International Journal of Environmental Sciences
ISSN: 2229-7359
Vol. 11 No. 21s, 2025
https://theaspd.com/index.php

YOLO Object Detection with Opencv and Python

Gopisetty Harshitha[1], Jampani satish Babu[2]
[1]Department. Of CSE, Koneru Lakshmaiah Education Foundation, Green Fields Vaddeswaram, Guntur Dist, Andhra Pradesh, India. Itsharshithagopisetty@gmail.com
[2]Department. Of CSE, Koneru Lakshmaiah Education Foundation, Green Fields Vaddeswaram, Guntur Dist, Andhra Pradesh, India. jampanisatishbabu@gmail.com

Abstract- *This project outlines a Python-based object detection pipeline that integrates the YOLO (You Only Look Once) deep learning architecture with OpenCV's Deep Neural Network (DNN) module. The system is engineered to analyze static images, precisely locating objects by employing a pre-trained YOLO model, which is configured via user-supplied files for its setup, weights, and class labels. Input images are transformed through blob processing to ensure compatibility with the network, then fed into the neural network to acquire predictions from its designated output layers. Object detections are subsequently refined by applying a confidence threshold, and Non-Maximum Suppression (NMS) is utilized to eliminate redundant findings and boost accuracy. Every confirmed object is visually marked on the original image with a colored bounding box and its corresponding class label. The final annotated image is then both displayed to the user and saved for future reference. This codebase provides a versatile and adaptable utility for deploying pre-trained YOLO models on image datasets, establishing a robust foundation for future real-time detection systems or broader computer vision applications.*

Keywords- Object Detection, YOLO (You Only Look Once), Open CV, Deep Learning, Non-Maximum Suppression, Computer Vision

1.INTRODUCTION

In recent years, object detection has emerged as a cornerstone of computer vision, empowering machines to identify and locate various objects within digital images or video frames. Its widespread utility spans diverse applications, ranging from autonomous vehicles and intelligent surveillance to robotics and human-computer interaction. Among the most effective and widely adopted methodologies in this field is the YOLO (You Only Look Once) technique, celebrated for its optimal balance of speed and accuracy, making it particularly well-suited for real-time object detection tasks.

This project delves into the implementation of a YOLO-based object detection system using Python and OpenCV. Leveraging a pre-trained YOLO model, this system can accurately identify objects across 80 distinct object classes, as annotated by the COCO (Common Objects in Context) dataset, within static images. The process commences with loading the image and preparing it using OpenCV's blobFromImage function, which structures the image into a 4D blob format suitable as input for a neural network. The network's architecture and its learned weights are loaded from dedicated configuration and weight files, enabling the detection of objects based on established patterns.

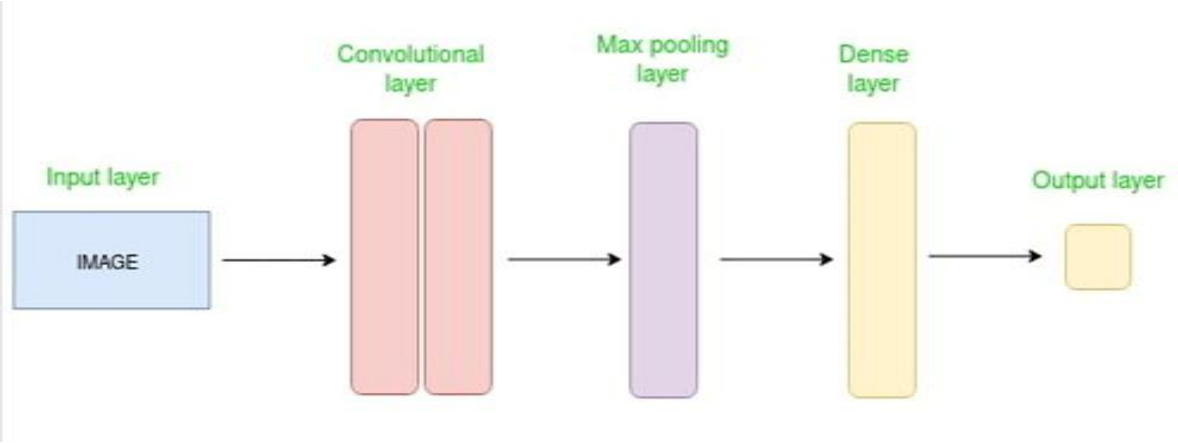
The core object detection procedure involves interpreting the output from the YOLO network layers to yield bounding boxes and confidence values for recognized objects. Detections deemed weak are suppressed by a defined confidence threshold, while Non-Maximum Suppression (NMS) is applied to eliminate overlapping boxes and retain only the most robust predictions. Subsequently, each identified object is annotated on the image with a bounding box and its class label, providing a clear visual summary of the detection outcomes.The implementation is a simple and flexible foundation for understanding how pre-trained deep learning models can be utilized for real-world object detection. It is constructed in modular fashion for effortless extension, and thus best suited to learning uses, research, or application in greater computer vision pipelines. The simplicity of the technique, combined with the power of YOLO and OpenCV, demonstrates the effectiveness of modern deep learning techniques in visual perception.

2. Related work

Object detection has experienced significant progress in the field of computer vision with the advancements of deep learning techniques and the current availability of large annotated datasets such as COCO and PASCAL VOC. Object detection approaches such as Viola-Jones and Histogram of Oriented Gradients (HOG) were traditional in design, of limited capability, and could not manage multiple objects in cluttered scenes.

The introduction of convolutional neural networks (CNNs) was a paradigm shift.Region-based Convolutional Neural Networks (R-CNN) and its extensions, such as Fast R-CNN and Faster R-CNN,

enhanced detection performance by generating regions of interest (RoIs) and classifying them with CNNs. These approaches, though, involved several processing steps and were computationally intensive for real-time use.



YOLO (You Only Look Once) transformed object detection by transforming it into a single regression problem, predicting bounding boxes and class probabilities directly from entire images in one forward pass. The architecture strongly minimized inference time, enabling real-time detection. Later iterations like YOLOv2, YOLOv3, and YOLOv4 enhanced detection accuracy, model resilience, and scalability to varied image resolutions and object sizes.

Other models like SSD (Single Shot MultiBox Detector) and RetinaNet provide competitive performance, specifically in balancing speed and accuracy. Nevertheless, YOLO is still one of the most commonly used models because of its simplicity and efficiency, especially in embedded applications and real-time contexts.

This work advances these by applying a YOLO-based detection system to OpenCV's DNN module in Python. The use of pre-trained weights for YOLO on the COCO dataset enables efficient detection of common objects, while methods such as Non-Maximum Suppression are utilized to condition output by removing redundant predictions. This implementation is made current with extensibility and simplicity for incorporation into larger computer vision systems.

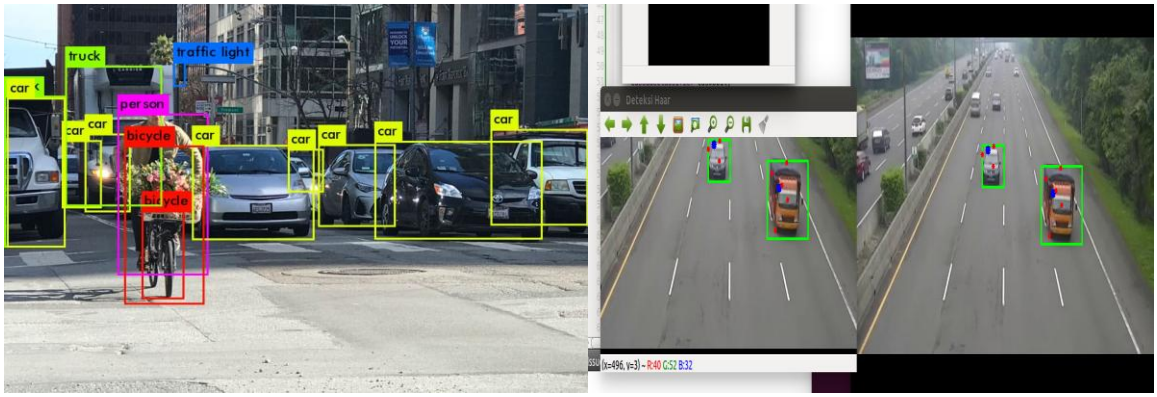
the key techniques employed for object detection in computer vision. The techniques vary from conventional methods such as background subtraction to current deep learning methods.

2.1. Learning YOLO and CNN Object Detection

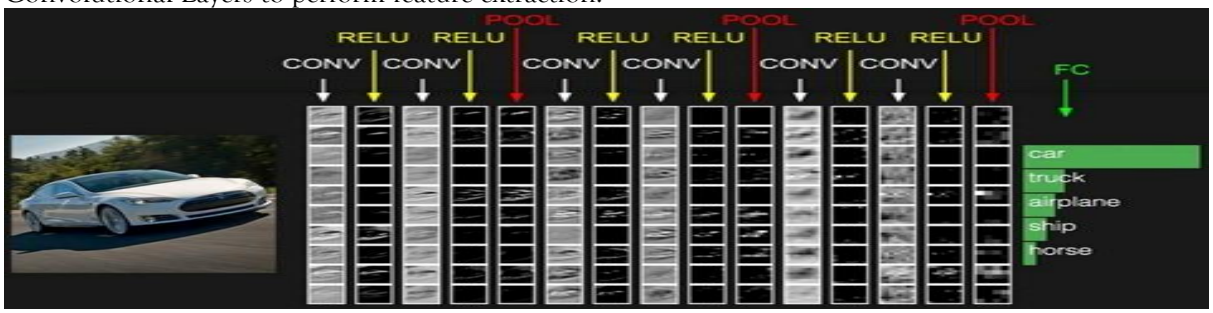
YOLO (You Only Look Once)

YOLO, which stands for You Only Look Once, is a widely used real-time object detection algorithm that frames object detection as a single regression task. In contrast to running separate algorithms to identify and categorize an object, YOLO does a single pass through the entire image and predicts class probabilities and bounding boxes from full images directly.

YOLO is implemented in the provided code using OpenCV's dnn module, which loads pre-trained weights (.weights) and the YOLO config file (.cfg). YOLO divides the input image into a grid and predicts class scores and bounding boxes for each grid cell simultaneously. This allows it to detect objects fast and effectively and is therefore best used in real-time applications like surveillance, robotics, and self-driving cars.



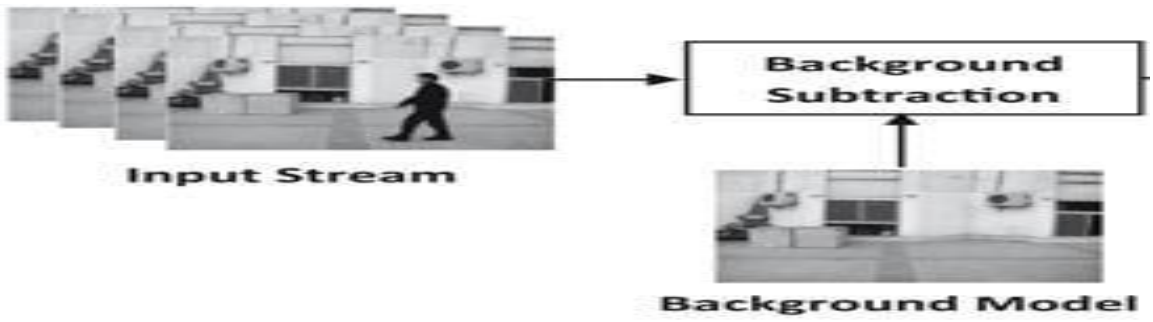
Key steps are:
Pre-processing input image to blob using cv2.dnn.blobFromImage.
Feeding the blob to YOLO network.
Passing it through the network to obtain predictions.
Application of Non-Maximum Suppression (NMS) to eliminate overlapping detections.
Convolutional Neural Networks (CNN)
CNNs form the fundamental architecture that lies behind YOLO. A CNN is a model of deep learning that is designed specifically to handle visual data. It employs convolutional layers in order to extract the spatial features like edges, texture, and patterns, which are vital for the identification of objects in an image.
In YOLO, there is a CNN, which is composed of various layers:
Convolutional Layers to perform feature extraction.



Batch Normalization and Activation Functions (such as Leaky ReLU) to accelerate training and enhance accuracy.
Shortcut connections, which are analogous to residual connections, for enhancing gradient flow in deeper networks.
Every layer enables the network to learn more complex features, from basic edges in shallow layers to complete object forms in deeper layers. YOLO utilizes this hierarchy of features to detect objects at different scales.
Integration in the Code
In your code:
The architecture of CNN is specified by .cfg file, which defines the network structure (layers, filters, kernel size).
Pre-trained weights are loaded to prevent training from scratch.
The image is fed into the CNN, and YOLO processes the final output to draw bounding boxes on detected objects.
The integration of YOLO and CNN in this code facilitates fast, high-speed object recognition that performs well even on complicated scenes with multiple objects.

3. Current methods of detecting moving objects

1. Background Subtraction
Theory:
This technique involves modeling the background scene and subtracting it from the current frame to detect foreground (moving) objects.

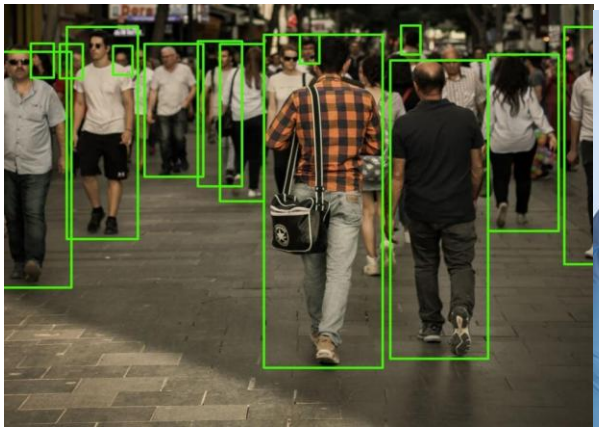


a. Running Average / Frame Averaging
Maintains an average of the last few frames as the background.
If the difference between the current frame and the background exceeds a threshold → it's a moving object.
Equation:
ini
Copy
Edit
Background_t = α * Frame_t + (1 - α) * Background_(t-1)
Where:
α is the learning rate (between 0 and 1).
Pros: Fast and simple.
Cons: Not robust to sudden illumination or dynamic background (e.g., trees moving).
b. Gaussian Mixture Model (GMM)
Models the background of each pixel as a mixture of Gaussians.
If a new pixel value doesn't fit into the existing Gaussian models → it's considered foreground.

Equation:
java
Copy
Edit
P(X) = Σ (w_i * η(X, μ_i, σ_i²)) for i = 1 to K
Where:
w_i is the weight of the i-th Gaussian,

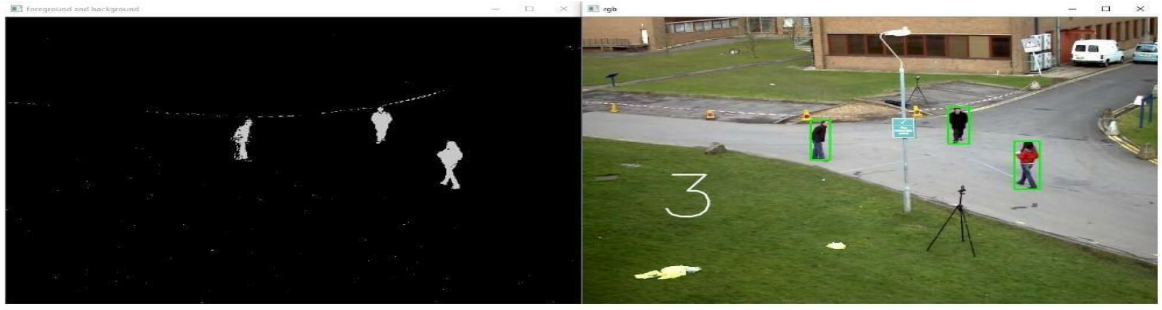
μ_i, σ_i² are the mean and variance,

η() is the Gaussian probability density.
Pros: Handles multi-modal backgrounds (e.g., flickering screen).
Cons: Computationally more expensive.
c. KNN-Based Background Subtraction
Uses a non-parametric model by storing recent history of pixel values.
For a given pixel, checks how many previous values are "close".
If not enough similar values → it's foreground.
Pros: Robust to noise, illumination changes.
Cons: Slower than GMM.
2. Optical Flow
Theory:
Optical flow estimates the motion of objects by calculating the apparent movement of brightness patterns between consecutive frames.




Equation:
mathematica
Copy
Edit
I(x, y, t) = I(x + dx, y + dy, t + dt)
Where:
I is the intensity of pixel,
dx, dy are spatial movements,
dt is the time difference.
a. Lucas-Kanade (Sparse)
Assumes small motion and locally constant flow.
Solves using least squares in a neighborhood.
b. Farneback (Dense)
Estimates a polynomial expansion of the neighborhood and computes flow for all pixels.
Pros: Can detect motion without background modeling.
Cons: Sensitive to noise and camera motion.

3. Frame Differencing
Theory:
Calculates the absolute difference between consecutive frames.



Significant differences indicate motion.
Equation:
ini
Copy
Edit
Diff = |Frame_t - Frame_(t-1)|
Pros: Extremely fast and simple.
Cons: Can miss stationary or slow-moving objects; sensitive to noise.

4. Deep Learning-Based Methods
These methods use trained neural networks for robust motion detection and classification.

a. YOLO (You Only Look Once), SSD, Faster R-CNN

Not designed specifically for motion, but detect objects in each frame.

Combine with tracking to detect movement.

Architecture Example (YOLO):

One pass of CNN detects bounding boxes and class probabilities.

Detect + Compare box positions in frames → motion.

Pros: High accuracy, supports object classification.
Cons: Requires GPU, slower without acceleration.

b. Two-Stream CNNs
One stream processes static frames (RGB),

Other stream processes optical flow (motion).
Architecture:
Both branches are CNNs; outputs are combined for final prediction.
Pros: Simultaneously learns motion and appearance.
Cons: Needs computation of optical flow (time-consuming).
c. 3D CNNs
Performs 3D convolutions across spatial and temporal axes.
Can find patterns across multiple frames.
Example:
I3D, C3D networks utilized in video understanding.
Pros: Learns spatio-temporal complex patterns.
Cons: Needs large datasets and compute.
d. ConvLSTM / LSTM + CNN
Couple of CNN for spatial features with LSTM for temporal dependencies.
Good for sequential prediction and motion detection.
Pros: Holds long-term dependencies.
Cons: Maturity of training is high.
5. Thermal and Infrared Detection
Operates under low-light or night conditions.
Performs background subtraction on temperature data rather than RGB.
Pros: Can operate in the dark.
Cons: Less resolution; requires infrared cameras.
6. Event-Based Vision (Neuromorphic Cameras)
Rather than full images, these cameras transmit only when a pixel changes.
Creates asynchronous "events".
Pros: Very fast and energy-efficient.
Cons: Needs specialized hardware and algorithms.
7. Hybrid Methods
Mix several methods to take advantage of their respective strengths.
Example:
GMM for first motion mask + YOLO for object detection + Deep SORT for tracking.

4. Importance of Object Recognition

Object recognition is the most important tool to enable machines to view and make sense of the world visually. It is what a system employs to recognize and categorize objects within images or videos. The importance of object recognition can be highlighted by the following main points:

1. Enhancing Automation and Safety

In autonomous driving, object recognition helps vehicles recognize pedestrians, traffic signals, other vehicles, and obstructions. This becomes critical for making decisions and avoiding collisions, making driving safer and more efficient without any human involvement.

2. Smart Surveillance Systems
Contemporary security setups use object recognition to automatically spot intruders or unusual behavior. By identifying individuals, vehicles, or unattended items, these systems lessen the need for continuous human oversight and can respond to threats with greater speed.

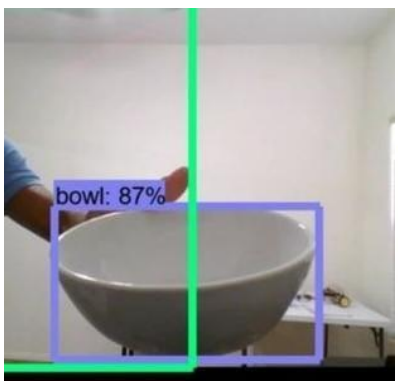
3. Human-Computer Interaction
Object recognition enhances interfaces that interact with our physical environment, like gesture-based controls, augmented reality (AR), and virtual assistants. It allows devices to comprehend real-world scenarios and react appropriately, leading to more intuitive user experiences.

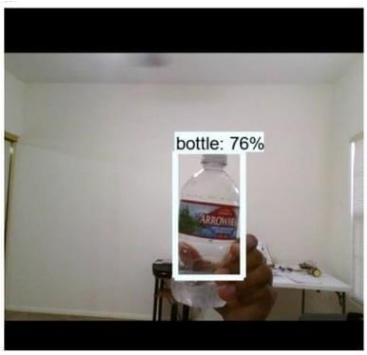
4. Medical Imaging and Diagnostics
Within the medical field, object recognition plays a crucial role in identifying tumors, lesions, or other abnormalities in medical images such as X-rays, MRIs, or CT scans. This automated detection assists doctors in delivering quicker and more precise diagnoses, particularly in urgent or high-volume situations.

5. Retail and Inventory Management
In retail, object detection finds applications in self-service kiosks, inventory tracking, and preventing losses. It's used to register products on shelves or at the checkout, which both streamlines operations and boosts customer convenience.
6. Environmental Monitoring and Agriculture
Plant, insect, or wildlife identification by drone or satellite imaging helps with precision agriculture and conservation. Farmers and researchers can monitor crop health, detect disease, or track animal migration with high efficiency.
7. Robotics and Industrial Applications
Object recognition in robotics and manufacturing helps machines identify parts, assemble components, or inspect products on assembly lines. It makes automation, quality inspection, and low-cost production possible.

8. Information Retrieval and Search
Object recognition powers devices like Google Lens or image-based search engines so that the user can click or upload a photo to gain access to related information. Object recognition closes the gap between the virtual and physical worlds so that information can be accessed quicker.

5. RESULT
In this study, we developed an object detection system by integrating the YOLO (You Only Look Once) algorithm with OpenCV's deep learning module. The system leverages a pre-trained YOLO model to identify and categorize multiple objects within an input image. After processing the image into a suitable "blob" format, it's fed into the YOLO model, which then generates predictions including bounding boxes, class probabilities, and confidence scores for objects present in the image. To enhance the accuracy of detections, we implemented a confidence threshold of 0.5 to filter out weaker predictions. Additionally, Non-Maximum Suppression (NMS), with a threshold of 0.4, is applied to effectively eliminate overlapping and redundant bounding boxes. Finally, the calculated bounding boxes are drawn onto the image, each marked with the predicted object class using color-coded labels for clear visualization.





The output is both displayed on the screen and saved as a picture file named "object-detection.jpg." The system performs well with common objects such as persons, vehicles, animals, and household items, based on classes defined in the model's configuration. The code demonstrates that YOLO, even when run under OpenCV without GPU acceleration, can perform rapid and good object detection on still images.This underscores its applicability to a broad class of real-time and offline applications, particularly where computation resources are constrained.

Future Work

Although the current implementation is promising for static image-based object detection, there are several areas in which this work can be extended further. The next logical step would be to take the system's capability for real-time object detection from video streams to the next level. Processing all the frames of a video or webcam feed, the system can be applied to live surveillance, activity monitoring, and real-time inspection. That would also include incorporating object tracking features to track objects between frames, allowing for more profound temporal comprehension of motion and action.

The model offers further avenues for optimization, potentially by upgrading to more advanced iterations of YOLO, such as YOLOv5 or YOLOv8. These newer versions often provide superior detection performance and improved computational efficiency. Alternatively, tailoring the model by training it on in-house datasets could significantly enhance its effectiveness for specific application domains like medical imaging, agriculture, or industrial robotics. This focused optimization would enable the system to recognize domain-specific objects that fall outside the purview of generic YOLO datasets.

Another promising approach involves deploying the model on edge and mobile hardware. This can be achieved by converting it into lightweight formats such as ONNX, TensorFlow Lite, or OpenVINO. Such optimization would render the detection system more portable and feasible for use in low-resource environments. Finally, developing a user-friendly interface or dashboard for tasks like image uploading, displaying results, and exporting reports would greatly boost the system's usability and adoption. Incorporating these future enhancements would transform the object detection framework into a highly robust and flexible tool applicable across numerous real-world scenarios.

CONCLUSION

This Python script effectively demonstrates real-time object detection using the YOLO deep learning model integrated with OpenCV. After loading pre-trained weights and parameters, the script processes the input image, extracts features through a neural network, and efficiently identifies and classifies objects within the scene.

The strategic use of functions like cv2.dnn.blobFromImage() and net.forward() efficiently prepares the image for detection and retrieves outputs from the neural network. Furthermore, applying Non-Maximum Suppression (NMS) plays a crucial role in eliminating redundant bounding boxes, thereby minimizing false positives. The system accurately detects a wide range of object classes and visually annotates them with bounding boxes and class labels.

This project showcases the practical application of deep learning models for visual recognition, making it suitable for integration into real-time video streams, surveillance systems, and smart automation applications. It underscores YOLO's effectiveness in achieving both speed and detection accuracy, solidifying its position as a valuable tool for real-world object recognition tasks.

REFERENCES

[1] Mahalakshmi, G., Khajahussen, S., Harathi, P., Nagarjuna, U., Harshini, M., & Govindu, B. S. (2024). DEVELOPING A YOLO BASED OBJECT OBSERVATION APPLICATION USING OPEN CV. MATERIAL SCIENCE, 23(04).

[2] Saha, S., Shil, S., Sutradhar, K., & Sengupta, Y. Object Detection Using OpenCV. In Embedded Artificial Intelligence (pp. 252-264). Chapman and Hall/CRC.

[3] Parjane, P., Raktate, G., Shelar, K., Wakchuare, R., More, S., & Kale, J. (2024, December). Enhancing Real-Time Object Detection With Advanced YOLOv9 and OpenCV in Python. In 2024 International Conference on Decision Aid Sciences and Applications (DASA) (pp. 1-5). IEEE.

[4] Mohan, G. K., Shaik, M. F., Babu, G. U., Prasanna, R. G. V., Rao, P. V., & Raja, I. R. (2025). Revolutionizing object recognition beyond CNN and YOLO with deep learning breakthrough. In Emerging Trends in Computer Science and Its Application (pp. 26-32). CRC Press.

[5] Gupta, M., Verma, A., & Rana, K. (2024). Drowsiness Detection System: combining YOLO pytorch, and python for enhanced road safety.

[6] Sudeshna, M. P., Karthika, G. K., Prathyusha, K., & Indhu, K. (2025). OBJECT DETECTION AND RECOGNITIONS USING WEBCAMS WITH VOICE USING YOLO ALGORITHM. environments, 25(01)..
[7] Tiwari, A., Kumar, E. S., Mishra, A., Sehgal, J., & Samita, E. S. (2024, April). Automatic Number Plate and Speed Detection using YOLO and CNN. In 2024 IEEE 9th International Conference for Convergence in Technology (I2CT) (pp. 1-6). IEEE.
[8] Uke, S., Junghare, P., Kenjale, S., Korade, S., & Kothwade, A. (2024, December). Comprehensive Real-Time Intrusion Detection System Using IoT, Computer Vision (OpenCV), and Machine Learning (YOLO) Algorithms. In 2024 4th International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS) (pp. 1680-1689). IEEE..
[9] Krishnaveni, A., & Arvinth, K. A. (2025, February). Smart Animal Monitoring System for Farms using YOLO and OpenCV. In 2025 International Conference on Electronics and Renewable Systems (ICEARS) (pp. 1969-1974). IEEE.
[10]Darmadi, D., Pratikso, P., & Rahmat, M. (2024). Traffic counting using YOLO version-8 (case study of jakarta-cikampek toll road). Astonjadro, 13(1), 115-124.