© APR 2026 | IRE Journals | Volume 9 Issue 10 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I10-1716506

Implementation Of Object Detection Using OpenCV and Python

HIMANSHU RAI[1], VANSH KHATANA[2], DR. MOHD DANISH[3]
[1, 2, 3]*Department of Data Science (DDCS), GNIOT College, Greater Noida, India*

Abstract- ***Real-time object detection has emerged as a critical application in computer vision, enabling intelligent systems to perceive and interact with their environment. This study presents a comparative analysis of lightweight deep learning models, namely MobileNet-SSD and YOLO-based architectures, for real-time object detection using OpenCV and Python. The proposed system processes live video streams by leveraging OpenCV's DNN module to perform efficient inference on CPU hardware. The performance of the models is evaluated based on frame processing speed (FPS), detection accuracy, and computational efficiency. Experimental results demonstrate that MobileNet-SSD achieves higher processing speed, making it suitable for resource-constrained environments, while YOLO-based models provide improved detection accuracy. The study highlights the trade-off between speed and accuracy and provides insights into selecting appropriate models for real-time applications such as surveillance, automation, and smart monitoring systems.***

I. INTRODUCTION

Object detection is a fundamental task in computer vision that involves identifying and localizing objects within images or video streams. With the rapid advancement of deep learning techniques, object detection models have significantly improved in terms of both accuracy and computational efficiency.

Traditional approaches such as Haar Cascades and Histogram of Oriented Gradients (HOG) were limited in performance and scalability. In contrast, modern deep learning–based models such as Single Shot Detectors (SSD) and You Only Look Once (YOLO) frameworks have enabled real-time detection capabilities.

This research focuses on developing a real-time object detection system using OpenCV and Python, while analyzing the performance of lightweight models on CPU-based systems. The objective is to evaluate the trade-offs between speed and accuracy and to identify suitable models for real-time deployment without requiring GPU acceleration.

II. LITERATURE REVIEW

Recent advancements in object detection have been driven by deep learning–based approaches. YOLO introduced a unified architecture that performs object detection in a single forward pass, significantly improving speed compared to traditional multi-stage detectors. SSD further enhanced detection efficiency by eliminating region proposal steps and enabling faster inference.

Lightweight architectures such as MobileNet-SSD have been specifically designed for resource-constrained environments, offering a balance between computational efficiency and detection accuracy. These models use depthwise separable convolutions to reduce computational cost while maintaining acceptable performance.

III. METHODOLOGY

A. System Overview

The proposed system captures live video streams from a camera and processes each frame using deep learning models integrated with OpenCV's DNN module. The system is designed to be lightweight and efficient, ensuring smooth execution on standard CPU hardware without requiring specialized accelerators.

B. Workflow

Video frame acquisition from webcam
Frame preprocessing (resizing and normalization)
Conversion into blob format suitable for neural networks Forward pass through the neural network

Extraction of predictions (class labels and bounding boxes) Visualization using bounding boxes and labels Performance evaluation using FPS and latency This pipeline ensures continuous detection and real-time feedback, making the system suitable for dynamic environments.

C. Models Used

MobileNet-SSD: Optimized for high-speed detection on CPU systems with minimal computational overhead YOLO-based Model: Provides improved accuracy with moderate speed, suitable for applications requiring better precision

Both models are selected due to their balance between performance and computational efficiency.

D. Dataset

The models are pretrained on standard datasets such as COCO and PASCAL VOC, which include a wide range of object categories like humans, vehicles, and everyday objects. These datasets ensure robust generalization across different real-world scenarios.

IV. RESULTS AND ANALYSIS

The system was evaluated based on frame processing speed, detection accuracy, and overall responsiveness under different conditions.

Model FPS (CPU)	Accuracy	Performance
MobileNet-SSD	20–25 FPS	Moderate High Speed
YOLO (Tiny/Lightweight)	12–18 FPS	Higher Balanced

Key Observations:

MobileNet-SSD provides smooth real-time performance with low latency and consistent frame processing

Environmental factors such as lighting conditions, object size, and motion significantly influence detection performance

The system maintains stable performance even under moderate computational constraints.

V. DISCUSSION

The results indicate a clear trade-off between speed and accuracy in real-time object detection systems. Lightweight models are essential for deployment on CPU-based devices where computational resources are limited.

MobileNet-SSD is suitable for applications requiring high-speed processing, such as traffic monitoring and basic surveillance. On the other hand, YOLO-based models are preferred in scenarios where detection precision is critical, such as security systems and advanced analytics.

VI. CONCLUSION

This study presents a real-time object detection system using OpenCV and Python, along with a comparative analysis of lightweight deep learning models. The results demonstrate that efficient object detection can be achieved on CPU hardware without requiring high-end GPUs.

The findings highlight the importance of selecting appropriate models based on application needs, balancing speed and accuracy. The proposed system can be effectively utilized in surveillance, automation, and intelligent monitoring systems.

VII. FUTURE WORK

Future enhancements may include:

Integration of advanced models such as YOLOv8 for improved accuracy

Optimization using GPU acceleration and TensorRT for faster inference Deployment on embedded systems like Raspberry Pi and NVIDIA Jetson

Incorporation of additional evaluation metrics such as precision, recall, and mAP

Development of adaptive systems that adjust performance based on real-time conditions.

REFERENCES

[1] J. Redmon et al., “You Only Look Once: Unified, Real-Time Object Detection,” CVPR, 2016.

[2] W. Liu et al., “SSD: Single Shot MultiBox Detector,” ECCV, 2016. OpenCV Documentation – DNN Module

[3] Bochkovskiy et al., “YOLOv4: Optimal Speed and Accuracy,” 2020