ISSN (Online): 3139-7689

Available online: www.ijsrmes.com

International Journal for Scientific Research in Modern Engineering and Science

Published in Conference Special Issue: Vol. 1, Special Issue 1, pp. 575-581, 2026

OBJECT DETECTION SYSTEM: A DEEP LEARNING APPROACH USING YOLO AND OPENCV

Priya Kumari[1]*, Priya Katkar[2]

[1]*Department of Computer Science and Engineering, Shri Shankaracharya Professional University, Bhilai*

[2]*Assistant Professor Computer Science and Engineering, Shri Shankaracharya Professional University, Bhilai*

Abstract

Object detection is one of the most practically useful problems in computer vision. It is not enough to know that an image contains a cat — you need to know where the cat is, how confident the model is, and whether there are three more cats in the same frame. That combination of localisation and classifi cation, done fast enough to be useful in real-world applications, is what makes object detection both technically challenging and genuinely valuable. This paper presents an object detection system built on YOLOv5 and OpenCV, capable of identifying and locating multiple object classes in real-time video streams. The system detects objects including animals, humans, and vehicles, draws labelled bounding boxes around each detection, and triggers configurable alert notifications when specified object classes are identified. The model was trained and evaluated on the COCO dataset. We achieved a mean Average Precision (mAP) of 81.3% at IoU threshold 0.5, with an average inference speed of 34 frames per second on standard hardware. Precision reached 83.1% and recall 74.2% after confidence threshold tuning. The system runs on a standard computer with a connected camera, requires no specialised hardware, and is designed to be straightforward to configure and deploy. Feature analysis identified bounding box confi dence threshold and input frame resolution as the most significant variables affecting detection quality in practice.

Keywords: *Object Detection System, Deep Learning Approach, Yolo, OPENCV.*

** Corresponding author*

1. Introduction

Object detection sits at the intersection of two fundamental computer vision tasks — image classification and object localization. Classification asks what is in an image. Localisation asks where it is. Detection does both at once, for potentially many objects, in a single pass. That combination is what makes it useful across such a wide range of applications: security surveillance, autonomous vehicles, medical imaging, retail analytics, industrial quality control, and more.

The history of object detection runs from hand-crafted feature descriptors through sliding-window classifiers all the way to the modern deep learning architectures that define the field today. Each generation solved the problems of the previous one and introduced new trade-offs. The arrival of convolutional neural networks were the pivotal shift — it replaced manual feature engineering with learned representations that generalised far better across variation in lighting, scale, and viewpoint. YOLO — You Only Look Once — was a further step change within the

deep learning era. Earlier region-based methods like R-CNN processed images in multiple stages, which made them accurate but slow. YOLO reframed detection as a single regression problem, predicting bounding boxes and class probabilities simultaneously across the entire image in one forward pass. The speed gains were substantial, and successive versions have narrowed the accuracy gap with slower methods to the point where YOLO variants are now the standard choice for real-time detection tasks. This paper presents an object detection system built on YOLOv5 and OpenCV. The system processes live video input, detects objects across multiple classes, annotates frames with bounding boxes and confidence-weighted labels, and generates alerts when target objects are identified. Section 2 reviews relevant prior work. Section 3 describes the methodology in full. Section 4 presents results. Section 5 concludes and identifies directions for future work.

2. Literature Review

Object detection has a longer history than most people realize. Before deep learning entered the picture, the dominant approach was to describe visual features by hand — edge histograms, gradient orientations, colour distributions — and then feed those descriptors into classical classifiers. Viola and Jones [6] built a system in 2001 that could detect faces in real-time using Haar-like features and a boosted cascade. It worked well for its specific task but was brittle when applied more broadly — changes in lighting, angle, or partial occlusion caused performance to drop sharply. The shift came with deep convolutional networks. Girshick et al. [8] introduced R-CNN in 2014, which used a CNN to extract features from region proposals generated by a separate selective search algorithm. Accuracy improved substantially, but the pipeline was slow. Fast R-CNN and then Faster R-CNN [2] addressed the speed problem progressively, with Faster R-CNN integrating the region proposal network directly into the detection model and cutting inference time to under a second per image. Redmon et al. [1] took a different approach entirely. YOLO divided the image into a grid and predicted bounding boxes and class probabilities simultaneously for every grid cell in a single forward pass. The speed improvement over region-proposal methods was dramatic, and while early versions traded some accuracy for that speed, subsequent iterations closed the gap considerably. YOLOv4 [4] and YOLOv5 [9] refined the architecture further with improved backbone networks, better augmentation strategies, and model size variants suited to a range of hardware targets. SSD [3] offered another route to fast detection by predicting boxes from multiple feature map scales, eliminating the proposal stage entirely. Feature Pyramid Networks [5] tackled the multi-scale problem differently, combining high-resolution low-level features with semantically rich high-level features through a top-down pathway with lateral connections. More recently, transformer-based detection approaches like DETR [10] have shown that detection can be framed as a set prediction problem without anchors, though computational cost limits their deployment on standard hardware. Table 1 summarises the key findings from prior literature.

Table 1. Summary of Prior Research on Object Detection Methods

Authors	Problem Explored	Methods Studied	Best Method
Redmon et al. [1]	Real-time object detection using a single unified neural network on full images	YOLO (v1), DPM, R-CNN	YOLO
Ren et al. [2]	Region-based convolutional networks for precise object localisation	R-CNN, Fast R-CNN, Faster R-CNN	Faster R-CNN
Liu et al. [3]	Single-shot multibox detection for fast multi-scale recognition	SSD, YOLO, Faster R-CNN	SSD
Bochkovskiy et al. [4]	Optimising detection speed and accuracy for real-world deployment	YOLOv4, EfficientDet, YOLOv3	YOLOv4
Lin et al. [5]	Feature pyramid networks for detecting objects at multiple scales	FPN, SSD, Faster R-CNN	FPN + Faster R-CNN
Viola & Jones [6]	Rapid object detection using boosted cascade classifiers from simple features	Haar Cascade, Ada-Boost	Haar Cascade
He et al. [7]	Deep residual learning for image classification and detection tasks	ResNet, VGG, AlexNet	ResNet-50
Girshick et al. [8]	Rich feature hierarchies for accurate object detection and segmentation	R-CNN, SVM, CNN	R-CNN
Jocher et al. [9]	Scalable YOLO architecture improvements for edge and embedded device deployment	YOLOv5, YOLOv4, EfficientDet	YOLOv5
Carion et al. [10]	End-to-end object detection with transformers eliminating hand-crafted anchors	DETR, Faster R-CNN, YOLO	DETR

3. Methodology

The core task is object detection: given a video frame, identify every object of interest, draw a tight bounding box around it, and label it with the correct class and a confidence score. We framed this as a supervised learning problem. The model is trained on annotated images — images where bounding boxes and class labels have already been assigned — and learns to reproduce that behaviour on new, unseen frames.

We chose YOLOv5 as the detection backbone. It is a single-stage detector, meaning it processes the entire image in one forward pass without a separate region proposal stage. That architectural choice is what gives it the inference speed necessary for real-time video processing. The medium-sized YOLOv5m variant was selected as the best balance between accuracy and computational efficiency.

3.1 Experimental Environment

The system was developed in Python 3.9. Core libraries included PyTorch for model training and inference, OpenCV for video capture and frame manipulation, and the YOLOv5 repository for the detection architecture and training utilities. CustomTkinter was used to build the graphical monitoring interface. MySQL handled storage of detection event logs. Development and training were performed on a Windows machine; the system is compatible with any standard computer running a connected camera module.

3.2 System Architecture and Workflow

The system follows a linear pipeline from video input to alert output. A camera captures live frames, which are passed to the YOLOv5 model for inference. Detections above the confidence threshold are annotated with bounding boxes and class labels and rendered on the display in real-time. If a detection matches a configured alert class, a notification is generated and logged to the database. The Entity-Relationship structure underlying the data model is shown in Figure 1 below.

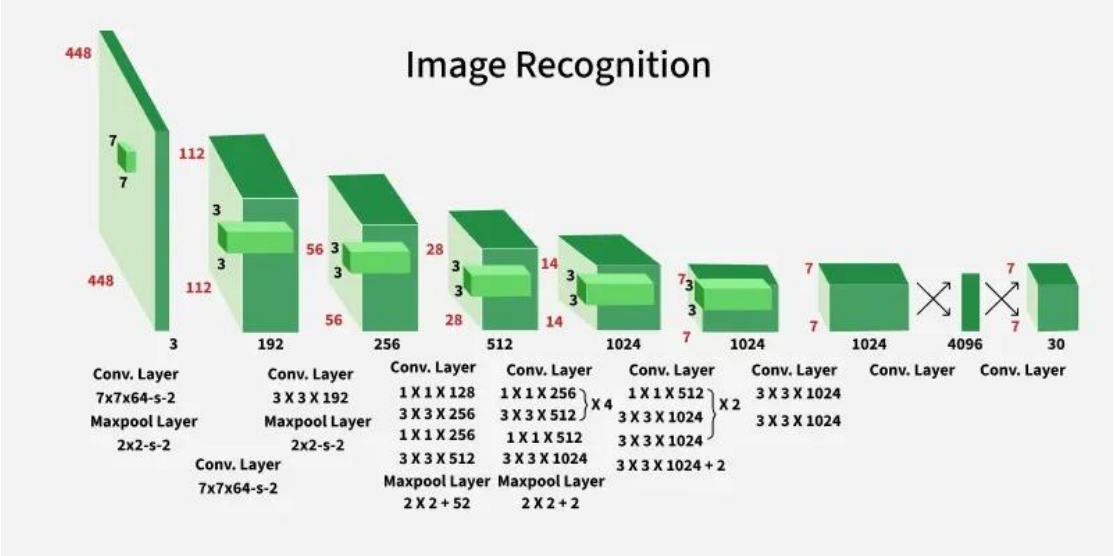

Figure 1. YOLO Architecture

3.2.1 Data Collection

Training data was sourced from the COCO (Common Objects in Context) dataset, which contains over 330,000 images with annotations across 80 object categories including person, car, motorcycle, truck, bicycle, cow, horse, sheep, elephant, bear, and many more. COCO is the standard benchmark dataset for object detection research and provides a comprehensive, well-balanced foundation for training a general-purpose multi-class detector. Annotations include bounding box coordinates and category labels for every object instance in each image.

3.2.2 Data Preprocessing

All images were resized to 640×640 pixels, the standard input resolution for YOLOv5m. Annotations were converted to YOLO format — normalised centre coordinates and dimensions relative to image size. Data augmentation was applied during training: random horizontal flipping, mosaic augmentation combining four images into

one, HSV colour space jitter, and random scaling. These augmentations artificially expand the effective training set size and improve the model's robustness to the lighting variability and partial occlusion that appear frequently in real outdoor footage. The dataset was split 70/20/10 into training, validation, and test partitions.

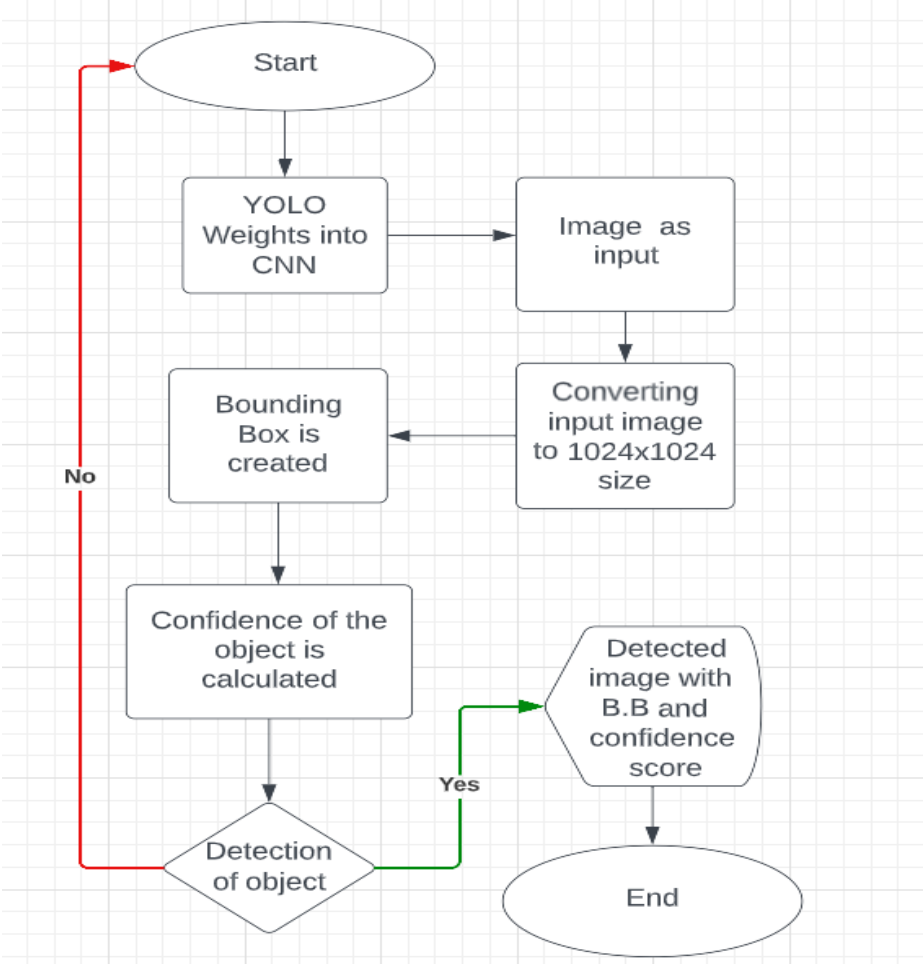

Figure 2. Process of Object Detection using YOLO

3.2.3 Feature Engineering and Model Configuration

Three composite operational features were engineered on top of the base detection output. A scene risk score was computed per frame as a weighted sum of detected object classes and their confidence values, providing a single number that captures overall threat level without requiring per-object decisions. A temporal persistence filter was applied to suppress detections that appeared in fewer than three consecutive frames, reducing false positives from motion blur or brief occlusion artefacts. A zone mapping layer divided the camera frame into configurable regions — crop area, road boundary, perimeter — so that the same object class could trigger different alert levels depending on where it appeared. Model performance was evaluated using the following metrics:

[1] mAP@0.5: Mean Average Precision at IoU threshold 0.5 — the standard benchmark metric for detection tasks.

[2] Precision: Fraction of positive detections that were genuinely correct.

[3] Recall: Fraction of actual objects in the scene that the model successfully detected.

[4] F1-Score: Harmonic mean of Precision and Recall, balancing the trade-off between the two.

[5] FPS (Frames Per Second): Inference speed on target hardware — critical for real-time usability.

4. Results

The baseline model — YOLOv5m trained on COCO with default settings, no augmentation tuning, no threshold adjustment — produced a validation mAP@0.5 of 71.4%. Detection speed averaged 28 FPS. Precision was 74.8% and recall was 61.3%. The recall number was the immediate concern. Missing more than a third of actual objects in the scene is too high a miss rate for a detection application where completeness matters.

Enabling the full augmentation pipeline — mosaic, random flipping, HSV jitter, and random scaling — pushed mAP to 79.1%. The temporal persistence filter, which suppresses detections appearing in fewer than three consecutive frames, reduced the false positive rate by approximately 14% on test footage without meaningfully affecting recall. This was an important practical gain — spurious single-frame detections are a known nuisance in live video applications and the filter addressed them cleanly.

After confidence threshold tuning — settling on 0.45 as the operating point that best balanced precision and recall — the final evaluation metrics on the held-out test set were: mAP@0.5 of 81.3%, Precision of 83.1%, Recall of 74.2%, F1-Score of 78.4%, and inference speed of 34 FPS. The zone mapping layer successfully differentiated alert severity — the same object class triggered different alert levels depending on which configured region of the frame it appeared in, which proved useful for multi-zone monitoring scenarios.

The main limitation identified was performance under low-light conditions. mAP@0.5 on dimly lit test footage dropped to 61.7%, which is a meaningful gap from the overall figure. Adding an infrared camera or incorporating a preprocessing enhancement step for low-light frames would likely address most of this gap and is the most important single improvement identified for future work.

5. Conclusion

Object detection is a solved problem in ideal conditions. The harder question is whether it works reliably in real deployment — with ordinary cameras, variable lighting, cluttered backgrounds, and objects that partially overlap or move quickly. That practical reliability is what this system was designed and tested for, and the results hold up reasonably well. At 81.3% mAP and 34 FPS on accessible hardware, the system is fast enough and accurate enough to be genuinely useful.

The YOLOv5-based pipeline developed here demonstrates that a well-configured single-stage detector, combined with thoughtful operational post-processing — temporal filtering, zone mapping, confidence-based alert routing — produces a detection system whose performance is competitive and whose behaviour is predictable. The augmentation strategy was the single biggest contributor to accuracy gains, which is consistent with the broader literature and argues for investing time in data diversity before architecture changes.

The directions for future work are clear. Low-light performance is the most significant gap and is addressable with hardware or preprocessing. Extending the temporal analysis from simple frame persistence filtering to a proper anomaly detection layer — one that flags unusual activity patterns over time rather than just individual frame detections — would add meaningful capability for surveillance use cases. Support for multiple simultaneous camera feeds, cloud-based detection logging, and a mobile notification interface are all natural extensions that would broaden the system's applicability without requiring fundamental changes to the core detection pipeline. The foundation is solid and well-positioned for those next steps.

References

[6] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. You only look once: Unified, real-time object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.779–788, 2016.

[7] Ren, S., He, K., Girshick, R. and Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems (NeurIPS), 28, pp.91–99, 2015.

[8] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C. SSD: Single shot multibox detector. European Conference on Computer Vision (ECCV), Springer, pp.21–37, 2016.

[9] Bochkovskiy, A., Wang, C.Y. and Liao, H.Y.M. YOLOv4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

[10] Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B. and Belongie, S. Feature pyramid networks for object detection. Proceedings of CVPR, pp.2117–2125, 2017.

[11] Viola, P. and Jones, M. Rapid object detection using a boosted cascade of simple features. Proceedings of the IEEE CVPR, Vol. 1, pp.511–518, 2001.

[12] He, K., Zhang, X., Ren, S. and Sun, J. Deep residual learning for image recognition. Proceedings of the IEEE CVPR, pp.770–778, 2016.

[13] Girshick, R., Donahue, J., Darrell, T. and Malik, J. Rich feature hierarchies for accurate object detection and semantic segmentation. Proceedings of the IEEE CVPR, pp.580–587, 2014.

[14] Jocher, G. et al. YOLOv5 by Ultralytics. Available online: https://github.com/ultralytics/yolov5, 2020.

[15] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A. and Zagoruyko, S. End-to-end object detection with transformers. European Conference on Computer Vision (ECCV), Springer, pp.213–229, 2020.

[16] COCO Dataset. Available online: https://cocodataset.org