WWW.IJRPR.COM

**International Journal of Research Publication and Reviews**

Journal homepage: www.ijrpr.com ISSN 2582-7421

# Object Detection and Counting

***Mr. Mohan Rajadurai[1], Esakki Kannan P[2], Arun S[3], Chandru S[4], Akash V[5]***

Sri Shakthi Institute of Engineering and Technology, Coimbatore, 641062, India.

**A B S T R A C T :**

The Object Counting System leverages the power of computer vision and deep learning to detect and count objects in images and real-time video streams. Traditional object counting methods often fall short when dealing with variations in object shape, size, orientation, occlusion, and environmental conditions. This project addresses these challenges using advanced neural network architectures such as YOLOv5 and SSD, known for their speed and accuracy in object detection tasks.The core idea of this project is to build a scalable and real-time object counting solution that can be applied across various domains—such as traffic surveillance, crowd monitoring, inventory management, and smart farming. The system uses a deep learning model pre-trained on large datasets (like COCO) and fine-tuned for specific use cases. It detects and counts objects frame-by-frame and displays the results through a simple, intuitive web interface powered by Flask.

**Keywords**: Object Detection, Object Counting, YOLOv5, Deep Learning, Real-Time Detection, Computer Vision, Convolutional Neural Networks (CNN)

## 1. Introduction

The field of computer vision has seen immense growth in the last decade, fueled by advancements in deep learning, availability of large datasets, and powerful computational hardware. One critical task in computer vision is **object detection and counting**, which involves identifying specific objects in an image or video and quantifying their occurrences. This has a wide range of practical applications, from tracking the number of vehicles on a road to monitoring footfall in a retail store or counting fruits on trees in an agricultural field.

Traditionally, object counting was carried out manually or through basic image processing methods such as edge detection, background subtraction, or contour analysis. While useful, these methods are limited in their ability to adapt to real-world challenges such as overlapping objects, varying lighting conditions, and dynamic backgrounds. With the introduction of **Convolutional Neural Networks (CNNs)** and advanced object detectors like **YOLO (You Only Look Once)** and **SSD (Single Shot MultiBox Detector)**, object detection has become faster and significantly more accurate, paving the way for automated, real-time object counting systems.

This project explores the design and implementation of an **Object Counting System** using deep learning-based object detectors. The goal is to accurately detect and count objects from both static images and live video streams, and to present the results to users in an intuitive and interactive format.

## 2. Literature Review

The field of object detection and counting has evolved significantly in recent years, driven by advancements in computer vision and deep learning. Earlier methods relied heavily on classical image processing techniques such as background subtraction, edge detection, and blob analysis. However, these traditional approaches were limited in their ability to handle occlusion, varying object sizes, lighting changes, and complex backgrounds. With the rise of deep learning, especially Convolutional Neural Networks (CNNs), object detection and counting systems have achieved remarkable accuracy and generalizability across diverse environments.

## 3. Methodology

The proposed methodology for the Object Counting and Detection System is designed to ensure high accuracy, real-time performance, and flexibility for various use cases. It leverages a deep learning-based object detection model (YOLOv5) to detect and count objects from images or video streams and presents results through a user-friendly web interface. The methodology involves multiple stages, including data acquisition, model selection, object detection, counting logic, backend processing, and frontend display.

### *3.1 System Overview*

The system follows a modular pipeline, consisting of the following major components:

1. *Data Acquisition* – Input images or live video stream from a webcam or uploaded files.
2. *Preprocessing* – Resizing and normalization of input frames.
3. *Object Detection using YOLOv5* – Detection of objects with bounding boxes and class labels.
4. *Object Counting* – Incremental count based on detection results.
5. *Visualization* – Real-time rendering of the detection and count overlay.
6. *Backend Processing (Flask API)* – Integration of model inference with HTTP endpoints.

### *3.2 Data Collection and Preprocessing*

*The input data for the system can be:*

- *Static images (JPG, PNG),*
- *Real-time video streams (from webcam or IP camera),*
- *Pre-recorded video files (MP4, AVI).*

### *3.3 Object Detection with YOLOv5*

*The core detection mechanism uses YOLOv5, a state-of-the-art single-stage object detector. YOLOv5 divides an image into a grid and for each cell predicts bounding boxes, confidence scores, and class probabilities.*
*YOLOv5 is chosen for:*

- *Its high inference speed (suitable for real-time),*
- *Compact model size (can run on CPU or GPU),*
- *Pretrained models on COCO dataset for 80+ object classes,*
- *Flexibility to retrain on custom datasets if needed.*

### *3.4 Object Counting Logic*

*Once the objects are detected, the counting logic is applied. There are two approaches:*

- *Frame-wise Counting: Count all detected objects in a single frame. This is suitable for static images.*
- *Tracking-based Counting: Maintain unique object IDs across frames to avoid double-counting (e.g., using Deep SORT or OpenCV trackers). This is essential for video streams*

### *3.5 Backend Integration (Flask API)*

*The backend is built using Flask, a lightweight Python web framework. It provides RESTful APIs for:*

- *Uploading images or videos,*
- *Triggering object detection,*
- *Returning results as JSON and annotated images.*

## 4.Implementation Details

The proposed object counting and detection system was implemented using Python, leveraging popular deep learning frameworks such as PyTorch for model integration and Flask for backend development. The frontend interface was built using HTML, CSS, and JavaScript, allowing users to upload media files, perform real-time detection, and visualize the results with object counts. This section discusses the specific tools, libraries, frameworks, and development environment used in the project.

### *4.1 Development Environment*

The development environment was configured as follows:

- Operating System: Windows 10 / Ubuntu 20.04
- Programming Language: Python 3.8
- Deep Learning Framework: PyTorch (via Ultralytics YOLOv5 repository)

### *4.2 Flask Backend API*

The backend is responsible for handling user input, performing object detection, and returning annotated outputs and count data. Flask endpoints were created as follows:

- /upload: Accepts image or video uploads
- /detect: Performs detection and returns image + count
- /video_feed: Streams live camera feed with detections

### *4.3 Frontend Design*

The frontend was built with a clean interface using HTML and Bootstrap. It allows users to:

- Upload an image or select live webcam feed
- View detected objects with bounding boxes
- Display object counts dynamically
- Download the processed result

### *4.4 Output and Visualization*

Detected objects are displayed with bounding boxes and class labels using OpenCV functions such as cv2.rectangle() and cv2.putText(). Counts are displayed beside the image as a summary.
An example output frame would show:

- Annotated objects (e.g., "Person", "Car")
- Count display: {Person: 4, Car: 2}

## 5.Result and Discussion

The proposed object counting and detection system was tested using both standard benchmark datasets and real-world input scenarios, such as images captured from surveillance cameras and user-uploaded video files. This section presents the experimental results, discusses model performance in various settings, and evaluates the accuracy and efficiency of the system across different object categories.

## 6. Conclusion

This research presented a deep learning–based object detection and counting system leveraging the YOLOv5 model. The proposed framework provides an end-to-end solution that integrates image/video input acquisition, real-time object detection, object counting, backend processing, and frontend visualization within a unified architecture. Designed using open-source tools such as PyTorch, Flask, and OpenCV, the system offers efficient performance and ease of deployment for various real-world applications.

***Acknowledgements***

I would like to express my sincere gratitude to my mentor, the department staff, and the Head of Department (HoD) for their invaluable guidance, support, and encouragement throughout the course of this project. Their expertise and constant assistance have been instrumental in the successful completion of this work.

**REFERENCES**

**1.** J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
   A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.

**2.** K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.
   A. Dosovitskiy et al., "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv preprint arXiv:2010.11929*, 2020.

**3.** J. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple Online and Realtime Tracking," in *IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464–3468.

**4.** OpenCV Developers, "OpenCV: Open Source Computer Vision Library," 2023. [Online]. Available: https://opencv.org/

**5.** Flask Developers, "Flask: Web Development, One Drop at a Time," Pallets Projects, 2023. [Online]. Available: https://flask.palletsprojects.com/

**6.** COCO Dataset, "Common Objects in Context," [Online]. Available: https://cocodataset.org/

**7.** M. Everingham et al., "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.