be definitively assessed without considering various internal and external factors.

In addition to traditional metrics like precision, recall, and mAP, other visualizations play a crucial role in assessing object detection tasks. Among these, the recall-confidence and precision-confidence curves are particularly valuable. In object detection tasks, it's vital not only to detect objects accurately but also to assess the confidence level associated with each detection. The recall-confidence curve illustrates the trade-off between recall, which measures the proportion of true positives that are correctly identified, and confidence, representing the certainty of the detection. This curve provides insights into the system's ability to detect objects across different confidence thresholds. A high recall indicates that the system successfully identifies most of the relevant objects, while a high confidence suggests a low likelihood of false positives. The precision-confidence curve is a significant tool for

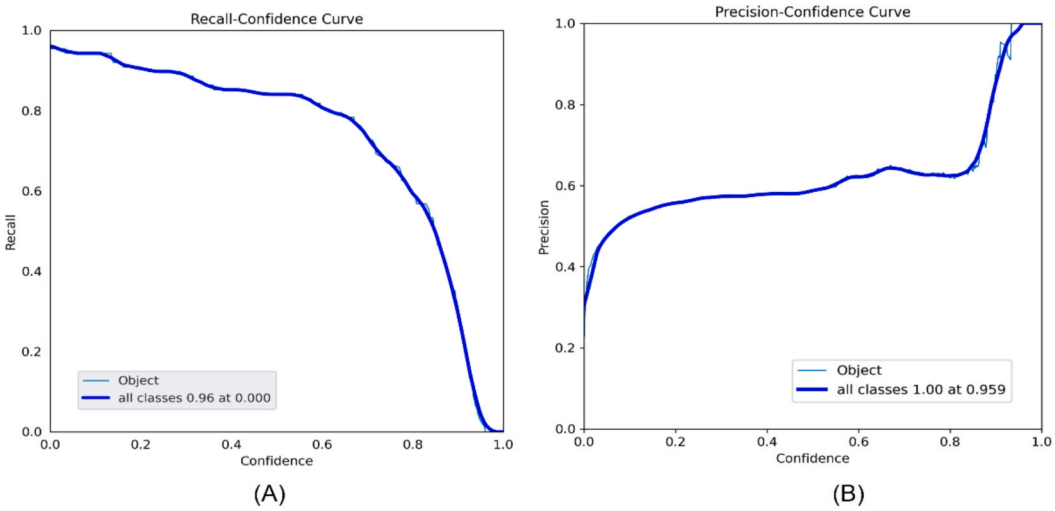

Fig. 16. (A) Recall-Confidence graph and (B) Precision-Confidence graph for YOLO v5 model.

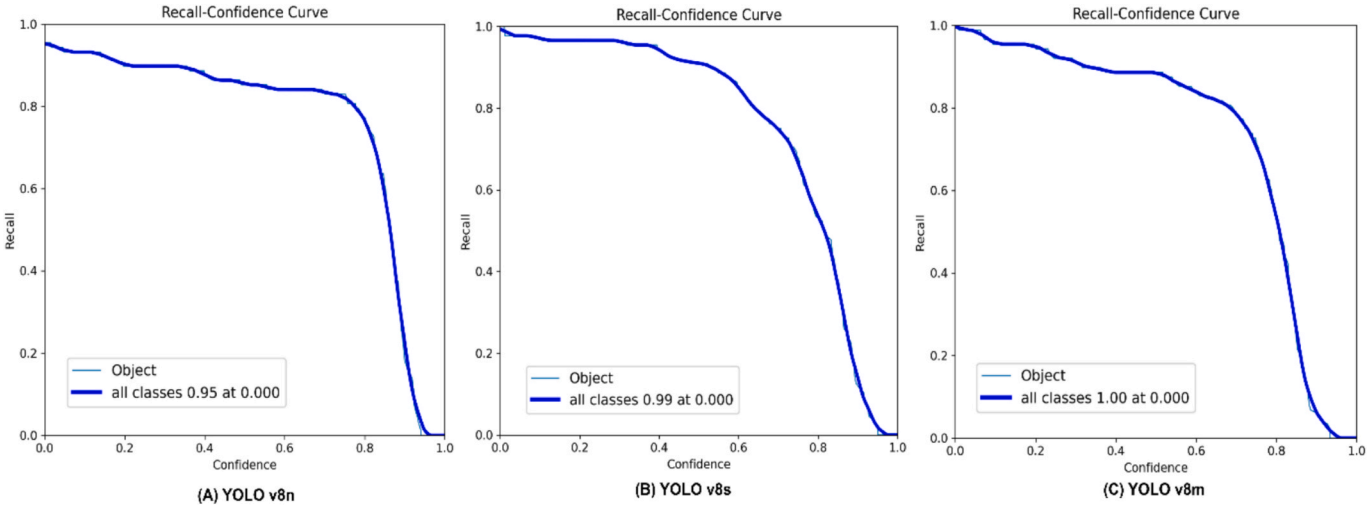

Fig. 17. Recall-Confidence Curve for YOLO v8 variants.

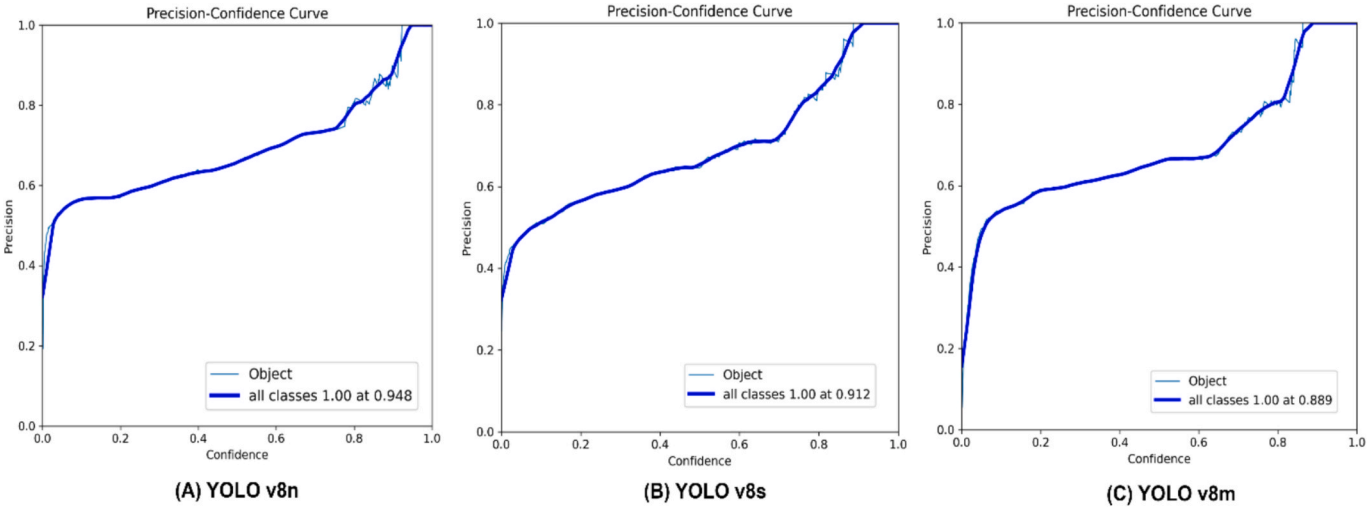

Fig. 18. Precision-Confidence Curve for YOLO v8 variants.

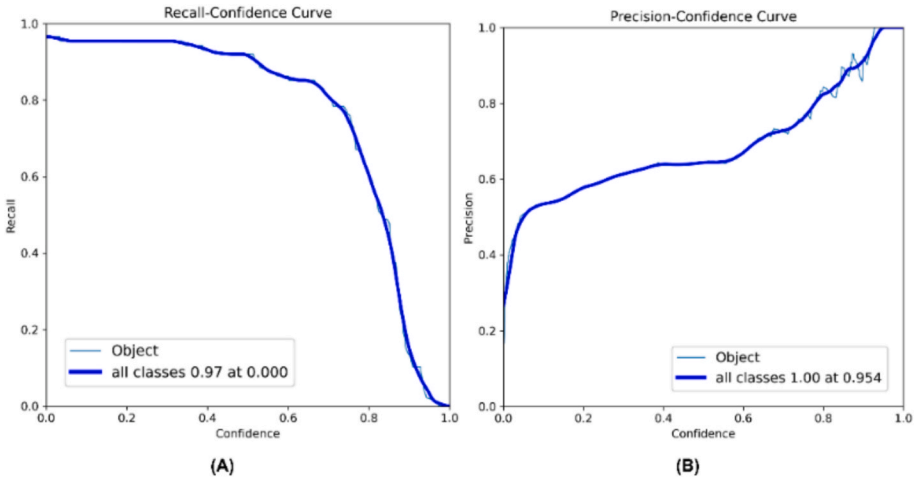

Fig. 19. (A) Recall-Confidence graph and (B) Precision-Confidence graph for YOLO v9c model.

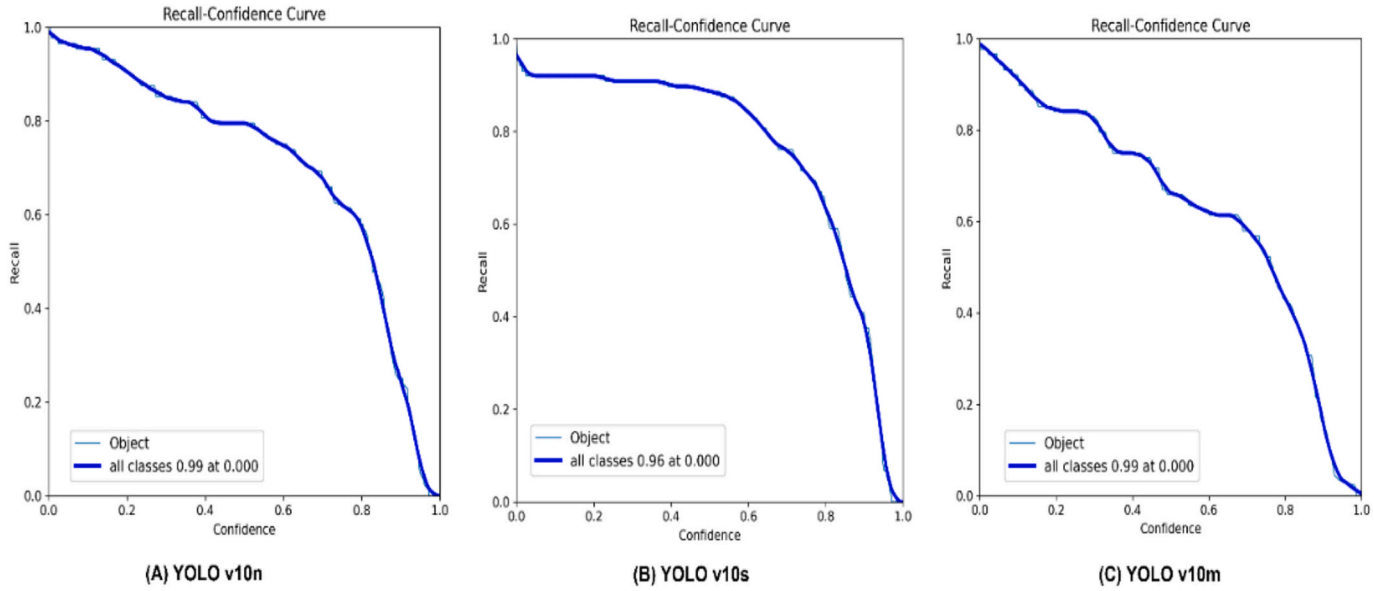

Fig. 20. Recall-Confidence Curve for YOLO v10 variants.

evaluating the performance of object detection systems. The precision-confidence curve plots precision against varying confidence thresholds, revealing how precision changes as the confidence requirement for detections is adjusted.

Fig. 16 (A) and (B) shows the recall-confidence and precision-confidence graph for the YOLO v5 model. From the graph it can be stated that initially the recall value for the model was very high but with the increase in epochs, it decreases while same but a reverse trend is observed in precision-confidence graph. From the graphs, it can be observed that the highest recall of 0.96 is observed at confidence score of 0.00 while highest precision value of 1.00 at confidence score of 0.95.

Fig. 17 shows recall-confidence graph plotted for variants of YOLO v8 model. As a regular trend initially, it shows high values which decreases slowly with the rise in confidence value because rise in score decrements the predictions of model. From figure, it can be said that the highest recall value for nano variant is 0.95 while for small and medium model is 0.99 and 1 respectively at confidence score of 0.

Fig. 18 shows the precision-confidence graph for YOLO v8 variants. A high precision at lower confidence levels indicates that the model is accurately identifying objects even when it is less certain, while a decline in precision at higher confidence levels can indicate a higher rate of false positives. By examining this curve, developers can identify the confidence threshold that maximizes precision, thus fine-tuning the detection algorithm to reduce false positives while maintaining high accuracy. From the figure, it can be stated that highest precision of 1.00 is achieved at the confidence of 0.94, 0.91 and 0.88 for nano, small and medium variants respectively.

Fig. 19 (A) and (B) shows precision-confidence and recall-confidence graph plotted for YOLO v9c model respectively. The recall slowly decreases with the increase in confidence score while an increment pattern can be seen in precision-confidence graph with increase in score. From the graph it can be seen that the maximum recall of 0.97 is achieved at the confidence score of 0.00 while maximum precision of 1.00 is achieved at the confidence score of 0.95.

Fig. 20 shows the recall-confidence graph plotted for various YOLO v10 variants. Similar decreasing trend is observed from a high recall like other models. Form the figure, it can state that YOLO v10 nano and medium variant has achieved the highest recall value of 0.99 at the confidence score of 0.00 while small variant has achieved 0.96. Fig. 21 shows the precision-confidence graph plotted for the YOLO v20 variants. Like other variants and models a similar upward trend is observed in the graphs. From the graphs values it can be observed that from all variants

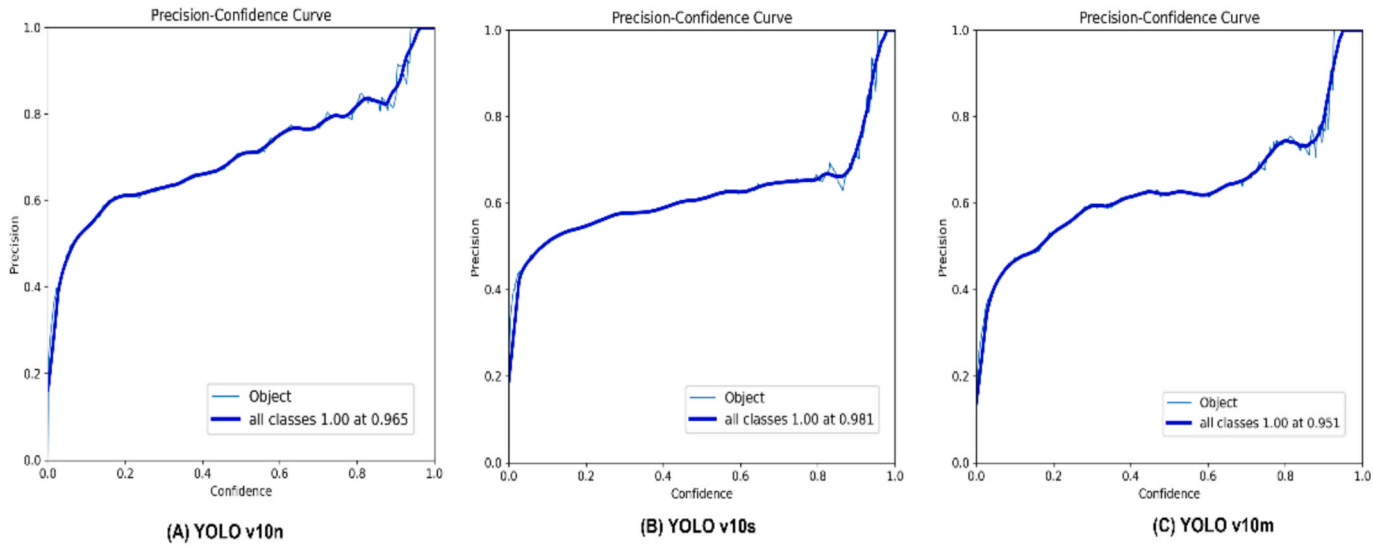

Fig. 21. Precision-Confidence Curve for YOLO v10 variants.

Fig. 22. Actual and Predicted labels for YOLO v9c model.

precision value of 1.00 is achieved at the lowest confidence score of 95.1 by medium variant while highest confidence score is 98.1 by small variant.

Fig. 22 shows various actual and predicted images for YOLO v9c model. From the images it can be seen that how well the model predicts. It can also be seen that some images with background or no objects are also there and also model predicts it correctly. This shows the well training of the model.

Table 11 presents the results of the ANOVA test conducted to evaluate the effect of training epochs on the performance of different YOLO models. ANOVA is a statistical method used to legalize whether there is significant variance between group means by comparing variance between groups (systematic variance) and within groups (random error variance). A higher F-ratio and a lower p-value (< 0.05) indicate that the factor (training epochs) significantly impacts model performance. The ANOVA test results confirm that training epochs significantly impact the

Table 11
ANOVA Results for Each YOLO Model.

Name	Degree of Freedom (DF)	F-ratio	p-value	η^2
YOLO v5	3	9.08	0.002	3.03
YOLO v8 nano	3	8.28	0.007	2.09
YOLO v8 small	3	9.2	0.002	3.07
YOLO v8 medium	3	10.98	0.001	3.66
YOLO v9c	**3**	**11.84**	**0.001**	**3.95**
YOLO v10 nano	3	5.19	0	2.06
YOLO v10 small	3	8.58	0	3.19
YOLO v10 medium	3	6.32	0.008	2.11

Table 12
Performance Comparison of YOLO on COCO, KITTI, and BDD100K.

Dataset	Model	Precision (%)	Recall (%)	mAP50	mAP50-95 (%)
COCO	YOLOv5	54.52	68.41	62.45	33.12
	YOLOv8 (Medium)	67.98	72.15	74.39	45.12
	YOLO v9c	70.21	74.66	75.33	46.72
	YOLOv10 (Nano)	66.48	68.23	70.12	40.11
KITTI	YOLOv5	62.31	70.15	67.23	38.12
	YOLOv8 (Medium)	70.91	75.33	76.12	44.89
	YOLO v9c	72.33	71.1	78.44	45.23
	YOLOv10 (Nano)	65.01	69.45	72.15	40.11
BDD100K	YOLOv5	58.94	67.82	62.3	32.74
	YOLOv8 (Medium)	66.23	72.66	73.18	42.16
	YOLO v9c	70.16	74.16	76.44	44.39
	YOLOv10 (Nano)	62.5	68.78	69.04	38.56

performance of all YOLO models, as indicated by low p-values (≤ 0.007) and high F-ratios. YOLO v9c ($F = 11.84$, $\eta^2 = 3.95$), YOLO v8 medium ($F = 10.98$, $\eta^2 = 3.66$), and YOLO v10 small ($F = 8.58$, $\eta^2 = 3.19$) exhibit the strongest response to increased training, suggesting that additional epochs improve their performance effectively. Models like YOLO v10 nano ($F = 5.19$, $\eta^2 = 2.06$) and YOLO v10 medium ($F = 6.32$, $\eta^2 = 2.11$) show a weaker impact of training, indicating that factors beyond epochs, such as hyper-parameter tuning or data augmentation, may play a role. These findings highlight the varying sensitivity of YOLO models to training duration, emphasizing the need for model-specific optimization strategies.

Table 12 below summarizes the estimated performance of several YOLO models on three widely recognized benchmark datasets COCO, KITTI, and BDD100K. Due to the larger scale, greater object diversity, and increased scene complexity in datasets like COCO and BDD100K, these models generally achieve slightly lower scores on them compared to KITTI, which contains more constrained and domain-specific driving scenarios. Among the models, YOLOv8 and YOLOv9c consistently show strong generalization capabilities, especially in complex urban and traffic environments. In contrast, YOLOv10-nano, while lighter and more efficient, delivers relatively lower accuracy, making it better suited for real-time or resource-constrained applications.

6. Conclusion

The study demonstrates the performance of various YOLO models like YOLO v5, YOLO v8, YOLO v9, and YOLO v10 in object detection tasks, revealing critical insights into their capabilities and limitations. The YOLO v9c model achieved the highest mAP50 of 82.20 %, indicating superior performance compared to other YOLO variants. Analysis of precision-confidence and recall-confidence curves shows the YOLO v9c model maintaining a high recall of 0.97 and achieving a precision of 1.00 at a confidence score of 0.95, indicating robustness in maintaining high detection accuracy even as confidence levels vary. In summary, the YOLO v9c model stands out for its overall performance making them suitable for different application scenarios depending on specific precision and recall requirements.

In future work, the research plan to significantly expand our dataset to include a more diverse and extensive collection of images. This will encompass a wider range of object classes, occlusion scenarios, and lighting conditions, aiming to improve the model's generalization and robustness. Alongside with dataset expansion, the study will further analyse failure cases observed during the current study, particularly in detecting small objects, handling heavy occlusions, and managing varying illumination. Insights from these analyses will guide architectural refinements and training strategy adjustments to enhance model performance in challenging scenarios. Moreover, the study also intends to deploy the trained models on the hardware platform discussed in this study to develop a real-time object detection and avoidance system. This deployment will enable validation of the model's performance in practical, dynamic environments. To further support real-time capability and edge deployment, the research will further also explores model compression techniques such as pruning and quantization to reduce computational load while maintaining accuracy. These combined efforts will help address the current limitations and strengthen the model's applicability in real-world robotics and embedded systems.

7. Contributions

All the authors contributed equally and confirm sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

8. Consent to participate

Not applicable' as the studies do not involve humans.

Ethical approval

Not applicable.

CRediT authorship contribution statement

Rohan Vaghela: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Formal analysis, Conceptualization. **Darsh Vaishnani:** Writing – review & editing, Writing – original draft, Visualization, Methodology, Investigation, Data curation, Conceptualization. **Jigar Sarda:** Writing – review & editing, Writing – original draft, Validation, Supervision, Conceptualization. **Amit Thakkar:** Writing – review & editing, Writing – original draft, Supervision. **Yash Nasit:** Writing – review & editing. **Biswajit Brahma:** Writing – review & editing. **Akash Kumar Bhoi:** Writing – review & editing, Validation.

Funding

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank the Computer Science Engineering Department of

Chandubhai S Patel Institute of Technology, CHARUSAT for providing the infrastructure facilities for conducting this study.

Data availability

Data will be made available on request.

References

[1] S. Kuutti, R. Bowden, Y. Jin, P. Barber, S. Fallah, A survey of deep learning applications to autonomous vehicle control, IEEE Trans. Intell. Transp. Syst. (2020).

[2] B. Mahaur, N. Singh, K.K. Mishra, Road object detection: a comparative study of deep learning-based algorithms, Multimed. Tools Appl. 81 (2022) 14247–14282, https://doi.org/10.1007/s11042-022-12447-5.

[3] S. Sivaraman, M.M. Trivedi, Looking at vehicles on the road: a survey of vision-based vehicle detection, tracking, and behavior analysis, IEEE Trans. Intell. Transp. Syst. 14 (4) (2013) 1773–1795.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.Y. Fu, A.C. Berg, SSD: Single shot multibox detector, in: Proceedings of the Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, 11–14 October 2016; Proceedings, Part I 14. Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 21–37.

[5] T.Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense object detection, in: In: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2980–2988.

[6] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.

[7] P. Purkait, C. Zhao, C. Zach, SPP-Net: Deep absolute pose regression with synthetic views. arXiv 2017, arXiv:1712.03452.

[8] G. Gkioxari, B. Hariharan, R. Girshick, Malik, J. R-CNNs for pose estimation and action detection. arXiv 2014, arXiv:1406.5212.

[9] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, in: Proceedings of the Advances in Neural Information Processing Systems 28 (NIPS 2015), Montreal, QC, Canada, 7–12 December 2015.

[10] H. Chen, Z. Chen, Yu. Hang, Enhanced YOLOv5: an efficient road object detection method, Sensors 23 (20) (2023) 8355, https://doi.org/10.3390/s23208355.

[11] P. Jiang, D. Ergu, F. Liu, Y. Cai, B.o. Ma, A review of Yolo algorithm developments, Procedia Comput. Sci. 199 (2022) 1066–1073.

[12] Peiyuan Jiang, Daji Ergu, Fangyao Liu, Ying Cai, Bo Ma, A review of Yolo algorithm developments procedia computer science, Volume 199, 2022, Pages 1066-1073, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2022.01.135.

[13] W. Yang, X. Tang, K. Jiang, Fu. Yang, X. Zhang, An improved YOLOv5 algorithm for vulnerable road user detection, Sensors 23 (18) (2023) 7761.

[14] M.L. Ali, Z. Zhang, The YOLO framework: a comprehensive review of evolution, applications, and benchmarks in object detection, Computers 13 (2024) 336, https://doi.org/10.3390/computers13120336.

[15] Rane, Nitin, YOLO and Faster R-CNN object detection for smart Industry 4.0 and Industry 5.0: applications, challenges, and opportunities (October 25, 2023). Available at SSRN: https://ssrn.com/abstract=4624206 or https://doi.org/10.2139/ssrn.4624206.

[16] E. Li, Q. Wang, J. Zhang, W. Zhang, H. Mo, Y. Wu, Fish Detection under occlusion using modified you only look once v8 integrating real-time detection transformer features, Appl. Sci. 13 (2023) 12645.

[17] X. Wang, Z. Wu, M. Jia, T. Xu, C. Pan, X. Qi, M. Zhao, Lightweight SM-YOLOv5 tomato fruit detection algorithm for plant factory, Sensors 23 (6) (2023) 3336, https://doi.org/10.3390/s23063336.

[18] T. Zeng, S. Li, Q. Song, F. Zhong, Lightweight tomato real-time detection method based on improved YOLO and mobile deployment, Comput. Electron. Agric. 205 (2023) 107625, https://doi.org/10.1016/j.compag.2023.107625.

[19] F. Aziz, D.U.E. Saputri, Efficient skin lesion detection using YOLOv9 network, J. Med. Informat. Technol. (2024) 11–15.

[20] Y. Zou, J. Wan, B. Wang, Lightweight pothole detection method based on improved YOLOv5, in: 2023 IEEE 3rd International Conference on Information Technology, Big Data and Artificial Intelligence (ICIBA) (Vol. 3, pp. 631-635). IEEE.

[21] X. Yue, K. Qi, F. Yang, X. Na, Y. Liu, C. Liu, RSR-YOLO: a real-time method for small target tomato detection based on improved YOLOv8 network, Discover Appl. Sci. 6 (5) (2024) 268.

[22] M. Sathvik, G. Saranya and S. Karpagaselvi, An intelligent convolutional neural network based potholes detection using Yolo-V7, in: 2022 International Conference on Automation, Computing and Renewable Systems (ICACRS), Pudukkottai, India, 2022, pp. 813-819, doi: https://doi.org/10.1109/ICACRS55517.2022.10029263.

[23] Vo, Hoang-Tu, Kheo Chau Mui, Nhon Nguyen Thien, and Phuc Pham Tien, Automating tomato ripeness classification and counting with YOLOv9, Int. J. Adv. Comput. Sci. Appl. 15, no. 4 (2024).

[24] R. Mirajkar, A. Yenkikar, S. Nawalkar, R. Kaul, A. Rokade, K. Rothe, Enhanced pothole detection in road condition assessment using YOLOv8, in: In 2024 IEEE International Conference for Women in Innovation, Technology & Entrepreneurship (ICWITE), 2024, pp. 429–433.

[25] T. Wu, Y. Dong, YOLO-SE: improved YOLOv8 for remote sensing object detection and recognition, Appl. Sci. 13 (24) (2023) 12977.

[26] X. Jia, Y. Tong, H. Qiao, M. Li, J. Tong, B. Liang, Fast and accurate object detector for autonomous driving based on improved YOLOv5, Sci. Rep. 13 (1) (2023) 9711.

[27] S.N. Appe, G. Arulselvi, G.N. Balaji, CAM-YOLO: tomato detection and classification based on improved YOLOv5 using combining atten tion mechanism, PeerJ Comput. Sci. 9 (2023) e1463.

[28] S. Fei, Z. Zexu, Z. Yanping, L. Tianhua, Z. Linlu, Detection of mature green tomato based on lightweight YOLO-v3, J Chin Agric Mech. 43 (3) (2022) 132.

[29] G. Yang, J. Wang, Z. Nie, H. Yang, S. Yu, a Lightweight YOLOv8 tomato detection algorithm combining feature enhancement and attention, Agronomy 13 (7) (2023) 1824, https://doi.org/10.3390/agronomy13071824.

[30] E. Ranyal, A. Sadhu, K. Jain, AI assisted pothole detection and depth estimation, in: 2023 International Conference on Machine Intelligence for GeoAnalytics and Remote Sensing (MIGARS), Hyderabad, India, 2023, pp. 1-4, doi: https://doi.org/10.1109/MIGARS57353.2023.10064547.

[31] C. Gao, Q. Zhang, Z. Tan, G. Zhao, S. Gao, E. Kim, T. Shen, Applying optimized YOLOv8 for heritage conservation: enhanced object detection in Jiangnan traditional private gardens, Heritage Sci. 12 (1) (2024) 31.

[32] Q. Zeng, G. Zhou, L. Wan, L. Wang, G. Xuan, Y. Shao, Detection of coal and Gangue based on improved YOLOv8, Sensors 24 (4) (2024) 1246.

[33] Lizhao Liu, Pinrui Li, Dahan Wang, Shunzhi Zhu, A wind turbine damage detection algorithm designed based on YOLOv8, Applied Soft Computing, Volume 154, 2024, 111364, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2024.111364.

[34] X. Jiang, X. Zhuang, J. Chen, J. Zhang, Y. Zhang, YOLOv8-MU: an improved YOLOv8 underwater detector based on a large kernel block and a multi-branch reparameterization module, Sensors 24 (9) (2024) 2905.

[35] Haiping Ma, Yajing Zhang, Shengyi Sun, Weijia Zhang, Minrui Fei, Huiyu Zhou, Weighted multi-error information entropy based you only look once network for underwater object detection, Eng. Appl. Artificial Intelligence, Volume 130, 2024, 107766, ISSN 0952-1976, https://doi.org/10.1016/j.engappai.2023.107766.

[36] S. K. Rout, K. S. Kumar, R. Sahu, S. Barik and S. Pradhan, "Tropical Cyclone Detection and Tracking Using YOLOv8 Algorithm," 2024 1st International Conference on Cognitive, Green and Ubiquitous Computing (IC-CGU), Bhubaneswar, India, 2024, pp. 1-6, doi: 10.1109/IC-CGU58078.2024.10530785.

[37] J. Li, F. Tang, C. Zhu, S. He, S. Zhang, Y. Su, BP-YOLO: a Real-time product detection and shopping behaviors recognition model for intelligent unmanned vending machine, IEEE Access 12 (2024) 21038–21051, https://doi.org/10.1109/ACCESS.2024.3361675.

[38] S. Xie, M. Zhou, C. Wang, S. Huang, CSPPartial-YOLO: a lightweight YOLO-based method for typical objects detection in remote sensing images, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 17 (2024) 388–399, https://doi.org/10.1109/JSTARS.2023.3329235.

[39] A. Ahmed, A.S. Imran, A. Manaf, Z. Kastrati, S.M. Daudpota, Enhancing wrist abnormality detection with YOLO: Analysis of state-of-the-art single-stage detection models, Biomed. Signal Process. Control 93 (2024), https://doi.org/10.1016/j.bspc.2024.106144.

[40] H. Li, Wu. Dengchao, W. Zhang, C. Xiao, YOLO-PL: helmet wearing detection algorithm based on improved YOLOv4, Digital Signal Process. 144 (2024), https://doi.org/10.1016/j.dsp.2023.104283.

[41] Y. Zhou, A YOLO-NL object detector for real-time detection, Expert Syst. Appl. 238 (Part E) (2024), https://doi.org/10.1016/j.eswa.2023.122256.

[42] M. Srihitha, S. Sricharan, D. Akshitha, M. Yesubabu, K.S. Kumar, S.K. Rout, Deep learning for pediatric dermatology: melanoma detection in child health using YOLOv8-MNC algorithm, in: Simic, M., Bhateja, V., Azar, A.T., Lydia, E.L. (Eds.) Smart Computing Paradigms: Advanced Data Mining and Analytics. SCI 2024. Lecture Notes in Networks and Systems, vol 1262. Springer, Singapore, 2025. https://doi.org/10.1007/978-981-96-1981-8_33.

[43] S. K. Rout, K. S. Kumar, R. Sahu, S. Barik and S. Pradhan, Tropical Cyclone Detection and Tracking Using YOLOv8 Algorithm, in: 2024 1st International Conference on Cognitive, Green and Ubiquitous Computing (IC-CGU), Bhubaneswar, India, 2024, pp. 1-6, doi: https://doi.org/10.1109/IC-CGU58078.2024.10530785.

[44] C. Anusha, K. S. Sanjay, S. Deekshitha, S. K. Rout, K. S. Kumar and S. Ranjith Reddy, Super Cyclone Detection and Tracking Using YOLOv8 Algorithm, in: 2024 International Conference on Emerging Systems and Intelligent Computing (ESIC), Bhubaneswar, India, 2024, pp. 255-260, doi: https://doi.org/10.1109/ESIC60604.2024.10481667.

[45] P. Mehta, R. Vaghela, N. Pansuriya, J. Sarda, N. Bhatt, A.K. Bhoi, P.N. Srinivasu, Benchmarking YOLO variants for enhanced blood cell detection, Int. J. Imaging Syst. Technol. 35 (1) (2025) 70037.

[46] R. Vaghela, J. Sarda, A. Thakkar, D. Vaishnani, A. Khokhariya and P. Mehta, Brain tumor detection using YOLO v7 and YOLO v8, in: 2024 International Conference on Modeling, Simulation & Intelligent Computing (MoSICom), Dubai, United Arab Emirates, 2024, pp. 160-165, doi: https://doi.org/10.1109/MoSICom63082.2024.10882136.

[47] M.Ş. Gündüz, G. Işık, A new YOLO-based method for social distancing from real-time videos, Neural Comput. Applic. 35 (2023) 15261–15271.

[48] M.Ş. Gündüz, G. Işık, A new YOLO-based method for real-time crowd detection from video and performance analysis of YOLO models, J. Real-Time Image Proc. 20 (2023) 5.

[49] Dataset creation website: https://roboflow.com/.

[50] G. Jocher, YOLOv5 by Ultralytics. 2020. Available online: https://github.com/ultralytics/yolov5 (accessed on 28 February 2023).

[51] J. Redmon, A. Farhadi, Yolov3: an incremental improvement, 2018. arXiv: 180402767.

[52] G. Jocher, YOLOv8 by Ultralytics. 2022. Available online: https://github.com/ultralytics/ultralytics.
[53] Wang, Chien-Yao, I-Hau Yeh, and Hong-Yuan Mark Liao, YOLOv9: Learning what you want to learn using programmable gradient information, arXiv preprint arXiv: 2402.13616 (2024).
[54] Wang, Ao, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding, YOLOv10: real-time end-to-end object detection, arXiv preprint arXiv: 2405.14458 (2024).