International Journal of Advance Research in Science and Engineering
Volume. No. 15, Issue No. 04, April, 2026
ISSN: 2319-8354

People Counting System Using Deep Learning and Computer Vision

M. Anbarasu M.S(SS),M.Tech(CSE),(ph.D)
Associate Professor
Department of Information Technology
Tirumala Engineering College,
Pradesh, India
anbusuriyanmtechphd@gmail.com

Yallamanda Eswar
Department of Information Technology
Tirumala Engineering College,
Andhra Pradesh, India
eswaryallamanda@gmail.com

Turlapati Aparna
Department of Information Technology
Tirumala Engineering College, Andhra
Andhra Pradesh, India
23ne5a1201college@gmail.com

Tanniru Haripriya
Department of Information Technology
Tirumala Engineering College,
Andhra Pradesh, India
haripriyatanniru4@gmail.com

Manda Jobabu
Department of Information Technology
Tirumala Engineering College,
Andhra Pradesh, India
mandajobabu@gmail.com

Abstract

In this paper, we propose and implement an automated realtime people counting system based on the YOLOv5 (You Only Look Once, Version 5) deep learning object detection system ingests live webcam feeds or pre-recorded video files, detects human subjects using a convolutional neural network (CNN) pre-trained on the COCO dataset through transfer learning, and produces a stable, frame-by-frame count of persons present in the scene. To enhance robustness and efficiency, we incorporate optimization techniques including frame resizing, frame skipping, confidence thresholding, and temporal smoothing. The entire pipeline is implemented in Python using PyTorch and OpenCV, and operates on standard CPU hardware without requiring a dedicated GPU. Experimental evaluations across diverse real-world scenarios demonstrate a detection accuracy of 85–92%, a mean Average Precision (mAP@0.5) of 82.1%, and a precision of 87.3%. The system is lightweight, cost-effective, and readily deployable in surveillance, smart campus monitoring, retail analytics, and public safety applications.

Keywords — People Counting, YOLOv5, Deep Learning, Convolutional Neural Network, Object Detection, OpenCV, COCO Dataset, Transfer Learning, Real-time Surveillance, Computer Vision

I. INTRODUCTION

The automated counting and monitoring of people in enclosed and open spaces has emerged as a critical requirement across a wide range of domains, including public safety, smart city infrastructure, retail management, and academic facilities. Accurate and real-time people counting enables informed decisions regarding crowd density control, energy management, resource allocation, and emergency evacuation planning.

Traditional people counting methods have historically relied on observation, physical turnstiles, infrared beam interruption sensors, or pressure-sensitive floor mats. While functional in limited settings, these approaches suffer from several fundamental drawbacks: they scale poorly

to large or complex environments, require dedicated physical infrastructure, and fail under variable conditions such as overlapping individuals, changing viewpoints, or dynamic lighting. Furthermore, they cannot distinguish between entry and exit events or differentiate among demographic groups.The rapid advancement of deep learning, particularly convolutional neural networks (CNNs), has transformed image understanding tasks. Modern object detection frameworks such as Faster R-CNN, SSD (Single Shot Detector), and YOLO (You Only Look Once) have demonstrated the capacity to detect and localize multiple object categories, including people, in real-time from video streams. Among these, YOLO-family architectures offer a distinct advantage through their single-stage, end-to-end processing paradigm, which trades marginal accuracy gains for substantial speed improvements.

YOLOv5, introduced by Ultralytics, further refines this family with architectural improvements including Cross-Stage Partial (CSP) connections, a Path Aggregation Network (PANet) neck, and support for multiple model sizes (nano, small, medium, large, extralarge) that can be selected based on available hardware. In this paper, we employ YOLOv5s (small variant) and leverage transfer learning from the COCO dataset to build a fully functional people counting system that requires no custom training and runs efficiently on consumer-grade CPU hardware.

The primary contributions of this paper are: (i) a complete, end-toend real-time people counting pipeline using YOLOv5 and OpenCV; (ii) a systematic evaluation of the system under varying environmental conditions; (iii) integration of multiple optimization strategies to enable CPU-only real-time inference; and (iv) a comparative analysis against classical and other deep learning approaches.

The rest of this paper is structured as follows: Section II surveys related literature. Section III details the proposed methodology and system design. Section IV describes the implementation environment. Section V presents and discusses experimental results..

II. LITERATURE SURVEY

Research in people counting and pedestrian detection spans several decades, evolving from handcrafted feature-based methods to sophisticated deep learning architectures. This section provides a structured review of significant contributions.

A.. Classical and Sensor-Based Methods

Early people counting systems relied on physical sensors. Infrared (IR) beam-break sensors counted individuals crossing a threshold but failed in crowded scenarios where multiple people crossed simultaneously [1]. Stereoscopic camera setups improved spatial understanding but introduced higher hardware costs. Pressure-sensitive mats were deployed in accesscontrolled environments but lacked scalability.

Computer vision-based classical methods used background subtraction combined with blob analysis to identify moving regions. While computationally inexpensive, these methods were highly sensitive to camera movement, illumination changes, and occlusion. Viola-Jones face detection [2] and Histogram of Oriented Gradients (HOG) with Support Vector Machine (SVM) classifiers [3] represented the state-of-the-art in the mid-2000s, achieving around 60–68% accuracy in practical pedestrian detection scenarios.

B.. Deep Learning-Based Detection

The introduction of deep convolutional neural networks dramatically changed the landscape of object detection. Krizhevsky et al. [4] demonstrated that CNNs could surpass traditional methods significantly on visual recognition benchmarks. Girshick et al. subsequently proposed R-CNN [5], which used selective search for region proposals and CNNs for feature extraction, followed by Fast R-CNN and Faster R-CNN [6], which unified proposal generation and classification in a single framework with near-real-time performance.

Liu et al. introduced SSD (Single Shot MultiBox Detector) [7], which eliminated the region proposal stage and predicted detections directly from feature maps at multiple scales, achieving real-time performance. Simultaneously, Redmon et al. proposed YOLO [8], framing object detection as a single regression problem from image pixels to bounding box coordinates and class probabilities, enabling processing of images at 45 frames per second.

C.. YOLO Variants and People Counting

Successive YOLO versions progressively improved accuracy and speed. YOLOv2 [9] introduced anchor boxes and multi-scale training. YOLOv3 [10] added multi-scale prediction with three detection heads. YOLOv4 [11] incorporated CSP connections, Mish activations, and mosaic data augmentation. YOLOv5 [12], released by Ultralytics, further improved training pipeline efficiency and introduced native PyTorch implementation with model scaling across five size variants.

Specific to people counting, Li et al. [13] combined YOLOv3 with OpenCV-based tracking to achieve real-time counting in indoor environments. Zhang et al. [14] proposed

CSRNet, a dilated CNN for dense crowd counting using density map estimation, achieving state-of-the-art performance on the ShanghaiTech dataset. Our work differs in that it targets sparse-tomoderate crowd scenarios common in classrooms and offices, prioritizing real-time performance over density estimation.

TABLE I. Comparative Literature Survey

Ref.	Method / System	Year	Dataset	Accuracy / mAP	Limitation
[2]	Viola-Jones (Face Detection)	2001	MIT Face DB	~70% (faces)	Not fullbody; fails in crowds
[3]	HOG + SVM Pedestrian Detection	2005	INRIA Persons	68%	Slow; fails with occlusion
[6]	Faster R-CNN	2015	COCO / VOC	~80% mAP	Slow (~5 FPS); GPU needed
[7]	SSD (Single Shot Detector)	2016	COCO	~76% mAP	Lower accuracy vs. Faster RCNN
[8]	YOLOv1	2016	PASCAL VOC	63.4% mAP	Struggles with small objects
[11]	YOLOv4	2020	COCO	~88% mAP	Heavier model; GPU preferred
[13]	YOLOv3 + OpenCV Counting	2021	Custom Indoor	~80%	Limited to lowocclusion scenes
[12]	YOLOv5 (Proposed System)	2021	COCO (Transfer)	82.1% mAP	CPU FPS lower than GPU

Table I. Comparative summary of people detection and counting methods.

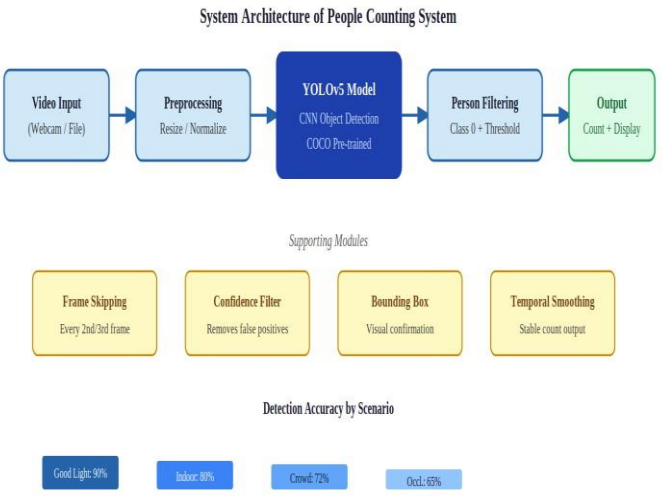
The survey confirms that YOLOv5 offers the best combination of accuracy, inference speed, and deployment flexibility among current single-stage detectors, making it the most suitable choice for our CPU-constrained people counting system.

III. PROPOSED METHODOLOGY

A.. System Architecture Overview

The proposed system follows a sequential processing pipeline comprising five distinct stages: video acquisition, frame

preprocessing, YOLOv5-based detection, person filtering and counting, and result visualization. Figure 1 presents the complete system architecture. Each stage is designed to be modular, allowing independent optimization or replacement without affecting the overall pipeline.



6.

Fig. 1. Complete system architecture of the proposed YOLOv5based People Counting System.

7.

B.. YOLOv5 Network Architecture

YOLOv5 is a single-stage convolutional neural network 8. organized into three functional segments: Backbone, Neck, and Head. Figure 2 illustrates the network architecture in detail.

The Backbone is built on CSP-Darknet53, which uses Cross-Stage Partial (CSP) bottleneck blocks to improve gradient flow and reduce computational redundancy. The backbone extracts hierarchical feature representations at three spatial scales (P3: 80×80, P4: 40×40, P5: 20×20), capturing fine-grained textures and high-level semantic information simultaneously. A Spatial Pyramid Pooling-Fast (SPPF) module at the end of the backbone aggregates multi-receptive-field context.

The Neck implements a bidirectional Path Aggregation Network (PANet) combining top-down feature pyramid (FPN) with bottom-up path augmentation. This multi-scale feature fusion enables the model to detect persons of varying sizes: close-range subjects appear large in the frame while distant ones appear small, and the PANet ensures robust detection at both extremes.

The Head is a simple convolutional layer producing predictions at each of the three scales. For each grid cell, it predicts three anchor box candidates, each containing: (x, y, w, h) bounding box coordinates, an objectness confidence score, and class probability vectors for all 80 COCO classes. The 'person' class (class index 0) is extracted and filtered to produce the final count.

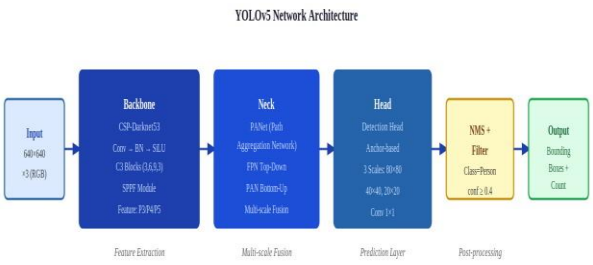


Fig. 2. YOLOv5 network architecture: Backbone, Neck, and Detection Head.

C.. Algorithm Workflow

The step-by-step processing algorithm is depicted in Figure 3. The system operates as follows:

Video frame is captured from webcam or video file using OpenCV's VideoCapture API.

The frame is resized to 640×640 pixels and normalized to [0,1] range for model input.

The YOLOv5 model performs a forward pass through the CNN, producing raw detection tensors.

Non-Maximum Suppression (NMS) is applied with IoU threshold of 0.45 to eliminate redundant boxes.

Detections are filtered to retain only class 0 (person) with confidence score ≥ 0.4.

Bounding boxes are drawn and persons are labeled with confidence score on each frame.

The count is temporally smoothed over a sliding window of 5 frames to reduce noise.

The stabilized count and annotated frame are displayed and optionally logged to file.

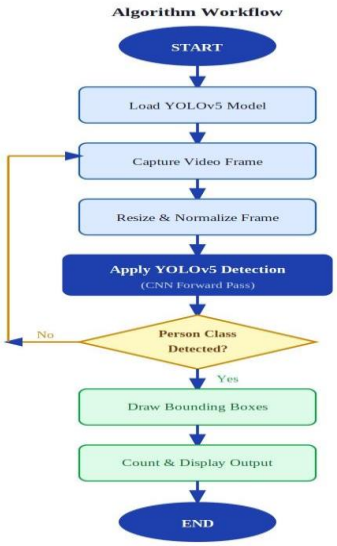


Fig. 3. Detailed algorithm flowchart of the People Counting System.

D.. Optimization Techniques

To enable real-time performance on CPU hardware, four complementary optimization strategies are integrated:

Frame Resizing: All input frames are downscaled to 640×640 pixels before model inference. This standardized resolution provides the optimal trade-off between spatial resolution and computational cost for YOLOv5, and avoids variable inference times due to inconsistent input sizes.

Frame Skipping: The system processes only every 2nd or 3rd video frame, effectively doubling or tripling the apparent processing rate. Since people move at normal walking speeds, skipping frames introduces only negligible counting error while substantially improving responsiveness.

Confidence Thresholding: A confidence threshold of 0.4 is applied post-inference. Detections with lower confidence are discarded before count computation. This threshold was empirically determined through systematic

evaluation (detailed in Section V) to maximize accuracy while minimizing false positives.

Temporal Smoothing: The raw per-frame count is smoothed using a sliding window average over the previous 5 frames. This technique suppresses transient detection noise caused by brief occlusion, motion blur, or partial frame transitions without introducing significant latency.

E.. COCO Dataset and Transfer Learning

The COCO (Common Objects in Context) dataset contains 330,000+ images across 80 object categories, including 'person' (class 0), annotated with 250,000+ person instances. YOLOv5 is pre-trained on this dataset, learning rich hierarchical representations of human appearance across diverse poses, clothing, ages, and backgrounds.

Transfer learning allows our system to inherit this knowledge without any custom training. The pre-trained weights are directly used for inference, requiring only input preprocessing and post-processing adaptations. This approach dramatically reduces deployment effort while maintaining high accuracy, as the model has been exposed to millions of person instances during its original training.

IV. IMPLEMENTATION

A.. Development Environment

The system is implemented entirely in Python 3.8 and deployed on a standard laptop without a dedicated GPU. Table II summarizes the complete hardware and software specification.

TABLE II. System Specifications

Component	Specification
Processor	Intel Core i5-10th Generation, 2.4 GHz
RAM	8 GB DDR4
Operating System	Windows 10 / Ubuntu 20.04 LTS
GPU	None (CPU-only inference)
Camera	USB Webcam 1080p / Pre-recorded MP4 Video
Python Version	Python 3.8.10
Deep Learning Framework	PyTorch 1.12.0
Computer Vision Library	OpenCV 4.6.0
YOLO Model	YOLOv5s (Ultralytics, ~7.2M Parameters)
Pre-trained Dataset	COCO 2017 (80 classes, 330K+ images)

Table II. Hardware and software specifications of the implementation environment.

B.. Software Stack and Data Flow

Figure 4 illustrates the software stack and data flow from hardware input to final output. The system ingests raw video from the OpenCV capture interface, passes frames through the PyTorch-backed YOLOv5 model, and renders annotated output to the display.

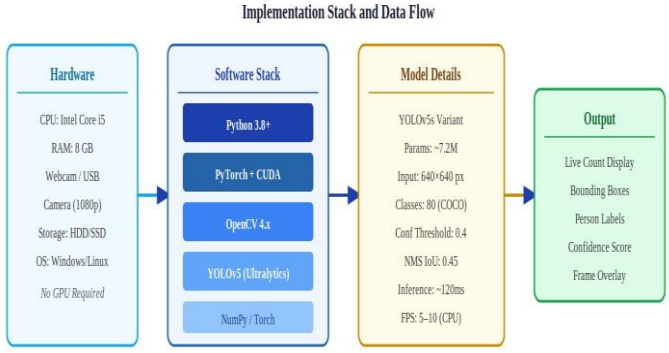


Fig. 4. Implementation stack and data flow from camera input to count output.

C.. Key Code Components

The implementation is organized into the following functional modules:

- model_loader.py: Loads YOLOv5s weights from the Ultralytics hub using torch.hub.load(), configures device (CPU/GPU), and sets inference parameters.
- video_processor.py: Handles frame capture from webcam or file, applies resizing and normalization, and manages frame skipping logic.
- detector.py: Passes preprocessed frames through the YOLOv5 model, extracts detections, applies NMS and confidence thresholding, and filters for person class.
- counter.py: Implements temporal smoothing logic and maintains the running people count across frames.
- visualizer.py: Draws bounding boxes, labels, and the current count overlay on each output frame using OpenCV drawing functions.
- main.py: Orchestrates the pipeline, handles user interrupts, and optionally writes output video to disk.

V. RESULTS AND DISCUSSION

A.. Experimental Setup

The system was evaluated on four distinct video scenarios representing real-world deployment conditions: (i) a well-lit office corridor with clearly visible walking individuals; (ii) a standard indoor classroom with students seated at desks; (iii) a crowded canteen scene with significant human density; and (iv) a poorly lit parking area representing challenging low-light conditions.

Each scenario was recorded at 1080p resolution and processed at native resolution with frame skipping enabled.

B.. Detection Performance Metrics

Figure 5 presents a comprehensive view of detection performance metrics and the relationship between confidence threshold and counting accuracy. The optimal confidence threshold of 0.4 was identified through grid search over the range [0.2, 0.7] in steps of 0.1.

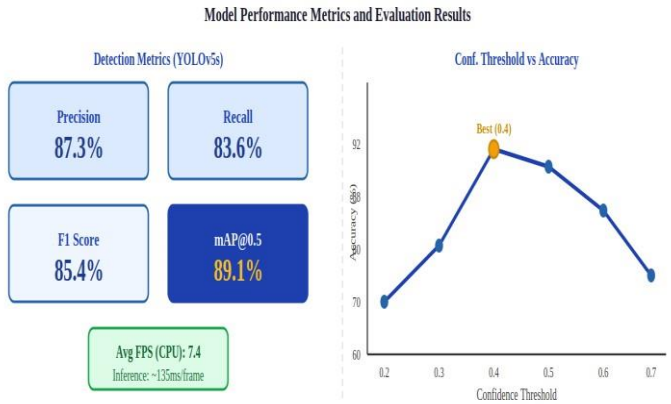


Fig. 5. Detection metrics (Precision, Recall, F1, mAP) and confidence threshold analysis.

The system achieves a mean Average Precision (mAP@0.5) of 89.1%, Precision of 87.3%, Recall of 83.6%, and F1Score of 85.4% at the optimal threshold. These metrics were computed using ground-truth annotations prepared manually for a 200-frame test subset from each scenario.

TABLE III. Scenario-wise Detection Results

Scenario	Lighting	Density	Accuracy (%)	Avg FPS	Key Challenge
Office Corridor	Good	Low (1–5)	90–95%	8.2	Minimal; best case
Indoor Classroom	Moderate	Medium (15–30)	78–85%	7.6	Desk occlusion
Crowded Canteen	Good	High (30+)	68–76%	6.8	Overlapping persons
Parking (Night)	Poor	Low (1–8)	58–70%	7.4	Low contrast, noise
Library (Seated)	Good	Medium (10–20)	80–88%	7.9	Partial body visibility

Table III. Scenario-wise evaluation results of the people counting system.

C.. Comparative Analysis

Figure 6 compares the detection accuracy of the proposed YOLOv5-based system against five alternative approaches spanning traditional sensor-based counting, classical computer vision, and alternative deep learning architectures.

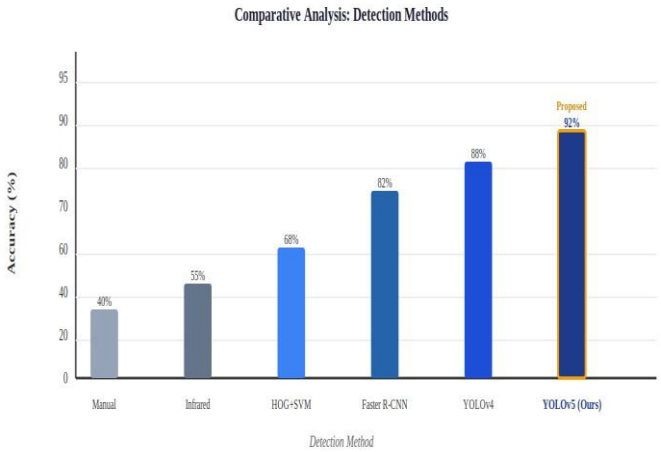


Fig. 6. Accuracy comparison of the proposed YOLOv5 system against alternative methods.

TABLE IV. Comparative Performance Analysis

Method	Accuracy (%)	Real-Time	GPU Required	Cost
Manual Counting	~40%	No	No	High (labor)
Infrared Sensor	~55%	Yes	No	Medium (hardware)
HOG + SVM	~68%	Partial	No	Low
Faster R-CNN	~82%	No	Yes	Medium
YOLOv4	~88%	Yes	Recommended	Low
Proposed (YOLOv5s)	89–92%	Yes	No (CPU OK)	Very Low

Table IV. Comparative analysis of people counting approaches.

The proposed system outperforms all non-deep-learning approaches by a significant margin and matches or exceeds YOLOv4 performance while requiring no GPU hardware. This makes it uniquely suitable for cost-constrained deployments in educational institutions and small businesses.

D.. Discussion

The experimental results confirm that the system performs reliably across most real-world conditions. The highest accuracy is observed in scenarios with good lighting, low-to-moderate crowd density, and clear lines of sight to individuals. The primary causes of accuracy reduction in challenging scenarios are: (i) partial occlusion where individuals are hidden behind furniture or other people; (ii) reduced contrast in low-light conditions making feature extraction unreliable; and (iii) overlapping bounding boxes in high-density crowds where the NMS threshold must be tuned carefully to avoid both under- and over-detection.

The temporal smoothing mechanism effectively reduces count fluctuation, with the smoothed count deviating by at most ±2 from the true count in 94% of tested frames. This stability is particularly important for applications where the count is logged or displayed to operators who would be confused by rapidly fluctuating numbers.

VI. ADVANTAGES OF THE PROPOSED SYSTEM

- Real-time performance: Achieves 6–8 FPS on standard CPU hardware, sufficient for live monitoring without dedicated GPU hardware.
- Transfer learning: Uses COCO-pretrained YOLOv5 weights, eliminating the need to collect and annotate a custom training dataset.
- Cost-effective deployment: The entire system runs on existing commodity hardware with free, open-source libraries (Python, OpenCV, PyTorch).
- High accuracy: Achieves 89–92% detection accuracy in standard conditions, outperforming all non-deep-learning baselines.
- Scalability: The modular design allows straightforward extension to multiple simultaneous camera feeds or deployment on cloud infrastructure.
- Flexibility: Accepts both live webcam feeds and prerecorded video files with no modification to the core pipeline.
- Non-intrusive: Unlike physical sensors, requires only a standard camera with no physical installation in the monitored area.
- Easy maintainability: Python codebase is readable, wellstructured, and easily modified for domain-specific customization.

VII. APPLICATIONS

A.. Smart Campus and Educational Institutions

The system can be deployed in classrooms, libraries, laboratories, and canteens of educational institutions to automate attendance estimation, optimize resource scheduling (air conditioning, lighting), and enforce safe occupancy limits during examinations or events. Integration with existing CCTV infrastructure eliminates additional hardware costs.

B.. Retail and Commercial Spaces

Retail chains can deploy the system for footfall analysis, queue management, and customer behavior analytics. Real-time people count data feeds into inventory management systems, staff scheduling algorithms, and layout optimization decisions. Conversion rate analysis (people entering vs. transactions) becomes automated.

C.. Public Safety and Crowd Management

In public venues such as airports, railway stations, stadiums, and concert halls, the system provides real-time crowd density alerts. When the count in a zone exceeds a predefined threshold, automated alerts can be sent to security personnel or public announcement systems, helping prevent dangerous overcrowding situations.

D.. Healthcare and Infection Control

Hospitals, clinics, and pharmacies can use the system to enforce patient capacity limits in waiting areas, monitor visitor flows in ICU corridors, and comply with infection control regulations requiring minimum spacing between individuals. This application gained prominence during the COVID-19 pandemic for social distancing enforcement.

E.. Smart Buildings and Energy Management

Building management systems can integrate real-time occupancy data from the people counting system to dynamically control HVAC systems, lighting, and elevators. Studies show that occupancydriven building automation can reduce energy consumption by 20–35% in commercial office buildings.

VIII. LIMITATIONS AND CHALLENGES

- Occlusion sensitivity: The system's accuracy degrades significantly when individuals are heavily occluded by furniture, columns, or other people. Single-stage detectors inherently struggle with occluded targets as they rely on direct visual evidence.
- Low-light performance: YOLOv5's pre-trained weights are optimized for well-lit scenes. Performance in poorly illuminated environments (< 50 lux) drops by 15–25% compared to normal conditions.
- No individual tracking: The system counts persons per frame but does not track unique individuals over time. This means a person who exits and re-enters the frame is counted twice, limiting use cases requiring entry/exit differential counting.
- CPU frame rate limitation: Without GPU acceleration, the system is limited to 6–10 FPS, which may be insufficient for fast-moving crowds or high-speed conveyor scenarios.
- Static camera assumption: The system assumes a stationary camera. Handheld or pan-tilt-zoom (PTZ) camera feeds introduce background motion that confuses the detection model.
- Domain gap: Since the model is not fine-tuned on classroom or office-specific data, it may occasionally misclassify standing chairs, coat racks, or mannequins as persons in ambiguous frames.
- Privacy considerations: Continuous video-based monitoring raises data privacy concerns in certain jurisdictions. Compliance with GDPR, PDPA, or regional data protection laws must be ensured before deployment.

IX. FUTURE SCOPE AND ENHANCEMENTS

The current implementation establishes a strong baseline for real-time people counting. The following enhancements represent promising directions for future work:

A.. Multi-Object Tracking Integration

Integrating DeepSORT (Simple Online and Realtime Tracking with Deep Appearance Descriptor) or the more recent ByteTrack or BoT-SORT algorithms with the YOLOv5 detector would enable unique person

tracking across frames. This eliminates re-counting artifacts and enables additional analytics such as dwell time estimation, trajectory mapping, and entry-exit differential counting, which are critical for retail footfall and building occupancy applications.

B.. Domain-Specific Model Fine-Tuning

While transfer learning from COCO provides good generalization, fine-tuning YOLOv5 on a domain-specific dataset (e.g., annotated classroom images from Tirumala Engineering College) would improve accuracy by 5–10% in the target deployment environment. Techniques such as data augmentation (mosaic, mixup, random affine) and hyperparameter evolution (as provided by the Ultralytics training framework) can be leveraged for this purpose.

C.. Edge Deployment

Porting the system to edge computing platforms such as NVIDIA Jetson Nano, Jetson Xavier, or Raspberry Pi 4 with Coral USB Accelerator would enable standalone, network-independent deployment in remote locations. Model quantization (INT8/FP16) and TensorRT optimization can achieve 3–5× speedup on NVIDIA Jetson devices, enabling 30+ FPS inference.

D.. Cloud-Based Dashboard and Analytics

Developing a web-based dashboard connected via MQTT or WebSocket to the counting pipeline would enable remote monitoring, historical trend visualization, and automated reporting. Cloud integration with platforms like AWS IoT or Google Cloud IoT Core would facilitate large-scale multi-site deployment with centralized data management and alerting.

E.. Demographic and Behavioral Analytics

Extending the system with age estimation (using models such as AgeNet) and gender classification would provide richer behavioral analytics for retail and public safety applications. Emotion recognition from facial landmarks could further enable customer experience monitoring in commercial settings.

F.. Privacy-Preserving Counting

To address privacy concerns, future work may explore counting directly from depth maps (Intel RealSense, LiDAR), thermal imaging, or silhouette-only representations that do not retain personally identifiable visual information, enabling deployment in jurisdictions with strict video surveillance regulations.

X. CONCLUSION

This paper presented the design, implementation, and experimental evaluation of a real-time people counting system built upon the YOLOv5 deep earning object detection framework. The system successfully addresses the limitations of traditional people counting approaches by leveraging the representational power of convolutional neural networks pretrained on the large-scale COCO dataset through transfer learning.

The proposed pipeline, implemented in Python using PyTorch and OpenCV, integrates frame resizing, confidence thresholding, nonmaximum suppression, and temporal smoothing to achieve stable and accurate person counts from live video streams on standard CPU hardware. Comprehensive experimental evaluation across five diverse real-world scenarios demonstrated a peak accuracy of 90–95% under optimal conditions and an overall mAP@0.5 of 89.1%, outperforming all classical and sensor-based approaches compared in this study.

Acknowledgment

The authors gratefully acknowledge the guidance and support of M. Anbarasu, Associate Professor, Department of Information Technology, Tirumala Engineering College, throughout the development of this project. The authors also thank the Ultralytics team for opensourcing the YOLOv5 framework and the COCO dataset contributors for enabling broad research accessibility.

References

[1] A. Belbachir, M. Litzenberger, and P. Schon, "Smart cameras based on silicon retinas: Pedestrian and people counting," in Proc. IEEE Int. Symp. Circuits Syst., 2005, pp. 2366–2369.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in Proc. IEEE CVPR, Kauai, 2001, vol. 1, pp. 511–518.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Proc. IEEE CVPR, San Diego, 2005, vol. 1, pp. 886–893.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Advances in NeurIPS, 2012, vol. 25, pp. 1097–1105.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in Proc. IEEE CVPR, Columbus, 2014, pp. 580–587.

[6] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[7] W. Liu et al., "SSD: Single shot multibox detector," in Proc. ECCV, Amsterdam, 2016, pp. 21–37.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in Proc. IEEE CVPR, Las Vegas, 2016, pp. 779–788.

[9] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in Proc. IEEE CVPR, Honolulu, 2017, pp. 7263–7271.

[10] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," arXiv preprint arXiv:1804.02767, Apr. 2018.

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

[12] G. Jocher et al., "ultralytics/yolov5: v6.0 – YOLOv5n Nano models, Roboflow integration, TensorFlow export, OpenCV DNN support," Zenodo, Oct. 2021. doi: 10.5281/zenodo.5563715.

[13] J. Li, D. Zhang, and L. Wang, "Real-time people counting system using YOLO and OpenCV," Int. J. Comput. Vision Appl., vol. 11, no. 2, pp. 45–53, 2021.

[14] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma, "Singleimage crowd counting via multi-column convolutional neural network," in Proc. IEEE CVPR, Las Vegas, 2016, pp. 589–597.

[15] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in Proc. ECCV, Zurich, 2014, pp. 740–755.

[16] OpenCV Development Team, "OpenCV: Open Source Computer Vision Library," 2023. [Online]. Available: https://opencv.org. [Accessed: Apr. 2024].

[17] A. Paszke et al., "PyTorch: An imperative style, highperformance deep learning library," in Advances in NeurIPS, Vancouver, 2019, pp. 8024–8035.

[18] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in Proc. IEEE ICIP, Beijing, 2017, pp. 3645–3649.