IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 4, August 2026, pp. 3581~3590
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i4.pp3581-3590

Real-time object detection for autonomous driving: a comparative study of YOLO and Faster R-CNN

Madhura M. Bhosale[1], Yogesh S. Angal[2]
[1]Department of Electronics and Telecommunication Engineering, JSPM's Rajarshi Shahu College of Engineering, Pune, India
[2]Department of Electronics and Telecommunication Engineering, JSPM's Bhivarabai Sawant Institute of Technology and Research, Pune, India

Article Info

Article history:

Received Jun 30, 2025
Revised May 13, 2026
Accepted May 24, 2026

Keywords:

Autonomous driving
Computer vision
Deep learning
Machine learning
Object detection
Optimizer

ABSTRACT

Over the past few years, object detection has experienced remarkable progress and development, primarily driven by the development of one-stage and two-stage detection algorithms. Among these, Faster region-based convolutional neural network (Faster R-CNN) and you only look once (YOLO) have achieved notable success due to their strong performance and computational efficiency. Object detection plays a crucial role in various applications, particularly in autonomous driving systems, where accurate detection of pedestrians, vehicles, and road signs is essential for ensuring safety and reliability. This paper conducts a comparative evaluation of YOLO and Faster R-CNN to analyze their performance in autonomous driving environments. The experiments were conducted using the KITTI open-source dataset, which is widely used for benchmarking object detection models. All experiments were performed on an NVIDIA RTX A5000 GPU to ensure efficient computation, with implementations developed using Python version 3.9.13. The experimental findings indicate that YOLO surpasses Faster R-CNN in performance, attaining an accuracy rate of 90%. These findings highlight the effectiveness of YOLO for real-time object detection tasks, making it a suitable and preferred choice for time-sensitive applications such as autonomous driving systems.

Corresponding Author:

Madhura M. Bhosale
Department of Electronics and Telecommunication Engineering
JSPM's Rajarshi Shahu College of Engineering
Pune 411033, India
Email: Madhurabhosale4@gmail.com

1. INTRODUCTION

Object detection is a core task in computer vision that involves identifying and localizing objects within images or video frames using bounding boxes. Unlike image classification, which only determines object presence, object detection provides precise spatial localization, enabling applications in autonomous driving, video surveillance, medical imaging, and industrial automation. Early object detection approaches relied on handcrafted features and traditional machine learning techniques, such as Viola and Jones [1], histograms of oriented gradients (HOG) [2], deformable part models (DPM) [3], selective search [4], and scale-invariant feature descriptors [5]. Although effective for constrained scenarios, these methods lacked robustness and scalability for complex real-world environments. The introduction of deep learning and convolutional neural networks (CNNs) marked a significant shift in object detection research, enabling automatic feature learning and improved generalization [6]. This led to the development of region-based convolutional neural network (R-CNN) frameworks, including Fast R-CNN [7] and Faster R-CNN [8], where

region proposal networks (RPN) enabled end-to-end training and significantly improved detection accuracy. Faster R-CNN and its extensions, such as Cascade R-CNN, are categorized as two-stage detectors and are known for high localization precision, with increased computational cost [8], [9].

In parallel, one-stage detectors emerged to address real-time requirements by unifying localization and classification into a single network. You only look once (YOLO) framework introduced regression-based formulation that processes the entire image in one forward pass, enabling real-time object detection [9]. Subsequent developments, including anchor-free and lightweight variants, further improved efficiency and performance across diverse applications. YOLO-based approaches have demonstrated strong results in autonomous driving, edge computing, sensor fusion, and aerial image analysis [10]–[17].

Several studies have conducted comparative evaluations of modern object detection algorithms, highlighting the trade-offs between accuracy and inference speed among YOLO, single shot multibox detector (SSD), and Faster R-CNN–based models [18]–[23]. These findings emphasize practical importance of selecting detection frameworks based on application constraints. Motivated by these observations, this paper presents a comparative study of YOLO and Faster R-CNN using the KITTI dataset [24]–[26]. Experiments are conducted on high-performance hardware equipped with an NVIDIA RTX A5000 GPU to evaluate detection accuracy, inference speed, and computational efficiency. The objective is to provide practical insights into the suitability of one-stage and two-stage detectors for real-time and accuracy-critical computer vision applications.

2. OBJECT DETECTION FRAMEWORKS

2.1. Faster region-based convolutional neural network

Faster R-CNN, introduced by Ren *et al*. [8] in 2015, represented a major step forward in object detection by proposing a unified framework that can be trained in an end-to-end manner. In contrast to earlier approaches such as R-CNN and Fast R-CNN, which depended on separate external region proposal techniques, Faster R-CNN integrates the RPN directly into its overall architecture. This integration eliminated the bottleneck of generating region proposals separately, leading to a substantial improvement in speed and efficiency. Figure 1 illustrates the architecture of Faster R-CNN.

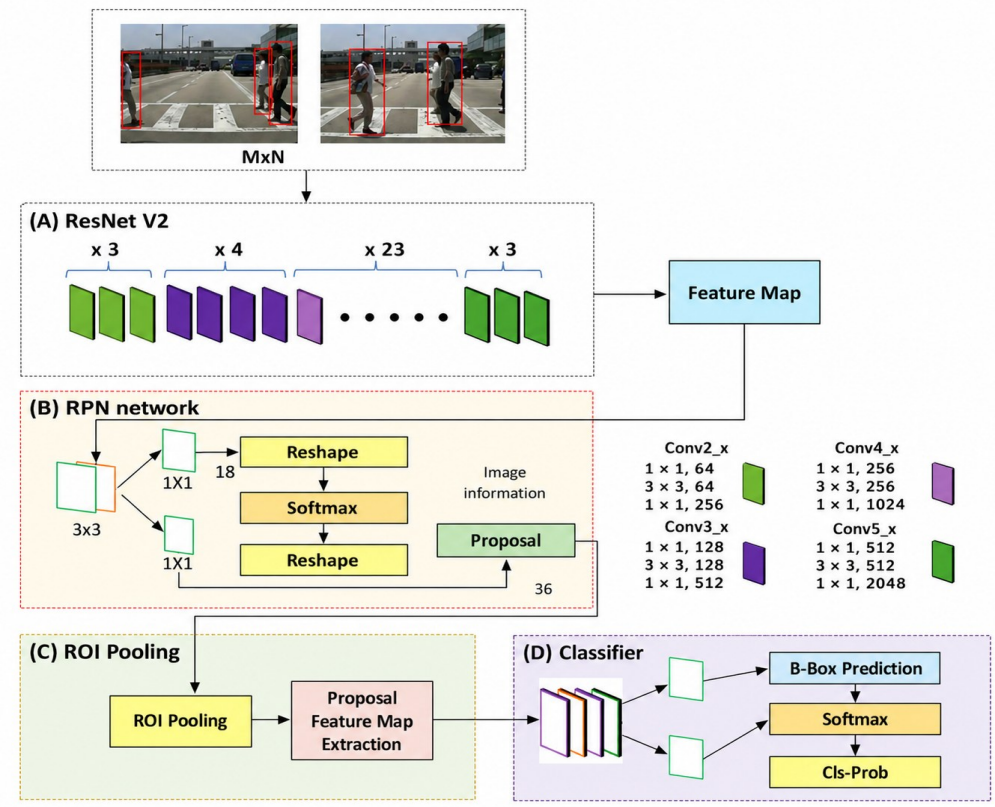


Figure 1. Complete Faster R-CNN architecture [5]

The Faster R-CNN architecture is composed of three main components: a convolutional backbone for extracting feature maps, a RPN for generating candidate object regions, and a detection head that performs object classification along with bounding box regression. The RPN leverages shared convolutional features to efficiently generate high-quality region proposals, significantly reducing computational overhead. Additionally, the region of interest (ROI) pooling layer ensures that feature maps corresponding to proposed regions are resized for the detection stage, facilitating accurate classification and localization. Compared to earlier models, Faster R-CNN achieves superior performance by tightly coupling the region proposal and detection tasks within a single network. For example, it is significantly faster than R-CNN and Fast R-CNN while maintaining high accuracy, this makes it well-suited for tasks that demand high accuracy, including medical image analysis, video processing, and autonomous driving applications. Since its introduction, Faster R-CNN has become a foundational model in object detection research. It has been adapted and improved in various contexts, such as real-time detection [8] and multi-scale detection tasks [8]. Its capability to effectively tradeoff between computational speed and detection accuracy has made it a standard reference for assessing contemporary object detection methods.

2.2. You only look once

Object detection is a core problem in computer vision, widely used in areas such as self-driving vehicles and security surveillance systems. Among the many algorithms developed, YOLO has emerged as one of the most widely adopted and impactful frameworks for real-time object detection. YOLO [17] revolutionized the field by presenting a unified architecture capable of detecting multiple objects in images with remarkable speed and accuracy. Unlike traditional object detection methods, which apply a sliding window or RPN, YOLO performs detection in a single forward pass through a deep neural network, making it significantly faster and suitable for real-time applications. Figure 2 illustrates the YOLO architecture.

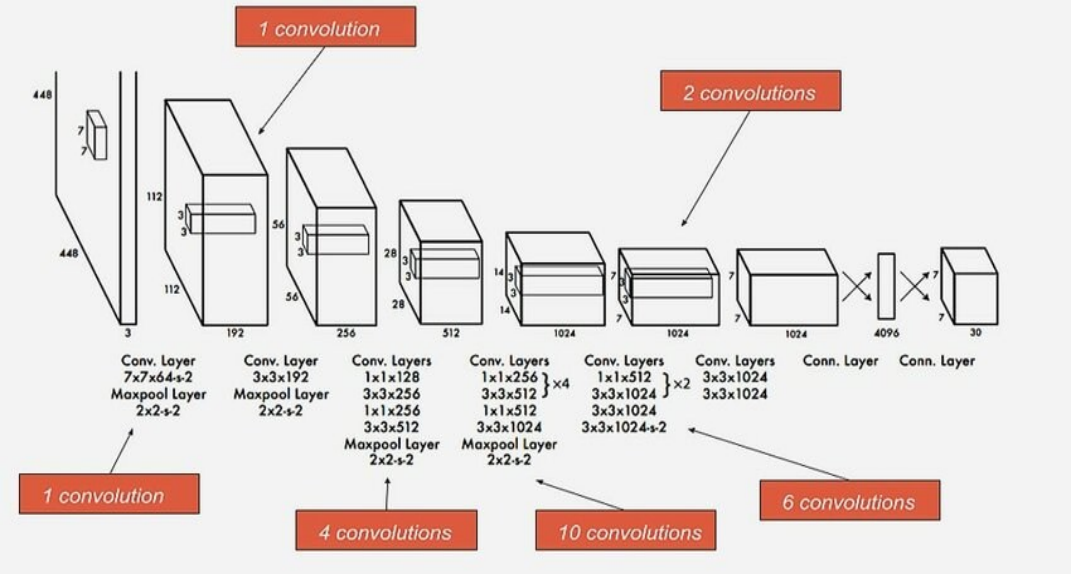


Figure 2. YOLO architecture [9]

YOLO was first introduced by Redmon *et al.* [9] in 2016 with the release of YOLOv1. The core idea was to treat object detection as a single regression problem, where the network directly predicts bounding boxes and class probabilities from an image. This approach resulted in a significant speed advantage over other detection algorithms like R-CNN, which required multi-stage pipelines and were computationally expensive. Subsequent versions of YOLO, such as YOLOv2 (2017) and YOLOv3 (2018), introduced several improvements. YOLOv2 incorporated batch normalization and anchor boxes, improving accuracy and generalization. YOLOv3 further enhanced performance by using a more powerful backbone, Darknet-53, and introducing a multi-scale detection approach, enabling the detection of smaller objects. These improvements allowed YOLO to achieve state-of-the-art accuracy while maintaining its real-time performance, which is crucial for many real-world applications. The evolution of YOLO continued with YOLOv4 and YOLOv5, where the focus shifted towards optimizing the model for different hardware

environments and increasing robustness. YOLOv4 incorporated several enhancements, including Mosaic data augmentation, self-adversarial training, and complete intersection over union (CIoU) loss, to boost detection accuracy. YOLOv5, although not officially part of the original YOLO series, brought further optimizations with a focus on ease of use and deployment, often being regarded for its performance in edge computing applications. The journey of YOLO did not stop with YOLOv5. In 2022, YOLOv6 was introduced, aiming at optimizing the framework for both performance and efficiency. YOLOv6 focused on improving the backbone architecture and incorporating newer techniques in object detection, such as the use of transformer layers for better contextual understanding and attention mechanisms.

The release of YOLOv7 further refined these advances, improving accuracy on a variety of datasets, enhancing its robustness in detecting small objects, and optimizing inference time. YOLOv7 also integrated improved training strategies, such as augmented loss functions, to ensure better generalization to diverse environments. The latest iteration, YOLOv8, is a substantial leap forward, offering state-of-the-art performance in terms of speed and accuracy. YOLOv8 further optimizes the backbone and feature extractor, providing significant improvements in small object detection, multi-class detection, and model adaptability across different devices and use cases. With a focus on deploying in edge devices, YOLOv8 addresses the challenges of low-resource environments, offering both high efficiency and real-time detection capabilities. YOLOv8 also supports advanced features like better multi-object tracking and enhanced post-processing techniques.

3. METHOD

3.1. Object detection using you only look once

Figure 3 presents comprehensive workflow of the object detection framework showing both the training and testing stages. Figure 3(a) shows schematic diagram of the model training procedure, while Figure 3(b) shows schematic diagram of the model testing procedure. The input dataset is divided into training and validation subsets. The resulting datasets are provided as input to the object detection model. In this study, YOLOv8, one of the most recent advancements in object detection, was employed. Proper hyperparameter configuration plays a vital role in achieving efficient model training. The model was trained using a batch size of 16, an input image resolution of 640, and 150 training epochs. The selected YOLO architecture was YOLOv8-x, while the Adam optimizer was applied with momentum parameters set to 0.222, 0.555, and 0.999, respectively. After the model is trained using the training dataset and validated with the validation dataset, unseen data is supplied to evaluate its prediction capability. Figure 3(b) illustrates the testing workflow of the object detection system.

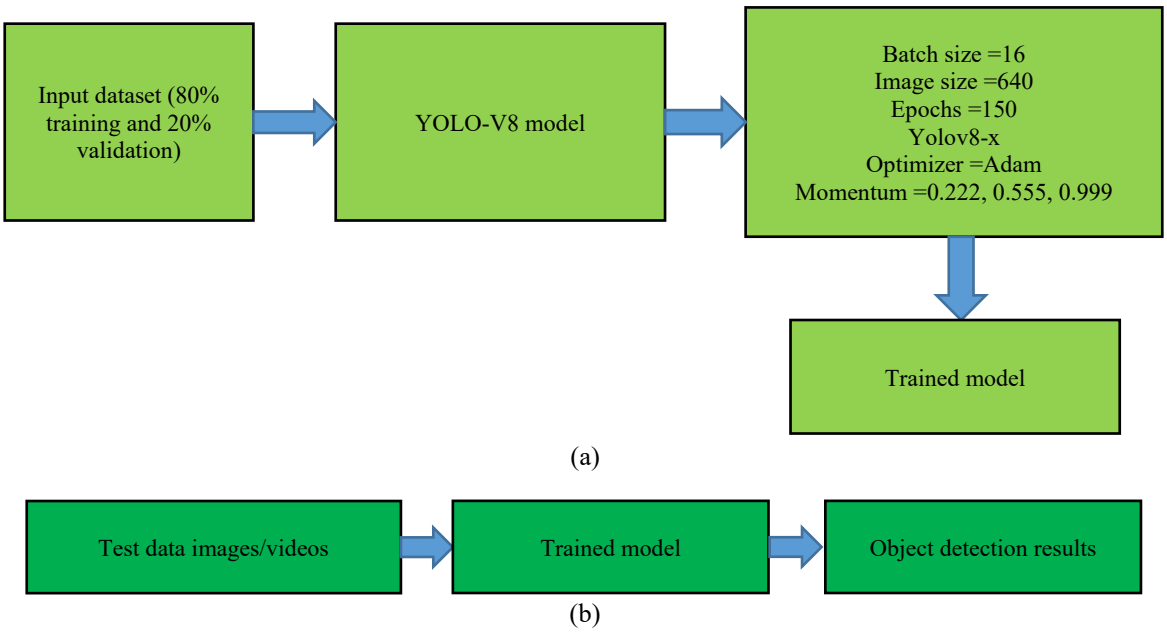


Figure 3. Schematic representation of the YOLO-based object detection model: (a) training process and (b) testing process

3.2. Object detection using Faster region-based convolutional neural network

Figure 4 presents the complete workflow of the object detection framework, covering both training and testing phases. Figure 4(a) shows the schematic representation of the model training pipeline, while Figure 4(b) shows the schematic representation of the model testing pipeline. The input dataset is split into training and validation sets, which are then used to train the object detection model. In this work, Faster R-CNN, a state-of-the-art object detection architecture, is utilized. Proper selection of hyperparameters is critical for effective model training. In this study, the model is trained with a batch size of 16, an input image resolution of 640, and 150 epochs, using the Adam optimizer with momentum parameters of 0.222, 0.555, and 0.999. Furthermore, experiments are performed using two backbone networks, ResNet50 and ResNet101, to evaluate their impact on performance. After training and validation, the model is tested using unseen data to assess its predictive capability. Figure 4(b) illustrates the testing workflow of the object detection system.

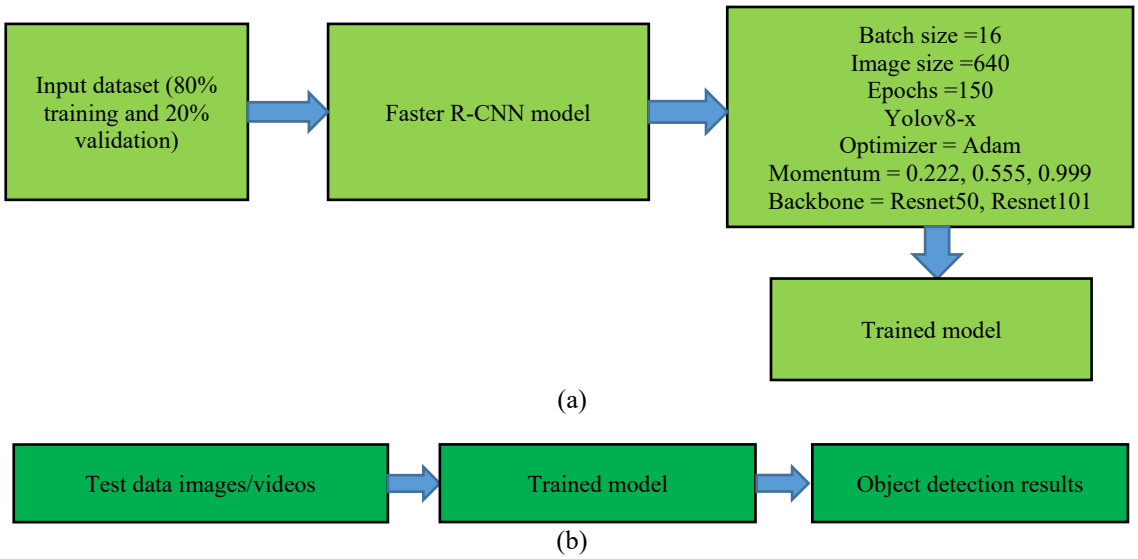
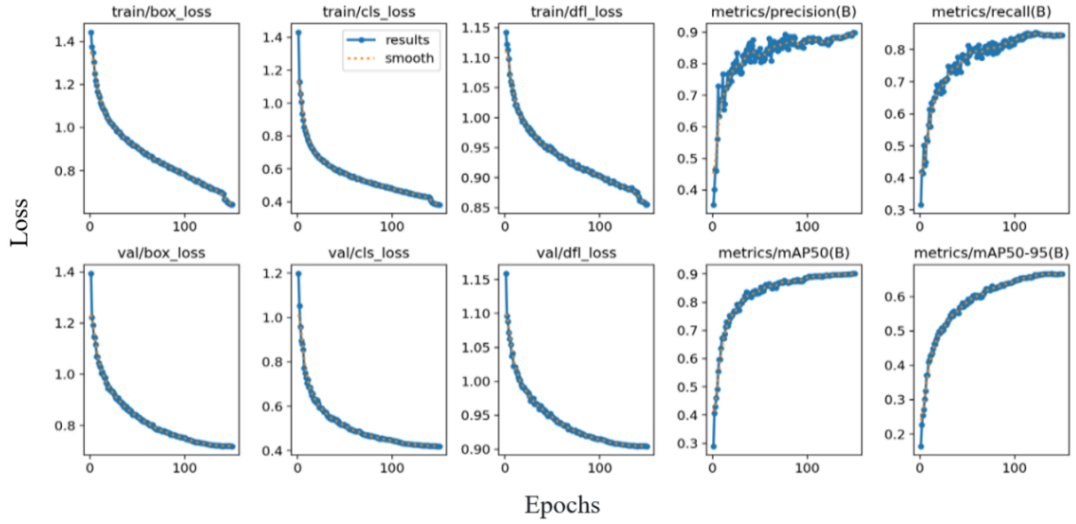


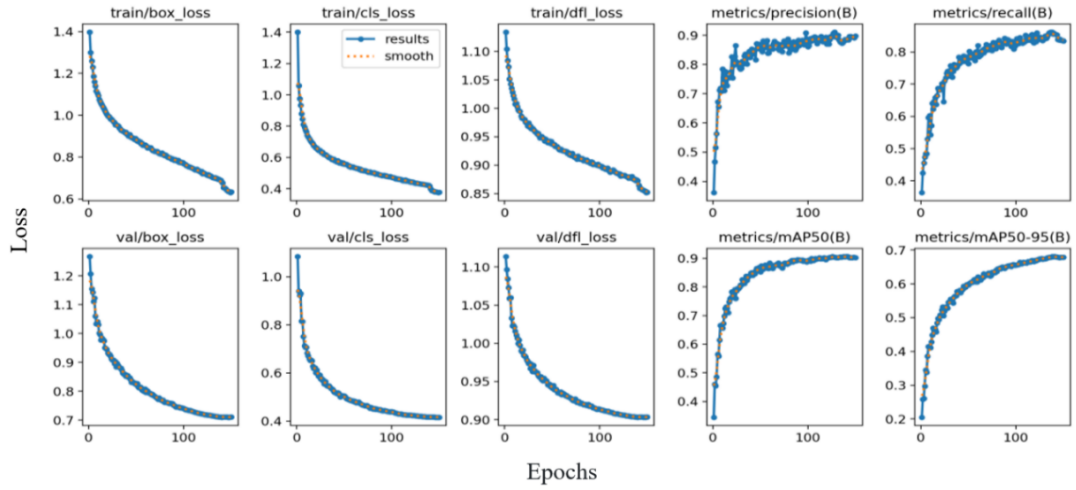
Figure 4. Schematic diagram of the Faster R-CNN-based object detection model: (a) training workflow and (b) testing workflow

4. RESULTS AND DISCUSSION

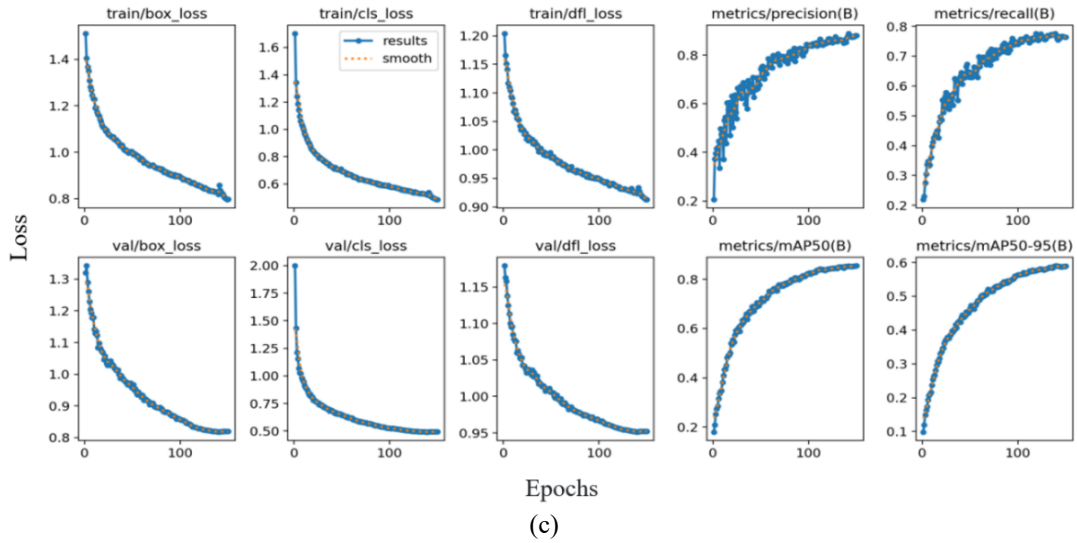
The experiments in this study were carried out using Python 3.9.13 on a Windows 10 Pro system, with 32 GB RAM and a 12th-generation Intel(R) Core (TM) i9-12900 processor running at 2.40 GHz. For accelerated computation, an NVIDIA RTX A5000 GPU with 12 GB VRAM was used, along with PyTorch 2.0.1 and CUDA 11.7. The dataset employed is the publicly available KITTI dataset [26], which contains multiple object categories including cars, pedestrians, vans, cyclists, trucks, miscellaneous objects, trams, and seated persons. In total, the dataset consists of approximately 5,984 training images and 1,400 validation images. The performance of YOLOv8 was analyzed by modifying the momentum parameter of the Adam optimizer while keeping other settings unchanged. The tested momentum values were 0.002, 0.005, and 0.999, with fixed configurations of 150 epochs, batch size of 16, image resolution of 640, and the YOLOv8-x [9] model variant. As shown in Figure 5, when the momentum was set to 0.222 as shown in Figure 5(a), the mAP@50 and mAP@50–95 scores remained below 90% and 70%, respectively. With a momentum value of 0.555 as shown in Figure 5(b), the model achieved improved performance, with mAP@50 exceeding 90% and mAP@50–95 surpassing 70%. However, when the momentum was increased to 0.999 as shown in Figure 5(c), performance declined, with mAP@50 and mAP@50–95 falling below 80% and 60%, respectively. For Faster R-CNN [8] with a ResNet50 backbone, the accuracies achieved with momentum values of 0.222, 0.555, and 0.999 were 18.33%, 26.57%, and 38.31%, respectively. Similarly, with ResNet101 backbone, the corresponding accuracies were 20.86%, 28.03%, and 38.64%. Figures 6 to 10 indicated Faster R-CNN mAP comparative plot of Resnet50 and Resnet101 backbone, mAP versus momentum value plot of Faster R-CNN, backone versus momentum value box plot of Faster R-CNN, backone versus momentum value dot plot of Faster R-CNN, momentum and Map50-95 comparison by backbone of Faster R-CNN. Table 1 indicated comparative performance analysis between YOLO and Faster R-CNN.



(a)



(b)



(c)

Figure 5. Performance analysis with (a) 0.222 momentum, (b) 0.555 momentum, and (c) 0.999 momentum

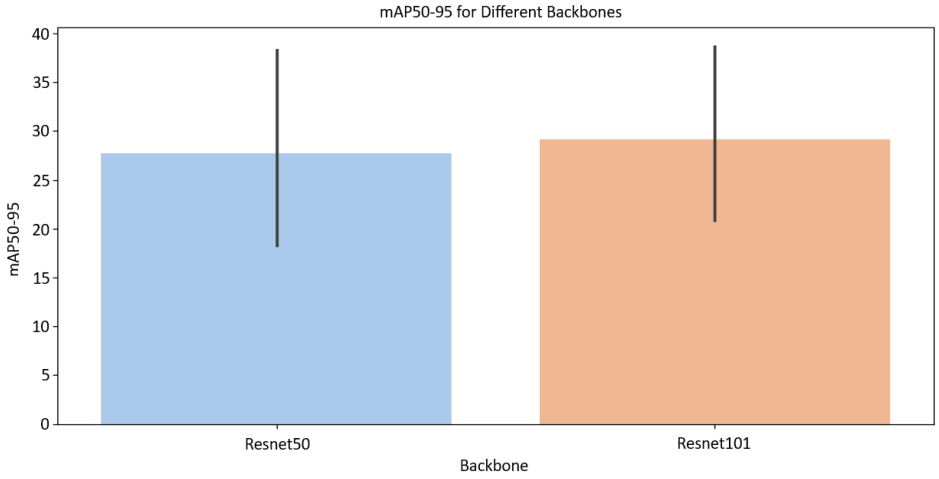


Figure 6. Faster R-CNN mAP comparative plot of Resnet50 and Resnet101 backbone

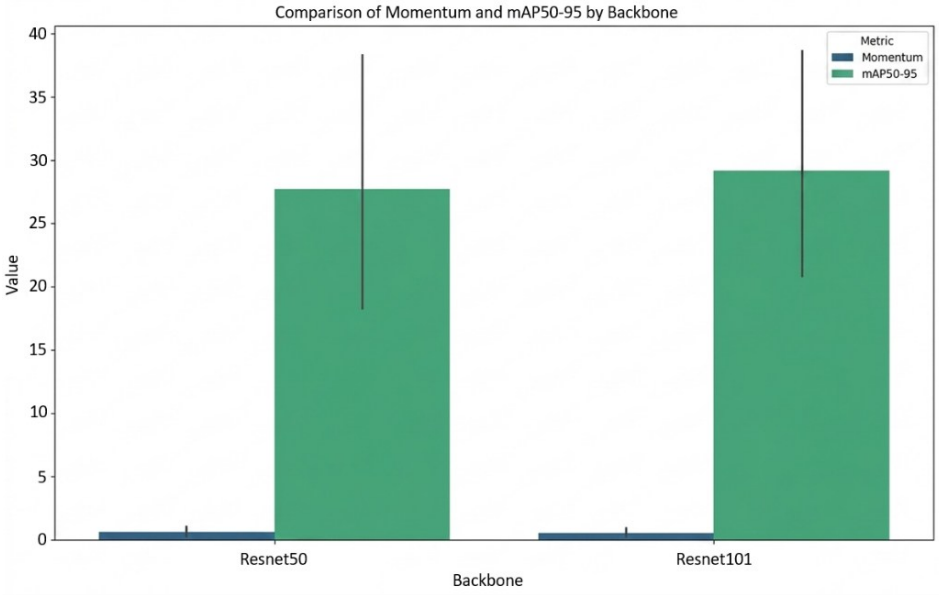


Figure 7. Backbone versus momentum value plot of Faster R-CNN

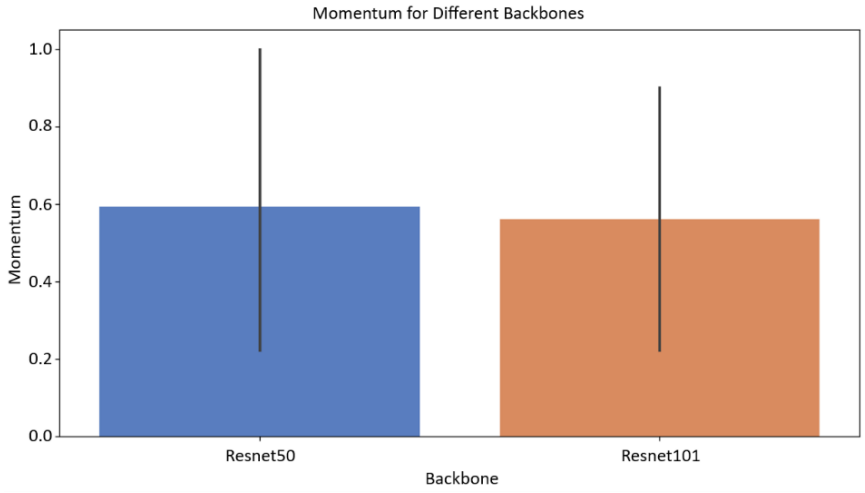


Figure 8. Backone versus momentum value box plot of Faster R-CNN

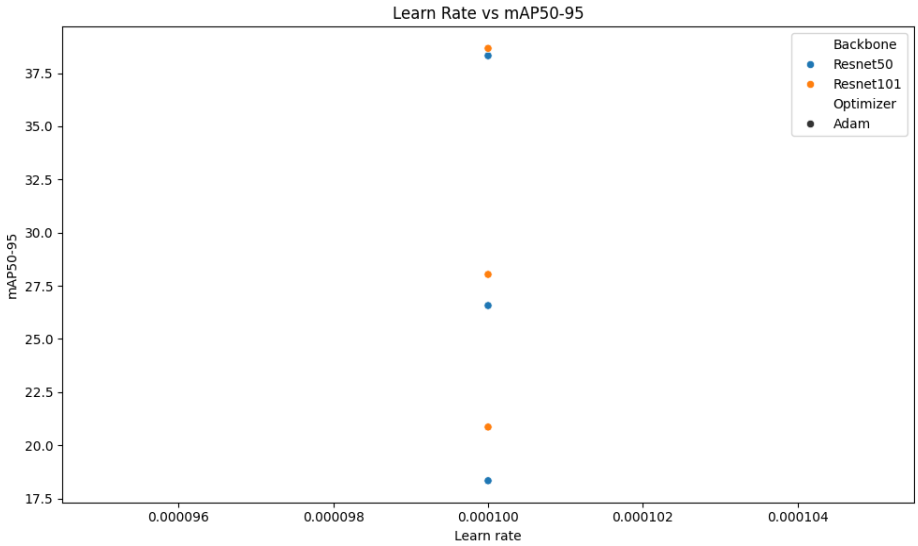


Figure 9. Backone versus momentum value dot plot of Faster R-CNN

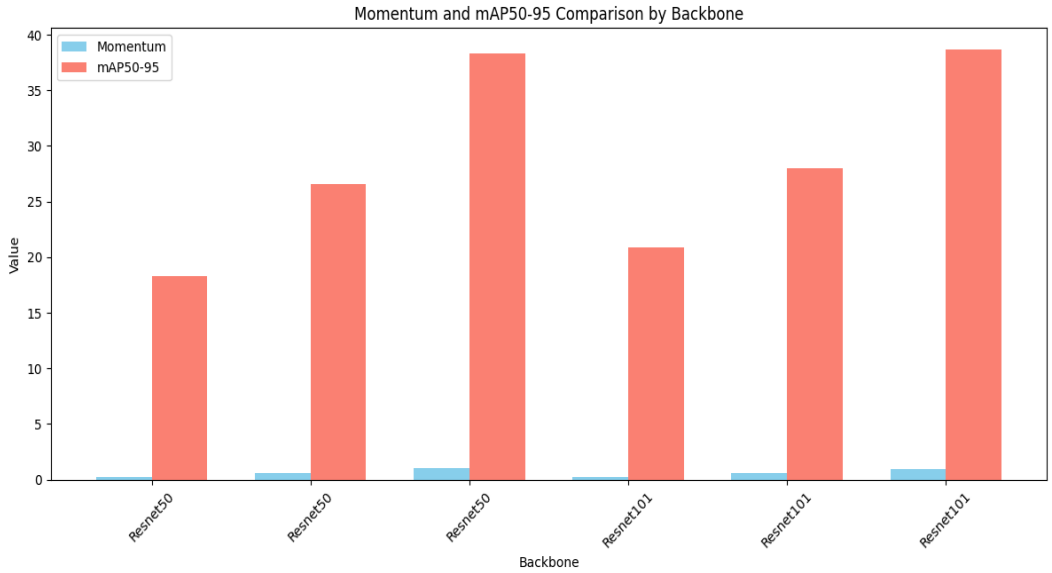


Figure 10. Momentum and mAP50-95 comparison by backbone of Faster R-CNN

Table 1. Comparative analysis between YOLO and Faster R-CNN

Sr. no	YOLO version	YOLO variant	Momentum value	mAP50 (%)	Training time
1	YOLO-V8	YOLOV8-x	0.222	90	3.482 hours
2	YOLO-V8	YOLOV8-x	0.555	90	3.271 hours
3	YOLO-V8	YOLOV8-x	0.999	80	3.375 hours
4	Faster R-CNN	Resnet50 backbone	0.222	18.33	13.08 minutes
5	Faster R-CNN	Resnet50 backbone	0.555	26.57	13.16 minutes
6	Faster R-CNN	Resnet50 backbone	0.999	38.31	13.09 minutes
7	Faster R-CNN	Resnet101 backbone	0.222	20.86	17.23 minutes
8	Faster R-CNN	Resnet101 backbone	0.555	28.036	17.25 minutes
9	Faster R-CNN	Resnet101 backbone	0.999	38.66	17.26 minutes

5. CONCLUSION

This study presents a comparative evaluation of YOLOv8 and Faster R-CNN on the KITTI dataset, focusing on how different Adam optimizer momentum settings influence model performance. The experimental results indicate that YOLOv8 consistently outperforms Faster R-CNN across all tested configurations. For YOLOv8, the best results were obtained at a momentum value of 0.555, where mAP@50

exceeded 90% and mAP@50–95 was above 70%. In comparison, using lower (0.222) and higher (0.999) momentum values led to a noticeable drop in accuracy, showing that YOLOv8 performance is affected by changes in this parameter. For Faster R-CNN, performance depended on the backbone architecture. The ResNet101 backbone generally produced better results than ResNet50. However, even with ResNet101 and the best-performing momentum setting, Faster R-CNN still achieved lower accuracy than YOLOv8 in all cases. Overall, the findings emphasize YOLOv8's superiority in terms of accuracy and suitability for real-time object detection tasks compared to Faster R-CNN. Future research could further explore advancements in model architectures and evaluate their effectiveness across diverse datasets and conditions to expand the applicability of these findings.

ACKNOWLEDGMENTS

We acknowledge JSPM's Rajarshi Shahu College of Engineering, Pune, India, for providing the infrastructure, laboratory facilities, and academic environment necessary for conducting this research.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Madhura M. Bhosale	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	
Yogesh S. Angal	✓					✓				✓		✓	✓	

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors have no conflict of interest to declare. All co-authors have seen and agree with the content of manuscript. We confirmed that submission is original work is not under review at any other publication. authors don't have any financial and non-financial interest in the subject matter or material discussed in this manuscript.

DATA AVAILABILITY

The data that support the findings of this study is KITTI dataset. This dataset openly available in Kaggle at https://www.kaggle.com/datasets/klemenko/kitti-dataset.

REFERENCES

[1] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 2001, pp. 1-9, doi: 10.1109/CVPR.2001.990517.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, pp. 886–893, doi: 10.1109/CVPR.2005.177.

[3] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2008, pp. 1–8, doi: 10.1109/CVPR.2008.4587597.

[4] J. R. R. Uijlings, K. E. A. V. D. Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, Sep. 2013, doi: 10.1007/s11263-013-0620-5.

[5] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, Nov. 1999, pp. 1150–1157, doi: 10.1109/ICCV.1999.790410.

[6] C. Szegedy *et al.*, "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.

[7] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1440–1448, doi: 10.1109/ICCV.2015.169.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," in *29th International Conference on Neural Information Processing Systems*, 2015, pp. 91–99, doi: 1109/TPAMI.2016.2577031.

[9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: unified, real-time object detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.
[10] P. Dai, S. Yao, Z. Li, S. Zhang, and X. Cao, "ACE: anchor-free corner evolution for real-time arbitrarily oriented object detection," *IEEE Transactions on Image Processing*, vol. 31, pp. 4076–4089, 2022, doi: 10.1109/TIP.2022.3167919.
[11] N. M. A. A. Dazlee, S. A. Khalil, S. A. -Rahman, and S. Mutalib, "Object detection for autonomous vehicles with sensor-based technology using YOLO," *International Journal of Intelligent Systems and Applications in Engineering*, vol. 10, no. 1, pp. 129–134, Mar. 2022, doi: 10.18201/ijisae.2022.276.
[12] S. Liang *et al.*, "Edge YOLO: real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 25345–25360, Dec. 2022, doi: 10.1109/TITS.2022.3158253.
[13] M. B. Blaschko and C. H. Lampert, "Learning to localize objects with structured output regression," in *Computer Vision – ECCV 2008 (ECCV 2008)*, Berlin, Heidelberg: Springer, 2008, pp. 2–15, doi: 10.1007/978-3-540-88682-2_2.
[14] X. Zou, "A review of object detection techniques," in *2019 International Conference on Smart Grid and Electrical Automation (ICSGEA)*, Aug. 2019, pp. 251–254, doi: 10.1109/ICSGEA.2019.00065.
[15] J. Terven, D.-M. C. -Esparza, and J.-A. R. -González, "A comprehensive review of YOLO architectures in computer vision: from YOLOv1 to YOLOv8 and YOLO-NAS," *Machine Learning and Knowledge Extraction*, vol. 5, no. 4, pp. 1680–1716, Nov. 2023, doi: 10.3390/make5040083.
[16] J. Han, Y. Liao, J. Zhang, S. Wang, and S. Li, "Target fusion detection of LiDAR and camera based on the improved YOLO algorithm," *Mathematics*, vol. 6, no. 10, Oct. 2018, doi: 10.3390/math6100213.
[17] Q. He, A. Xu, Z. Ye, W. Zhou, and T. Cai, "Object detection based on LightWeight YOLOX for autonomous driving," *Sensors*, vol. 23, no. 17, Sep. 2023, doi: 10.3390/s23177596.
[18] A. Ammar, A. Koubaa, M. Ahmed, and A. Saad, "Aerial images processing for car detection using CNNs: comparison between Faster R-CNN and YOLOv3," *Technical Report CISTER-TR-200107*, pp. 1–12, 2019.
[19] S. Srivastava, A. V. Divekar, C. Anilkumar, I. Naik, V. Kulkarni, and V. Pattabiraman, "Comparative analysis of deep learning image detection algorithms," *Journal of Big Data*, vol. 8, no. 1, Dec. 2021, doi: 10.1186/s40537-021-00434-w.
[20] S. M. Alkentar, B. Alsahwa, A. Assalem, and D. Karakolla, "Practical comparation of the accuracy and speed of YOLO, SSD and Faster RCNN for drone detection," *Journal of Engineering*, vol. 27, no. 8, pp. 19–31, Aug. 2021, doi: 10.31026/j.eng.2021.08.02.
[21] D. D. Aboyomi and C. Daniel, "A comparative analysis of modern object detection algorithms: YOLO vs SSD vs Faster R-CNN," *Information Technology Engineering Journals*, vol. 8, no. 2, pp. 96–106, Dec. 2023, doi: 10.24235/itej.v8i2.123.
[22] W. Liu *et al.*, "SSD: single shot multibox detector," in *Computer Vision – ECCV 2016 (ECCV 2016)*, Cham, Switzerland: Springer, 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.
[23] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, Feb. 2020, doi: 10.1109/TPAMI.2018.2858826.
[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013, doi: 10.1177/0278364913491297.
[25] Z. Cai and N. Vasconcelos, "Cascade R-CNN: delving into high quality object detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 6154–6162, doi: 10.1109/CVPR.2018.00644.
[26] A. Klemenko, "kitti dataset: kitti dataset 2012/2015 stereo images from camera," *kaggle.com*. 2020. Accessed: Jun. 18, 2024. [Online]. Available: https://www.kaggle.com/datasets/klemenko/kitti-dataset

BIOGRAPHIES OF AUTHORS

Madhura M. Bhosale received her M.Tech. degree in signal processing and is currently pursuing a Ph.D. from Savitribai Phule Pune University, Pune, India, in the domain of artificial intelligence (AI) and machine learning. Her doctoral research focuses on object detection and tracking from image and video data, with applications in intelligent transportation systems. She is an R&D engineer with over 5+ years of professional experience in AI, machine learning, and computer vision. Her work primarily involves the development of AI-based video analytics solutions, including automatic number plate recognition (ANPR), video incident detection systems (VIDS), vehicle actuated speed detection (VASD), and PTZ-based analytics. She has extensive experience in training deep learning models such as YOLO variants, working with OCR frameworks, and deploying solutions on GPU and edge AI platforms. Her research interests include deep learning, computer vision, signal and image processing, intelligent video analytics, and embedded AI systems. She can be contacted at email: madhurabhosale4@gmail.com.

Yogesh S. Angal is head the Department of Electronics and Telecommunication Engineering, JSPM's Bhivarabai Sawant Institute of Technology and Research, Pune, India. He has over 20 years of teaching experience and has made significant contributions to academics and research. He has published several research papers in reputed national and international journals and conferences. His areas of interest include electronics and communication engineering domains. He is actively involved in guiding students and faculty toward research, innovation, and academic excellence. He can be contacted at email: ysangal_entc@jspmbsiotr.edu.in.