IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511114

Smart Surveillance and Crowd Management System Using a Custom-Trained YOLO11n Person Detector

Tharunyaa S.S[1], N K Sriram[2], Kolamala Sai Divya[3], Susmitha R[4]

Department of Computer Science and Engineering (Emerging Technologies), Faculty of Engineering and Technology,

SRM Institute of Science and Technology, Vadapalani Campus, Chennai, India[1-4]

Abstract: Noticing that a crowd is building up is hard for a human operator on a long shift, and it is also hard for off-the-shelf detectors when the footage is aerial or camera-mounted high, because people are only a few pixels tall. We built an offline tool to help with this. It pairs JATAYU v2, a person detector that we trained ourselves on our own dataset (YOLO11n architecture), with a deliberately simple analysis layer: a user-defined grid of zones, LOW, MODERATE and HIGH labels based on count thresholds that the operator sets, a heatmap of where people were detected most often, a log of threshold crossings, and a Streamlit dashboard that runs on an ordinary laptop CPU. On its validation split the detector reached precision 0.724, recall 0.583, mAP50 0.638 and mAP50-95 0.312. On an 8-second aerial clip (240 frames, 960 × 540) the full pipeline analysed 120 frames in 11.7 s and found 154.8 people per frame on average, with a maximum of 176, whereas COCO-pretrained YOLOv8n and YOLOv8s found 0.41 and 0.11. The clip has no ground-truth counts, so this comparison is qualitative only. Additional tests show that the count depends heavily on the confidence threshold and the inference size, hardly at all on frame skipping, and that the zone grid and the zone threshold have to be chosen together. All numbers are detections per frame rather than unique people, and the heatmap and density labels are indicators, not calibrated measurements.

Keywords: crowd management, person detection, YOLO11, zone occupancy, detection heatmap, video surveillance, CPU inference

I. INTRODUCTION

When many people gather in a limited space, as at festivals, railway stations or sports events, local density can rise until movement is restricted and an emergency response becomes difficult. Accounts of crowd disasters tend to agree on one point: the build-up is gradual, but nobody was measuring it across the whole venue [13]. In practice the measuring is left to CCTV operators, who may have dozens of screens to watch for hours. Attention fades under such conditions, so counting people, comparing one area with another or noticing a slow increase is unreliable, and recordings are afterwards searched by eye.

A computer can take over the routine part of this watching. A person detector gives a count for every frame, a zone grid shows where the people are, accumulating the detections over time shows which parts of the scene are used most, and a simple threshold can raise an early flag. We see such outputs as a way of telling the operator where and when to look, not as a replacement for the operator. Detectors from the YOLO family [1], [2], [16] are fast enough for ordinary hardware and are easy to fine-tune, which matters for campuses and small venues that cannot pay for GPU servers or cloud analytics.

The difficulty is that detectors pretrained on COCO [3] have mostly seen ground-level photographs. In footage from a high mounting point or a drone, people are small, viewed from above and often overlapping, and benchmarks such as VisDrone [18] show how hard such small objects are. We therefore asked a practical question: can a person detector trained for this view, combined with an easily explained zone-and-threshold layer in an offline dashboard, give an operator useful counts, zone occupancy, a heatmap and congestion flags on a CPU, while being honest about what each output does and does not measure?

Our contributions are as follows.

- JATAYU v2, our own single-class person detector (YOLO11n architecture), trained by our team on a dataset we built for small people seen from above; it runs on a laptop CPU.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511114

- An analysis layer whose every number can be traced to a count, an operator threshold or a stated formula: zone occupancy, density labels, a foot-point heatmap and non-repeating threshold-crossing events.
- An offline dashboard that exports an annotated video, heatmap, count graph, per-frame CSV, JSON summary and logs, backed by 21 automated tests.
- An evaluation that combines validation metrics, an end-to-end run, a qualitative baseline comparison and sensitivity tests, with a plain statement of what this evidence cannot show.

The system works on recorded video (MP4, AVI, MOV, MKV). It does not read live streams, track identities, recognise faces or estimate people per square metre. Counts are detections per frame, and a congestion flag is not a prediction of a stampede.

II. RELATED WORK

Early pedestrian detectors paired hand-designed features such as Histograms of Oriented Gradients with a linear classifier [9]. A large benchmark study found that detection stayed disappointing for small and partly hidden pedestrians [15]. Two-stage CNN detectors such as Faster R-CNN [8] were more accurate but slower. Single-stage detectors, beginning with YOLO [1] and followed by later versions [2], [16], predict all boxes in one pass and therefore suit CPU use. Non-maximum suppression [11] removes duplicate boxes, but in a dense crowd it can also delete genuine neighbours, which is one reason counts in crowds are usually too low.

For very dense scenes the usual approach is to regress a density map and integrate it to obtain the count. Multi-column networks [10] cope with changes in head size, and CSRNet [5] uses dilated convolutions for congested scenes; a survey [6] groups crowd counting into detection-, regression- and density-based methods. These methods count well at extreme density, but they need dense point annotations and give no box for each person. Trackers such as SORT [12] and DeepSORT [17] give each person an identity and would turn per-frame counts into counts of unique people. The safety motivation comes from work on crowd panic [13] and from pedestrian level-of-service standards expressed in people per unit area [14].

TABLE I QUALITATIVE COMPARISON OF CROWD-MONITORING TECHNIQUES, BASED ON THE LITERATURE CITED ABOVE.

Technique	Strengths	Limitations
Manual observation or gate counting	No technology needed	Labour-intensive; no spatial view
HOG + SVM detectors [9]	Light and interpretable	Sensitive to pose, lighting and occlusion
Two-stage CNN detectors [8]	High accuracy	Slow on a CPU
Single-stage detectors (YOLO) [1], [2]	Real time; one box per person; can be fine-tuned	Under-counts at extreme density; needs training data
Density-map regression [5], [10]	Best at extreme density	No individual boxes; needs dense annotation

Four gaps motivated our work. Small, top-down and partly hidden people lower the recall of generic detectors. Counts per frame are not counts of people unless tracking is added. Heatmaps and density labels are often shown as if they were physical measurements, although they are not calibrated. And operators lack a simple offline tool that combines detection, zones, alerts, logs and exports without cloud services or special hardware.

III. PROPOSED FRAMEWORK

A. *Overview*

The framework is organised in six layers (Fig. 1). The dashboard collects the footage and the settings. The processing layer reads frames and runs the loop. The inference layer applies the JATAYU v2 detector. The analytics layer turns detections into zone counts, density labels, statistics and heatmap data, and the visualisation, logging and storage layers write the outputs for people and for programs. Each layer depends only on the one below it. The analytics layer does not depend on the detector at all, so we could test it without any model weights. Fig. 2 is the flowchart of one complete run.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511114

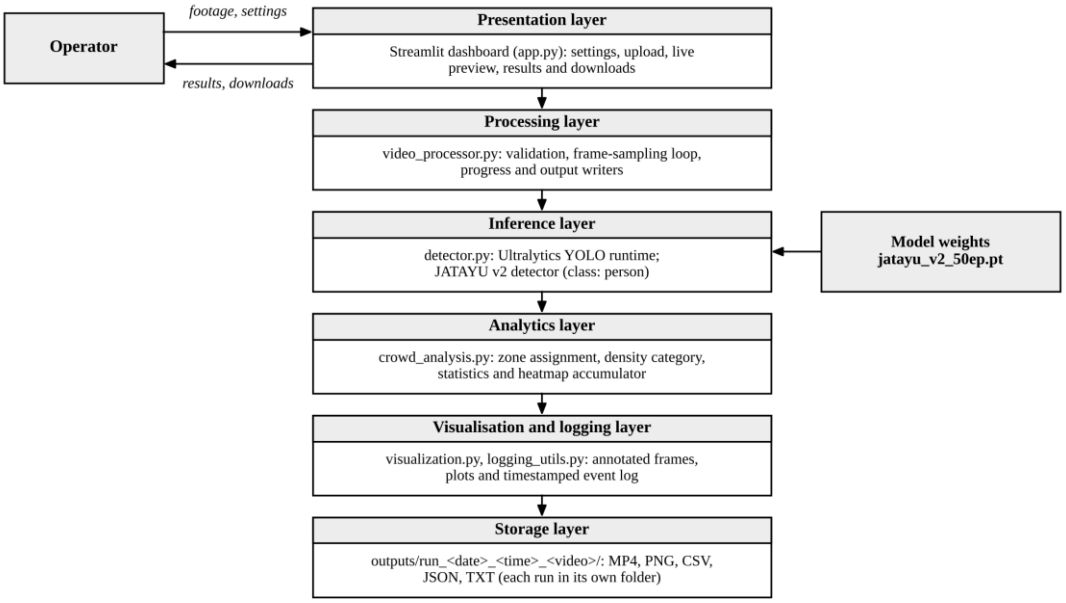


Fig. 1. Layered architecture of the system.

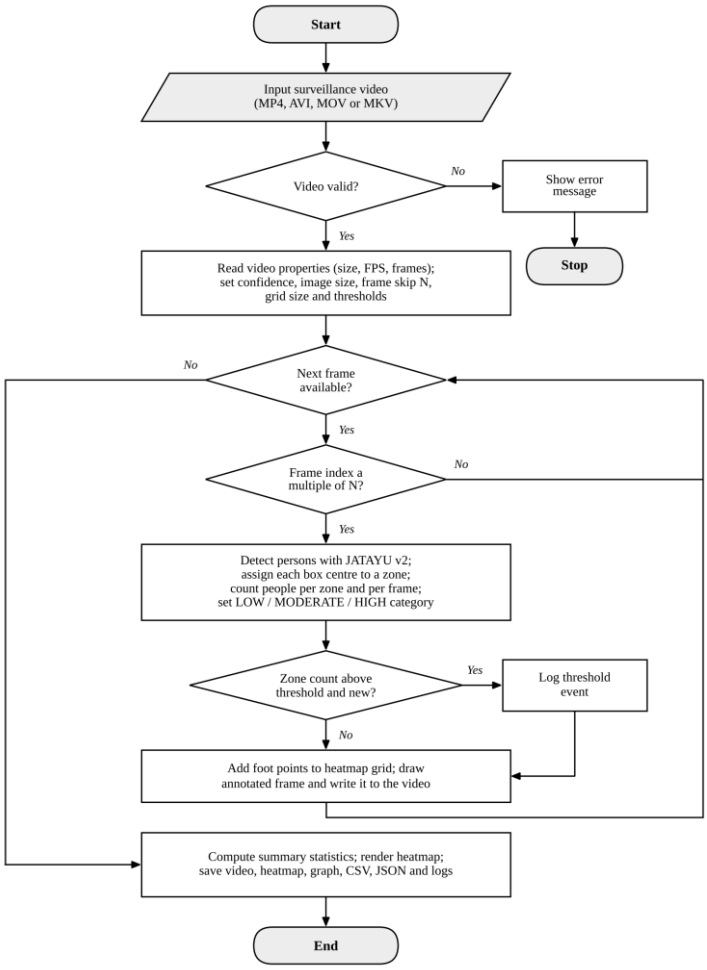


Fig. 2. Flowchart of one analysis run.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511114

B. *Detector Training*

JATAYU v2 uses the YOLO11n architecture [2] with one class, person, and was trained on JATAYU_V2_DATASET, a single-class person dataset assembled by our team. Training took place in two stages, which together account for the 50 epochs in the file name jatayu_v2_50ep.pt: an earlier 30-epoch stage (checkpoint jatayu_v2_30ep), followed by a final 20-epoch stage on a CPU that took about 11.7 hours (Table II). The training data, the training runs and the resulting weights are our own work; only the network architecture and the training framework come from Ultralytics [2]. The checkpoint stores the validation metrics of the final stage but not the number of images, the split or the annotation details of the dataset, so we do not report them; we return to this in Section VII.

TABLE II TRAINING CONFIGURATION OF THE FINAL STAGE OF JATAYU V2.

Setting	Value
Framework	Ultralytics 8.4.75
Architecture	YOLO11n (nano), single class: person
Starting point	jatayu_v2_30ep checkpoint
Epochs	20 (50 in total with the earlier stage)
Image size / batch size	640 px / 8
Hardware	CPU only, about 11.7 h for the final stage
Optimiser	Automatic selection (Ultralytics default)
Augmentation	Mosaic, random HSV changes, horizontal flip 0.5, translation 0.1, scale 0.5, random erasing 0.4

C. *Frame Sampling and Detection*

The input video is opened with OpenCV [4] and checked before any processing: the extension must be supported, the container must open and the first frame must decode. If the frame rate reported by the file is invalid, 25 FPS is used and a warning is logged. Every N-th frame (N = 2 by default) is passed to the detector with the chosen confidence threshold and inference size, restricted to the person class. For each box we use the centre to decide the zone and the bottom-centre, the foot point, for the heatmap.

D. *Zones, Density Labels and Events*

The frame is divided into R × C zones (3 × 3 by default, up to 6 in each direction). For a box centre (c_x, c_y) in a frame of width W and height H, the zone identifier is id = row·C + col + 1 with col = ⌊(c_x/W)·C⌋ and row = ⌊(c_y/H)·R⌋; points outside the frame are moved to the nearest edge zone. A zone is flagged when its count is strictly greater than the zone threshold. The whole frame is labelled LOW when the count is at most the LOW limit, MODERATE up to the MODERATE limit and HIGH above it. An event is logged only when a zone newly crosses the threshold, so a zone that stays congested does not fill the log with repeats; each event stores the frame, time, zone, count and a short message. Because we have no camera calibration, the thresholds are chosen by the operator, and the labels are therefore indicators rather than physical density. Algorithm 1 summarises the per-frame logic.

Algorithm 1. Per-frame analysis

```
open and validate video V; set heat grid to zero; mark every zone as "not above threshold"
for each frame f_k of V with k mod N = 0 do
    boxes ← detect(f_k, confidence c, image size s), person class only
    for each box b do: z ← zone of the centre of b; n_z ← n_z + 1; add the foot point of b to heat
    n ← number of boxes; label ← LOW if n ≤ L, MODERATE if n ≤ M, otherwise HIGH
    for each zone z do
        if n_z > T and z was not above T in the previous analysed frame then log event (k, time, z, n_z)
        remember whether n_z > T
    write the annotated frame to the output video; record (k, time, n, label, n_z)
smooth and normalise heat; write CSV, JSON, heatmap, count graph and logs
```

E. *Detection-Frequency Heatmap*

Foot points from all analysed frames are added to a coarse grid (at most 320 pixels on the long side). The grid is smoothed with a Gaussian filter whose standard deviation is 3% of the long side (at least 1.5 cells), scaled to the range 0 to 1, enlarged to the frame size and blended over the first frame with a colour map. The result tells the operator where people were detected most often. It is not a calibrated density map.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511114

F. *Implementation and Dashboard*

The program has about 1,500 lines of Python in eleven files. Ultralytics provides inference, OpenCV handles video and drawing, NumPy and pandas hold the data, Matplotlib draws the count graph and Streamlit provides the dashboard. All processing parameters can be changed in the dashboard, which shows a live annotated preview with progress and the current, maximum and average count, and offers the annotated video, heatmap, count graph, per-frame CSV, JSON summary and timestamped logs for download. Each run writes to its own folder, and the JSON summary records the video properties, model, device, settings, statistics and events, so a run can be audited later. Inference uses a CUDA GPU if one is available and the CPU otherwise. Twenty-one pytest tests cover zone assignment, counting, statistics, the density limits, the heatmap accumulator and a complete run on a synthetic video with a stub detector; all 21 passed in 0.80 s in a fresh virtual environment.

IV. EXPERIMENTAL SETUP

1) Evaluation clip: All system-level experiments use one aerial clip of a busy road (960 × 540, 29.97 FPS, 240 frames, 8.0 s). It is third-party stock footage, not our own, and we use it only to evaluate the system; we do not reproduce its frames in this paper. The clip comes with no ground-truth person counts.

2) Protocol: At model level, precision, recall, mAP50 and mAP50-95 on the validation split are read from the values the training run stored in the checkpoint. At system level we run the complete pipeline on the clip and compare it with the COCO-pretrained YOLOv8n and YOLOv8s models, counting the person class only. We then vary the confidence threshold, inference size, frame skip, grid size and zone threshold one at a time. Since the clip has no ground truth, we do not claim any counting accuracy: the experiments show how the outputs respond to the settings, not which setting is correct.

TABLE III SETTINGS OF THE MAIN RUN AND OF THE SENSITIVITY EXPERIMENTS.

Item	Value
Main run	JATAYU v2, confidence 0.35, imgsz 640, every 2nd frame, 3 × 3 grid, zone threshold 15, LOW ≤ 20, MODERATE ≤ 60. The code defaults are zone threshold 5, LOW ≤ 5 and MODERATE ≤ 15; we raised them because the clip contains well over a hundred people per frame.
Main-run platform	Ordinary CPU-only laptop
Sensitivity runs	Same clip; Ultralytics 8.4.203, PyTorch CPU build, 16-thread CPU; inference-only timing
Baselines	YOLOv8n and YOLOv8s (COCO-pretrained), person class only

Training used Ultralytics 8.4.75, whereas the sensitivity runs used 8.4.203 on a different machine. Absolute timings of the two sets are therefore not comparable, and we only interpret relative effects.

V. RESULTS

A. *Detector Validation Metrics*

TABLE IV JATAYU V2 VALIDATION METRICS AT THE LAST EPOCH OF THE FINAL TRAINING STAGE.

Metric	Value
Precision	0.724
Recall	0.583
mAP50	0.638
mAP50-95	0.312
Validation box loss	1.712
Validation class loss	1.034

During the 20 epochs the training box loss fell from 1.97 to 1.69 and the validation box loss from 1.86 to 1.71, while mAP50 rose from 0.581 to 0.638 and recall from 0.544 to 0.583 (Fig. 3). The training and validation losses went down together and did not drift apart, so we see no sign of over-fitting within this stage, and the curves were still improving at epoch 20. In plain terms, about seven in ten detections are correct and about six in ten annotated people are found. In dense parts of a scene the count is therefore more likely to be too low than too high.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511114

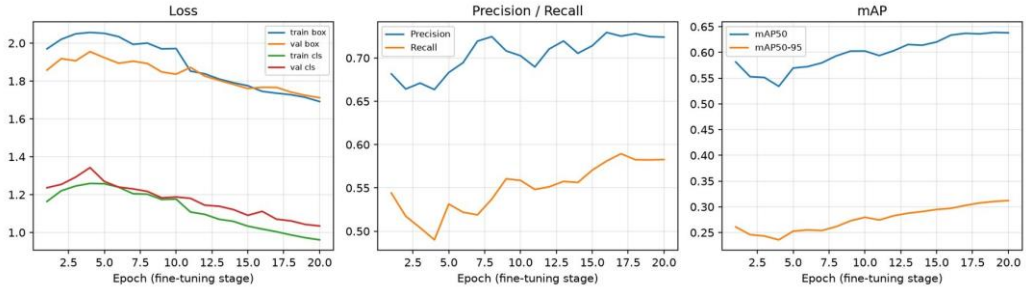


Fig. 3. Training and validation curves of JATAYU v2 (final 20-epoch stage).

B. *End-to-End Run and Count over Time*

The pipeline analysed 120 frames, every second frame, in 11.7 s on the CPU. That is 10.3 analysed frames per second including drawing, video writing and re-encoding. Each analysed frame stands for two source frames, so the pipeline handled about 20.5 source frames per second, or 0.68 times real time for this 29.97 FPS clip at frame skip 2. JATAYU v2 found 154.8 people per analysed frame on average (maximum 176 at source frame 62; 145 in the last analysed frame), and every one of the 120 frames fell into the HIGH category (Table V). The count stayed between 137 and 176 with a standard deviation of 8.7. Between consecutive analysed frames it changed by 4.9 detections on average (Fig. 4), so a moving average gives a steadier indicator than a single frame.

TABLE V RESULTS ON THE EVALUATION CLIP (CONFIDENCE 0.35, IMGSZ 640, EVERY 2ND FRAME, 3 × 3 GRID, ZONE THRESHOLD 15).

Result	JATAYU v2 (ours)	YOLOv8n (COCO)	YOLOv8s (COCO)
Average people per frame	154.8	0.41	0.11
Maximum in a frame	176	2	2
Frames in category HIGH	120 of 120	0 of 120	–
Zones above threshold (any frame)	9 of 9	0 of 9	–
Threshold events logged	67	0	–
Pipeline time, whole run	11.7 s	11.6 s	–
Pipeline speed	10.3 FPS	10.4 FPS	–
Inference time (ms per frame)*	26.4	26.5	61.7

– not measured in the main run. *Inference-only time on the machine of the sensitivity runs.

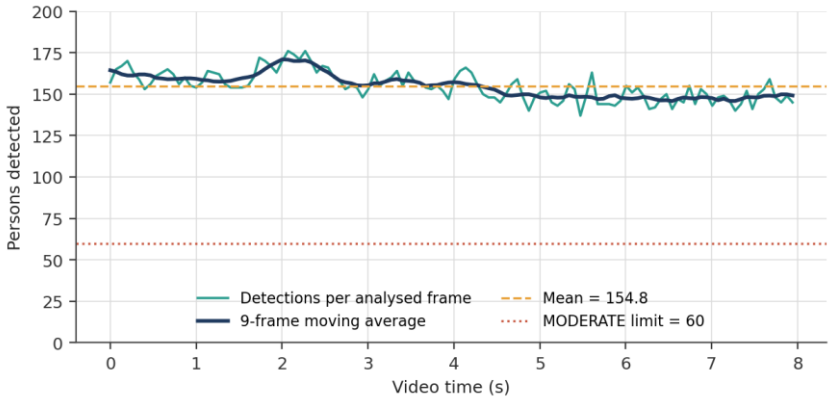


Fig. 4. Person count per analysed frame (JATAYU v2) with a 9-frame moving average and the MODERATE limit.

C. *Zone Occupancy and Threshold Events*

With a 3 × 3 grid and a zone threshold of 15, every zone exceeded the threshold in at least one frame (Table VI, Fig. 5). Zone 2 at the top centre was the busiest, with a mean of 24.0 people, a peak of 33 and a count above the threshold in 119 of 120 frames. Zone 3 at the top right was the quietest and exceeded the threshold in only 4 frames. The run logged 67 threshold-crossing events. Zones 5 (14 events), 9 (12), 4 and 6 (10 each) crossed most often, whereas zone 2 produced only 2 events because it stayed above the threshold almost continuously. Event counts therefore describe how often a

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511114

zone oscillates around the threshold, not how long it stays above it. The threshold itself has to suit the scene: raising it from 5 to 30 reduced the flagged zone-frames from 1077 to 6 out of 1080, and a useful value for this clip lies roughly between 15 and 25.

TABLE VI ZONE OCCUPANCY (3 × 3 GRID, ZONE THRESHOLD 15).

Zone	Avg people	Peak	Frames over threshold	% frames over
1	12.8	19	17	14.2
2	24.0	33	119	99.2
3	11.3	18	4	3.3
4	14.9	23	53	44.2
5	14.4	20	44	36.7
6	19.1	27	106	88.3
7	21.2	28	116	96.7
8	19.9	28	99	82.5
9	17.2	26	72	60.0

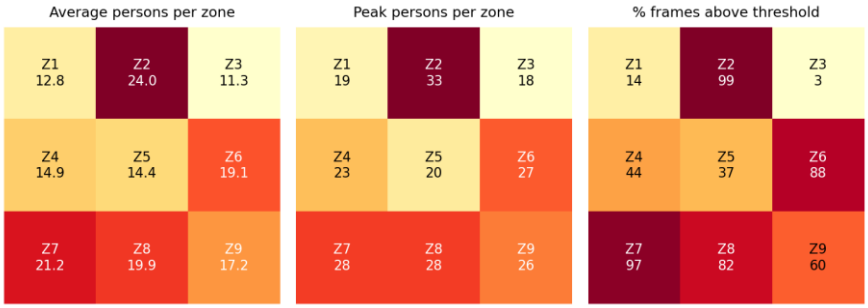


Fig. 5. Zone statistics on the 3 × 3 grid: average people, peak people and share of frames above the threshold.

D. *Detection-Frequency Heatmap*

The heatmap (Fig. 6) shows three hot spots: a cluster at the lower left (zone 7), an arc across the upper centre and right (zone 2 and part of zone 3) and a smaller cluster at the lower right (zone 9). This agrees with the zone statistics, where zones 2, 7 and 8 have the highest averages. The same people stay in the scene for the whole 8 s, so a high value means that detections piled up at that place. It does not mean that the physical density has a particular value.



Fig. 6. Detection-frequency map after Gaussian smoothing and normalisation (red = detected most often).

E. *Comparison with Pretrained Baselines*

On this clip the COCO-pretrained models found almost nobody, whereas JATAYU v2 found people across the whole scene (Table V, Fig. 7). The larger YOLOv8s was also slower (61.7 ms against about 26 ms per frame) and found even fewer people than YOLOv8n. This points to a mismatch between the training data and the footage rather than a lack of model capacity. People in this clip are on average only 19.1 px tall in a 540 px frame (3.5% of the frame height) and 10.6 px wide; over all 240 frames at confidence 0.35 the detector produced 37,143 detections with a mean confidence of 0.49.

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511114

Because the clip has no ground truth, the comparison is qualitative. It shows that generic models fail on this kind of view, but it does not tell us how accurate the counts of JATAYU v2 are.

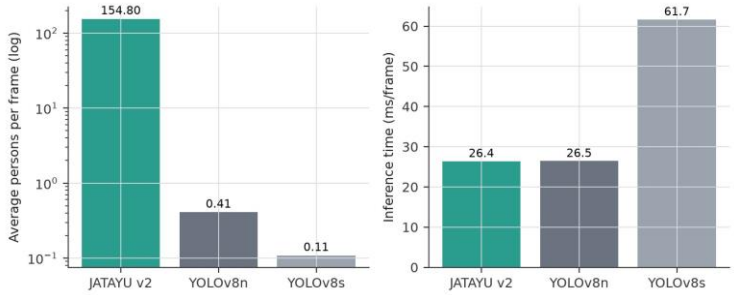


Fig. 7. Average persons per frame (log scale, left) and inference time (right) of the three detectors.

F. *Sensitivity to Processing Settings*

1) Confidence threshold: The count reacts strongly to this setting. It falls from 300 to 13 people per frame as the threshold goes from 0.15 to 0.65 (Table VII, Fig. 8). At 0.15 every frame reaches exactly 300 detections, which is the default maximum number of detections per image in the Ultralytics predictor, so the true value at that setting is cut off. Inference time does not depend on the threshold. With a precision of about 0.72, low thresholds let in false detections and high thresholds miss real people, so the absolute count should be read as depending on the threshold.

TABLE VII EFFECT OF THE CONFIDENCE THRESHOLD (JATAYU V2, IMGSZ 640, EVERY 2ND FRAME).

Confidence	Avg persons per frame	Max in a frame	Inference (ms/frame)
0.15	300.0	300	24.5
0.25	251.1	278	26.7
0.35	154.8	176	26.4
0.50	64.8	85	24.2
0.65	13.2	19	24.1

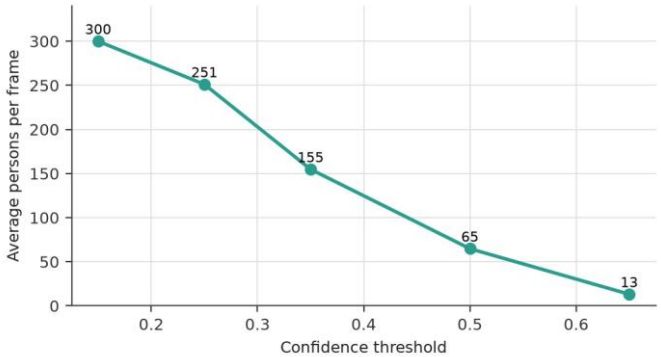


Fig. 8. Average detections per frame against the confidence threshold.

2) Inference size: Because people are small, shrinking the image removes them. At 320 px only 6.9 people per frame are found, at 640 px 154.8 and at 960 px 240.9, while the time per frame grows from 8.9 to 41.7 ms (Table VIII). A person of 19 px becomes a few pixels at low resolution, which the network cannot resolve. The model was trained at 640 px, so its behaviour is best understood there.

TABLE VIII EFFECT OF THE INFERENCE SIZE (JATAYU V2, CONFIDENCE 0.35, EVERY 2ND FRAME).

Inference size (px)	Avg persons per frame	Inference (ms/frame)
320	6.9	8.9
480	71.4	14.0
640	154.8	24.1
960	240.9	41.7



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511114

3) Frame skip: Analysing every 1st, 2nd, 3rd, 5th or 10th frame moved the estimated mean count only between 154.8 and 155.6 while cutting the work in proportion. This was estimated by sampling an all-frame run of a slowly changing 8 s scene; in a fast-changing scene a large skip would delay alerts.
4) Grid size: A finer grid shares the same crowd among more zones. The peak count in any zone was 176, 53, 33, 25 and 16 for 1 × 1, 2 × 2, 3 × 3, 4 × 4 and 6 × 6 grids, so a threshold of 15 that flags all nine zones of a 3 × 3 grid would flag almost nothing on a 6 × 6 grid. Grid and threshold are really one design decision.

VI. DISCUSSION

The most practical finding is how much the choice of detector matters for this kind of view. A small model trained for it found people where the generic nano model found none, at the same cost per frame. The second is that absolute totals are fragile. Because they depend so strongly on the confidence threshold and the inference size, comparisons between zones and over time are more trustworthy than any single number, and we would present the output to an operator in that way. On the laptop used, the pipeline needed 11.7 s for the 8 s clip at frame skip 2, so it is suitable for offline analysis of recorded footage. Live use would need a larger frame skip, a smaller region of interest or faster inference. Finally, the recall of 0.58 and the effect of non-maximum suppression mean that counts in dense regions are lower bounds. The thresholds and the grid have to be tuned for each camera view, and we kept the analysis layer simple so that an operator can see where each number comes from.

VII. LIMITATIONS

Only one 8-second clip was run end to end, so we cannot say how the system behaves in other scenes, lighting or camera heights. The clip has no ground-truth counts, so the counting error is unknown. The validation metrics refer to the detector's own validation split; we did not record a separate test split, and we cannot report the size or composition of the training dataset because the checkpoint does not store them. Counts are detections per frame, not unique people, since there is no tracking. The heatmap and density labels are not calibrated physical measures, and a congestion flag is a threshold event, not a prediction of a crowd disaster. Finally, the timing of the main run and of the sensitivity runs comes from different machines and library versions.

VIII. CONCLUSION AND FUTURE WORK

We presented an offline framework that combines a custom-trained YOLO11n person detector with zone occupancy, density labels, a detection-frequency heatmap, event logging and a dashboard that runs on a CPU-only laptop and keeps the footage on the local machine. On an aerial clip the custom detector found about 155 people per frame where COCO-pretrained baselines found fewer than one. The comparison is qualitative, and the absolute counts depend strongly on the confidence threshold and the inference size.

The next steps are clear. First, we want to evaluate against manually annotated frames, report the count error and choose thresholds on evidence, using a larger and more varied dataset with a separate test split. Second, tracking [12], [17] would give counts of unique people and dwell times. Third, a camera homography would allow density in people per square metre to be compared with level-of-service standards [14]. Further work includes live RTSP streams with alerts and faster inference (GPU, ONNX or OpenVINO), density-map counting for extremely dense scenes [5], [10], and privacy features such as face blurring and access control for stored outputs.

Author statement. All four authors took part in the project and have read and approved this manuscript. The work is original and is not under consideration elsewhere.

Competing interests. The authors declare no competing interests.

Funding. This work received no external funding.

Ethics. The system detects persons only. It performs no face recognition, identification or biometric processing.
Data and code availability. The detector weights and source code are available from the corresponding author on reasonable request. The evaluation clip is third-party stock footage and cannot be redistributed.

Use of AI tools. The authors used an AI writing assistant (Anthropic's Claude) to help restructure and edit the language of this manuscript. The project, experiments and results are the authors' own, and the authors take full responsibility for the content.



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511114

ACKNOWLEDGMENT

We thank our project guide, Mrs. S. Dhivya, Assistant Professor, Department of CSE (Emerging Technologies), SRM Institute of Science and Technology, Vadapalani Campus, for her guidance.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *in Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.
[2] G. Jocher and J. Qiu, "Ultralytics YOLO11," *2024*. [Online]. Available: https://github.com/ultralytics/ultralytics
[3] T.-Y. Lin et al., "Microsoft COCO: Common Objects in Context," *in European Conference on Computer Vision (ECCV)*, 2014, pp. 740–755.
[4] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.
[5] Y. Li, X. Zhang and D. Chen, "CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes," *in Proc. IEEE CVPR*, 2018, pp. 1091–1100.
[6] V. A. Sindagi and V. M. Patel, "A survey of recent advances in CNN-based single image crowd counting and density estimation," *Pattern Recognition Letters*, vol. 107, pp. 3–16, 2018.
[7] Streamlit Inc., "Streamlit documentation." [Online]. Available: https://docs.streamlit.io
[8] S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *in Advances in Neural Information Processing Systems (NIPS)*, vol. 28, 2015.
[9] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," *in Proc. IEEE CVPR*, vol. 1, 2005, pp. 886–893.
[10] Y. Zhang, D. Zhou, S. Chen, S. Gao and Y. Ma, "Single-Image Crowd Counting via Multi-Column Convolutional Neural Network," *in Proc. IEEE CVPR*, 2016, pp. 589–597.
[11] A. Neubeck and L. Van Gool, "Efficient Non-Maximum Suppression," *in Proc. 18th Int. Conf. Pattern Recognition (ICPR)*, vol. 3, 2006, pp. 850–855.
[12] A. Bewley, Z. Ge, L. Ott, F. Ramos and B. Upcroft, "Simple Online and Realtime Tracking," *in Proc. IEEE Int. Conf. Image Processing (ICIP)*, 2016, pp. 3464–3468.
[13] D. Helbing, I. Farkas and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, pp. 487–490, 2000.
[14] J. J. Fruin, *Pedestrian Planning and Design*. New York: Metropolitan Association of Urban Designers and Environmental Planners, 1971.
[15] P. Dollár, C. Wojek, B. Schiele and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 743–761, 2012.
[16] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv:1804.02767*, 2018.
[17] N. Wojke, A. Bewley and D. Paulus, "Simple online and realtime tracking with a deep association metric," *in Proc. IEEE ICIP*, 2017, pp. 3645–3649.
[18] P. Zhu et al., "Detection and tracking meet drones challenge," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 44, no. 11, pp. 7380–7399, 2022.

 This work is licensed under a Creative Commons Attribution 4.0 International License