Published by :
https://www.ijert.org/
An International Peer-Reviewed Journal

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

Enhancing Real-Time Object Detection With Yolo Algorithm

Madhuri Nanasaheb Borse
PG Student, Dept. Of Computer science & Engineering, Shreeyash college of Engineering
Chh. Sambhajinagar , India

Vijaykumar M. P
Asst.Professor, Dept. Of Computer science & Engineering, Shreeyash college of Engineering
Chh. Sambhajinagar , India

Abstract **- Real-time object detection has become an important area of computer vision due to its wide range of applications in intelligent surveillance, traffic monitoring, industrial automation, and smart city systems. This paper presents a real-time object detection system based on the YOLOv8 (You Only Look Once Version 8) deep learning model. The proposed system is designed to process both images and videos provided by the user and identify multiple objects with high detection speed and reliable accuracy. A pre-trained YOLOv8 Nano (yolov8n.pt) model is employed to perform object localization and classification by generating bounding boxes, object labels, and confidence scores. The implementation is carried out in Python using the Ultralytics framework along with OpenCV, NumPy, and Matplotlib for image processing, numerical computation, and result visualization. For image inputs, the detected objects are displayed with annotations, while for video inputs, each frame is processed and saved as an annotated output video. In addition, the system extracts confidence scores for all detected objects and performs confidence-based analysis through graphical visualization. Experimental evaluation demonstrates that the proposed approach provides fast inference and accurate detection across different input scenes while maintaining computational efficiency. The ability to accept both image and video inputs makes the system suitable for practical real-time applications where rapid and reliable object recognition is required. The proposed framework offers a simple, efficient, and scalable solution that can be further extended for domain-specific applications such as intelligent surveillance, traffic analysis, and industrial monitoring.**

Keywords **- *YOLOv8, Real-Time Object Detection, Deep Learning, Computer Vision, Image Processing, Video Processing, Object Localization, Artificial Intelligence (AI), Ultralytics, OpenCV.***

I. INTRODUCTION

Object detection is a fundamental task in computer vision that enables computers to identify, classify, and locate objects within digital images and video sequences. Recent advancements in artificial intelligence (AI) and deep learning have significantly improved the accuracy and speed of object detection systems, making them suitable for real-time applications. These systems are widely used in autonomous vehicles, intelligent surveillance, traffic monitoring, healthcare, industrial automation, robotics, retail analytics, and smart city infrastructure. The ability to accurately detect multiple objects in real time has become increasingly important as the demand for intelligent vision-based systems continues to grow.

Conventional object detection techniques relied on handcrafted features and machine learning algorithms, which often required separate stages for feature extraction, region proposal, and object classification. These approaches were computationally intensive and struggled to maintain high accuracy under challenging conditions such as varying illumination, object occlusion, complex backgrounds, and changes in object scale. The emergence of deep learning has transformed object detection by enabling neural networks to learn discriminative features directly from large datasets, thereby improving both detection performance and computational efficiency.

Among modern object detection algorithms, the YOLO (You Only Look Once) family has gained widespread attention due to its ability to perform object localization and classification in a single forward pass of the neural network. This single-stage detection strategy significantly reduces inference time while maintaining competitive detection accuracy. The latest version, YOLOv8, developed using the Ultralytics framework, introduces improvements in feature extraction, network optimization, and prediction capability, enabling faster and more reliable detection across a wide variety of object classes. Its lightweight architecture also makes it suitable for deployment in applications requiring real-time processing.

This research presents a real-time object detection system based on the pre-trained YOLOv8 Nano (yolov8n.pt) model. The proposed system accepts either an image or a video as input and automatically detects multiple objects present in the scene. During the detection process, the model predicts the object class, draws bounding boxes around detected objects, and assigns confidence scores that indicate the reliability of each prediction. For image inputs, the detected objects are displayed with annotations, whereas for video inputs, each frame is processed sequentially and the annotated video is generated as the final output.

In addition to object detection, the proposed system performs confidence-based analysis to evaluate the prediction performance. The detected object names and their confidence scores are extracted automatically, and graphical visualizations are generated to illustrate the confidence level of each detection. For video inputs, the average confidence score of

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

each detected object class is calculated across all processed frames, providing a comprehensive assessment of detection consistency throughout the video sequence. These analytical features make the proposed system more informative than conventional object detection frameworks that only display annotated outputs.

The implementation is carried out using Python in the Google Colab environment. The Ultralytics library is employed to implement the YOLOv8 model, OpenCV is used for image and video processing, NumPy supports numerical computations, and Matplotlib is utilized for graphical visualization of detection confidence. The use of open-source tools simplifies development while ensuring efficient execution and reproducibility.

The primary objective of this work is to develop a simple, efficient, and reliable real-time object detection system capable of processing both images and videos with high detection accuracy and low computational complexity. The proposed framework demonstrates that the YOLOv8 Nano model provides fast inference, accurate object localization, and effective confidence analysis, making it suitable for a wide range of practical applications, including intelligent surveillance, traffic monitoring, industrial inspection, robotics, and smart automation systems.

II. LITERATURE SURVEY

Wang et al. [1] introduced YOLOv7, which incorporated several trainable "bag-of-freebies" techniques to improve detection accuracy without increasing inference cost. The proposed model demonstrated superior performance on the MS COCO benchmark while maintaining real-time detection speed, making it suitable for practical computer vision applications.

In a subsequent publication, Wang et al. [2] further validated the effectiveness of YOLOv7 through extensive experimental evaluation. Their work showed that the optimized architecture achieved higher precision and faster inference than many existing one-stage detectors, particularly for autonomous driving and intelligent surveillance applications.

Terven and Cordova-Esparza [3] presented a comprehensive review of YOLO architectures from YOLOv1 to YOLOv8 and YOLO-NAS. The study analyzed architectural developments, improvements in feature extraction, detection heads, and training strategies. The authors concluded that recent YOLO models provide an excellent balance between computational efficiency and detection accuracy.

Jocher et al. [4] introduced the Ultralytics YOLOv8 framework, which adopted an anchor-free detection mechanism, improved C2f backbone, and decoupled detection head. The framework unified multiple computer vision tasks, including object detection, image classification, segmentation, and pose estimation, providing greater flexibility for real-world applications.

Yaseen [5] performed an in-depth analysis of the internal architecture of YOLOv8 and discussed its improved feature extraction capability, prediction mechanism, and computational efficiency. The study demonstrated that YOLOv8 achieves faster inference while maintaining high detection accuracy across diverse object categories.

Ramos and Sappa [6] reviewed the evolution of the YOLO family over the past decade. Their survey highlighted the progression from YOLOv1 to modern architectures, emphasizing improvements in backbone networks, feature fusion techniques, lightweight model design, and deployment on embedded systems. The authors identified YOLOv8 as one of the most efficient models for real-time object detection.

Sapkota et al. [7] conducted a comprehensive review of YOLO algorithms, discussing major architectural innovations, benchmark datasets, optimization techniques, and future research directions. Their study highlighted the importance of lightweight models for edge computing and real-time intelligent vision systems.

Poureskandar and Razzagzadeh [8] investigated different training strategies for YOLOv8 to improve detection accuracy and robustness. Their experimental analysis showed that proper hyperparameter optimization and dataset augmentation significantly enhance model performance in challenging environments.

Megantara and Utami [9] presented a systematic review of YOLOv8 applications in agriculture, healthcare, unmanned aerial vehicles, environmental monitoring, and industrial automation. The review concluded that YOLOv8 consistently provides reliable detection accuracy with reduced computational complexity, making it suitable for practical deployment.

Hidayatullah et al. [10] compared the architectures of YOLOv8, YOLOv9, YOLOv10, and YOLO11. Their comparative study demonstrated continuous improvements in feature representation, inference speed, and detection accuracy while reducing computational requirements. The authors emphasized that newer YOLO architectures are increasingly suitable for edge devices and embedded AI applications.

Ramos and Sappa [11] further analyzed the historical development of YOLO-based object detection and discussed emerging research trends, including transformer-based detection, lightweight neural networks, and hybrid vision architectures. Their work highlighted the importance of balancing accuracy with computational efficiency for real-time applications.

Jocher et al. [12] documented the complete Ultralytics YOLO model family and described the implementation, training procedures, validation methods, and deployment strategies for different YOLO versions. Their documentation serves as an important reference for researchers implementing modern object detection systems.

The survey presented in Neurocomputing [13] discussed the development and evolution of YOLO algorithms, focusing on improvements in backbone design, feature aggregation, loss functions, and model optimization. The authors concluded that recent YOLO models have significantly improved localization accuracy while maintaining real-time inference capability.

The study published in Results in Engineering [14] benchmarked several YOLO-based deep learning models for Advanced Driver Assistance Systems (ADAS) and intelligent transportation applications. Experimental results demonstrated that recent YOLO models outperform conventional detectors

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

in terms of processing speed and detection accuracy under real-world traffic conditions.

Finally, Xu [15] investigated mainstream YOLO object detection models and their practical applications across surveillance, healthcare, industrial automation, and autonomous systems. The study compared different YOLO variants and concluded that lightweight versions, particularly YOLOv8 Nano, provide an effective compromise between computational efficiency and detection accuracy for real-time deployment.

III. PROPOSED METHODOLOGY

The proposed system is designed to perform real-time object detection using the YOLOv8 (You Only Look Once Version 8) deep learning algorithm. The system accepts either an image or a video as input and automatically detects multiple objects present in the scene. A pre-trained YOLOv8 Nano (yolov8n.pt) model is employed to achieve high detection speed while maintaining reliable accuracy. The implementation is carried out in Python using the Google Colab platform, which provides a simple and efficient environment for model execution and evaluation.

Initially, the required software libraries, including Ultralytics, OpenCV, NumPy, and Matplotlib, are installed and imported. The Ultralytics library is used to load the YOLOv8 Nano model, OpenCV handles image and video processing, NumPy performs numerical computations, and Matplotlib is used to visualize the detection results and confidence analysis.

The proposed framework accepts both image and video files uploaded by the user. After the input file is received, the system automatically determines whether it is an image or a video based on its file extension. For image inputs, the system reads the image and extracts its resolution. For video inputs, important video properties such as frame width, frame height, frame rate (FPS), and total number of frames are obtained before processing begins.

The uploaded input is then supplied to the YOLOv8 model for object detection. Unlike conventional two-stage detectors, YOLOv8 performs object localization and classification simultaneously in a single forward pass through the neural network. The model predicts the object category, bounding box coordinates, and confidence score for every detected object. When an image is processed, the detected objects are highlighted with bounding boxes, object labels, and corresponding confidence values. For video processing, every frame is analyzed individually, and the detected objects are annotated throughout the video sequence. The processed video is automatically saved after the detection process is completed.

Following object detection, the proposed system extracts the names of all detected objects along with their confidence scores. For image inputs, the confidence value of each detected object is displayed directly. For video inputs, confidence scores collected from all processed frames are grouped according to object class, and the average confidence value is calculated for each detected category. This statistical analysis provides a better understanding of the consistency and reliability of the detection model throughout the video.

To further evaluate the detection performance, the system generates graphical representations of the confidence scores. A detection-wise line graph illustrates the confidence associated with each detected object, while a horizontal bar chart compares the average confidence values among different object classes. These visualizations assist in interpreting the performance of the object detection model and identifying variations in prediction confidence.

A) System Architecture

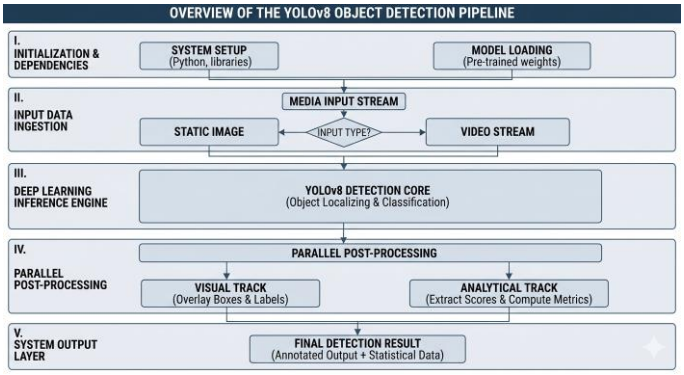

Figure 1: System architecture

The architecture consists of seven functional modules that work sequentially to process the input, perform object detection, analyze the results, and generate visual outputs. Figure 1 illustrates the overall architecture of the proposed system.

- **Environment Setup**

The first stage initializes the software environment by installing and importing all required Python libraries. The Ultralytics library provides the implementation of the YOLOv8 model, OpenCV is used for image and video processing, NumPy performs numerical operations, and Matplotlib is employed for graphical visualization. After the libraries are loaded successfully, the pre-trained YOLOv8 Nano (yolov8n.pt) model is initialized. This lightweight model has been trained on the COCO dataset and is capable of detecting a wide variety of everyday objects while maintaining fast inference speed.

- **Input Acquisition**

In the second stage, the system accepts visual data from the user in the form of either an image or a video. The uploaded file is automatically recognized based on its extension, allowing the same framework to process different types of input without requiring any modification to the detection pipeline. This flexibility makes the proposed system suitable for various real-time applications.

- **Input Pre-processing**

Once the input is received, the system performs basic preprocessing. If the uploaded file is an image, the image is read into memory and its resolution is extracted. If the input is a video, OpenCV is used to obtain important video properties such as frame width, frame height, total number of frames, and frames per second (FPS). These properties assist in managing the video processing task and preserving the original video characteristics during output generation.

- **Object Detection Using YOLOv8**

The preprocessed image or video frames are passed to the YOLOv8 inference engine. The model analyzes the entire image or each video frame in a single forward pass and simultaneously performs object localization and classification. For every detected object, YOLOv8 predicts the object class, bounding box coordinates, and confidence score. In image processing, the detected objects are immediately displayed with

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

bounding boxes and labels. During video processing, each frame is analyzed sequentially, and the detected objects are annotated before being combined into a processed output video.

- **Detection Analysis**

Following object detection, the system extracts detailed information about every detected object. The object names and corresponding confidence scores are recorded for further analysis. For image inputs, the confidence value of each detected object is displayed individually. For video inputs, confidence values obtained from all processed frames are grouped according to object class, and the average confidence score is calculated. This provides a more comprehensive evaluation of the detection consistency throughout the video sequence.

- **Confidence Visualization**

To improve the interpretation of detection performance, the proposed system generates graphical representations of the confidence values. A detection-wise line graph illustrates the confidence associated with each detected object, while a horizontal bar chart compares the average confidence scores of different object categories. These visualizations help users understand the reliability of the predictions and identify objects that are detected with lower confidence.

- **Output Generation**

The final stage produces the complete detection results. For image inputs, the annotated image containing bounding boxes, object labels, and confidence scores is displayed to the user. For video inputs, the processed video with object annotations is automatically saved and can be viewed after processing is completed. Along with the annotated output, the system presents the detected object list, confidence analysis, and graphical visualizations. The modular architecture ensures efficient execution while maintaining high detection accuracy and low computational complexity.

B) Flowchart

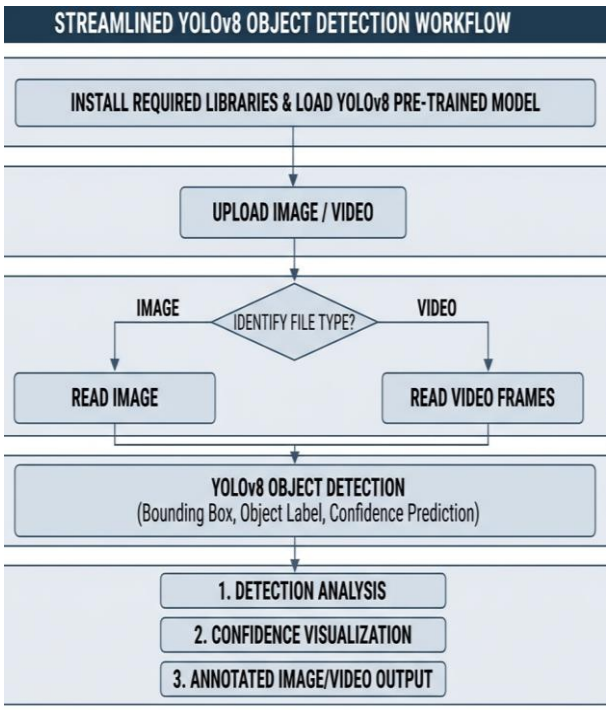

Figure 2: Flowchart

C) Algorithm

Algorithm 1: Proposed YOLOv8-Based Object Detection
Input: Image or Video
Output: Detected Objects with Labels and Confidence Scores

1. Start
2. Install and import required libraries.
3. Load the pre-trained YOLOv8 Nano model.
4. Accept an image or video as input.
5. Determine the input file type.
6. If the input is an image:
   - Read image dimensions.
   - Perform object detection.
   - Draw bounding boxes and labels.
7. Else if the input is a video:
   - Read video frames.
   - Extract resolution and FPS.
   - Detect objects in each frame.
   - Save the annotated video.
8. Extract object names and confidence scores.
9. Calculate average confidence for each detected object (video case).
10. Generate confidence plots and bar charts.
11. Display the final annotated output.
12. Stop.

D) Experimental Setup

The proposed object detection system was implemented using the Python programming language in the Google Colab environment. Google Colab provides cloud-based computational resources and simplifies the execution of deep learning models without requiring dedicated hardware. The implementation was carried out using the Ultralytics YOLOv8 framework together with OpenCV, NumPy, and Matplotlib libraries.

The lightweight YOLOv8 Nano (yolov8n.pt) pre-trained model was selected because it provides an effective balance between detection accuracy and inference speed. The model was trained on the COCO dataset, enabling it to recognize 80 common object categories without additional training.

During experimentation, the system accepted both images and videos as input. Image files were processed directly, whereas video files were processed frame by frame. OpenCV was used to extract image dimensions, video resolution, and frames per second (FPS). After preprocessing, the YOLOv8 model performed object detection and generated bounding boxes, object labels, and confidence scores for all detected objects.

The confidence values obtained from the model were further analyzed. For images, the confidence score of every detected object was recorded. For videos, confidence values from all processed frames were grouped according to object class, and the average confidence score for each class was calculated. These values were visualized using line graphs and horizontal bar charts generated through Matplotlib.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

IV. RESULT AND DISCUSSION

The proposed real-time object detection system was evaluated using several images and videos containing different object categories. The YOLOv8 Nano model successfully detected multiple objects, including persons, cars, buses, trucks, trains, and traffic lights, while maintaining fast inference speed. The experimental results demonstrate that the proposed framework is capable of processing both static images and continuous video streams efficiently.

For image inputs, the system accurately identified the objects present in the scene and generated bounding boxes together with class labels and confidence scores. The detected objects were clearly distinguished even in moderately complex backgrounds, indicating the robustness of the YOLOv8 Nano model. The annotated images confirmed that the model was able to localize multiple objects simultaneously without requiring additional preprocessing.

For video inputs, every frame was processed independently by the YOLOv8 inference engine. The detected objects remained consistently tracked across consecutive frames, and the processed video was successfully generated with object annotations. The extracted confidence values were grouped according to object class, and the average confidence score was calculated for each category. This analysis demonstrated that frequently detected objects generally maintained stable confidence values throughout the video sequence.

The confidence analysis graphs provided additional insight into the detection performance. The line graph illustrated the confidence variation among individual detections, while the horizontal bar chart compared the average confidence of different object classes. These visualizations showed that most detected objects achieved high confidence values, indicating reliable predictions by the YOLOv8 Nano model.

The lightweight architecture of YOLOv8 Nano contributed to fast inference while requiring relatively low computational resources. Consequently, the proposed system is suitable for real-time applications where rapid object detection is essential. Although the model performed effectively under normal conditions, slight reductions in confidence were observed for partially occluded objects, distant targets, and scenes with complex backgrounds. Nevertheless, the overall detection performance remained satisfactory for practical computer vision applications.

The experimental results demonstrate that the proposed system effectively combines accurate object detection, confidence analysis, and graphical visualization within a single framework. The ability to process both images and videos without additional model training makes the system flexible and suitable for intelligent surveillance, traffic monitoring, industrial automation, robotics, and smart city applications.

Figure 3. Output Image with Object Labels and Confidence Analysis

The experimental results shown in Figure X demonstrate the effectiveness of the proposed YOLOv8-based real-time object detection system. The uploaded traffic scene image was processed using the pre-trained YOLOv8 Nano model, which successfully identified multiple objects present in the scene. The first part of the figure shows the annotated output image generated by the YOLOv8 model. Green bounding boxes are drawn around the detected objects, while each bounding box is associated with the predicted object label and its confidence score. In this traffic scene, the model successfully detected several road users and traffic-related objects such as:

- Person
- Bus
- Truck
- Traffic light
- Handbag

The detection results indicate that the model can simultaneously recognize multiple object classes in a single image while accurately locating their positions. The generated bounding boxes confirm that the proposed system performs real-time multi-object detection efficiently.

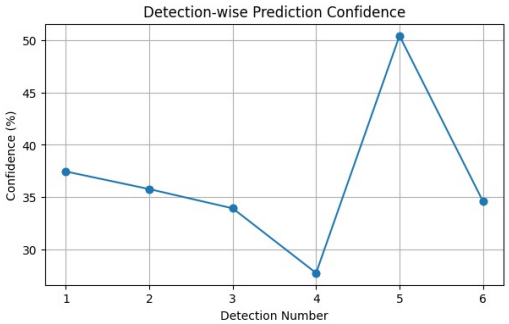

Figure 4: Detection-wise Prediction Confidence

The second graph illustrates the confidence score for each individual detection. The horizontal axis represents the detection number, whereas the vertical axis represents the prediction confidence (%).

From the graph, it can be observed that:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 15 Issue 07 , July - 2026

- Most detections achieved confidence values between 30% and 50%.
- One detected object obtained the highest confidence of approximately 50%, indicating a more reliable prediction.
- A small variation in confidence values exists because different objects have different sizes, viewing angles, illumination conditions, and levels of occlusion.

This graph helps evaluate the reliability of every individual detection produced by the YOLOv8 model.

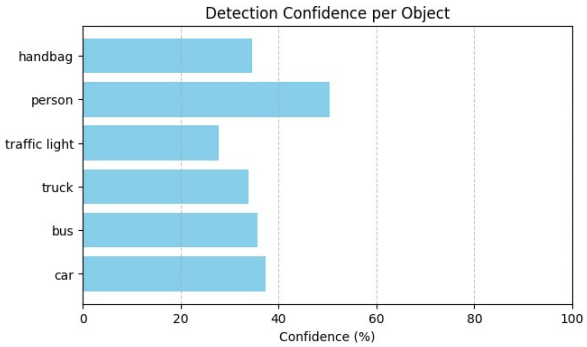

Figure 5: Detection Confidence per Object

V. CONCLUSION

The proposed work presented a real-time object detection system based on the YOLOv8 Nano deep learning model for processing both image and video inputs. The system was developed in Python using the Ultralytics framework and integrated with OpenCV, NumPy, and Matplotlib for image/video processing, numerical analysis, and result visualization. The developed framework automatically accepts an input image or video, identifies the input type, and performs object detection without requiring separate processing methods from the user.

The experimental results demonstrated that YOLOv8 was able to detect multiple objects such as persons, cars, buses, trucks, traffic lights, and other objects in complex scenes. Each detected object was represented using a bounding box, class label, and confidence score. The video processing module successfully analyzed individual frames and generated an annotated output video. In addition, the confidence analysis module calculated and visualized detection confidence, allowing the reliability of different object predictions to be examined.

The graphical results showed that the confidence level varies depending on object size, visibility, distance, background complexity, and partial occlusion. Despite these variations, the system provided satisfactory detection performance with fast inference, demonstrating the suitability of the lightweight YOLOv8 Nano model for real-time applications.

REFERENCES

[1] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," arXiv preprint arXiv:2207.02696, 2022.

[2] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 7464-7475.

[3] J. Terven and D. Cordova-Esparza, "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, 2023.

[4] G. Jocher, A. Chaurasia, and J. Qiu, Ultralytics YOLOv8 Documentation, Ultralytics, 2023. Ultralytics YOLOv8 Documentation

[5] M. Yaseen, "What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector," arXiv preprint arXiv:2408.15857, 2024.

[6] L. T. Ramos and A. D. Sappa, "A Decade of You Only Look Once (YOLO) for Object Detection: A Review," IEEE Access, vol. 13, pp. 192747-192794, 2025.

[7] R. Sapkota, M. Flores-Calero, R. Qureshi, C. Badgujar, U. Nepal, A. Poulose, and P. Zeno, "YOLO Advances to Its Genesis: A Decadal and Comprehensive Review of the You Only Look Once (YOLO) Series," Artificial Intelligence Review, vol. 58, 2025.

[8] R. Poureskandar and S. Razzagzadeh, "Improving Object Detection Performance through YOLOv8: A Comprehensive Training and Evaluation Study," arXiv preprint, 2025.

[9] N. A. Megantara and E. Utami, "Object Detection Using YOLOv8: A Systematic Review," Sistemasi, vol. 14, no. 1, 2025.

[10] P. Hidayatullah, N. Syakrani, M. R. Sholahuddin, T. Gelar, and R. Tubagus, "YOLOv8 to YOLO11: A Comprehensive Architecture In-depth Comparative Review," arXiv preprint arXiv:2501.13400, 2025.

[11] L. T. Ramos and A. D. Sappa, "A Decade of You Only Look Once (YOLO) for Object Detection," arXiv preprint arXiv:2504.18586, 2025.

[12] G. Jocher et al., "Ultralytics YOLO Model Family," Ultralytics Documentation, 2023. Ultralytics YOLO Models

[13] "Development and Evolution of YOLO in Object Detection: A Survey," Neurocomputing, 2025.

[14] "Benchmarking YOLO-Based Deep Learning Models for Real-Time Object Detection in Hybrid ADAS and Intelligent Transportation Systems," Results in Engineering, 2025.

[15] Y. Xu, "An Investigation of Mainstream YOLO Object Detection Models and Applications," 2026.