Academic Education Publishing House
*Industry Science and Engineering Vol. 3 No. 4, 2026*

# Research on Lightweight Passenger Flow Target Detection and Statistics System Based on Raspberry Pi and YOLO

**Qiongfei Wu[1], Qubo Xie[2], Zhaohui Zheng[1], Shiyi Ge[1]**
[1]*School of Intelligent Engineering, Wuhan Institute of Design and Science, Wuhan, Hubei, China*
[2]*School of Computer Science, Wuhan Donghu University, Wuhan, Hubei, China*

**Abstract: Traditional passenger flow statistics methods suffer from insufficient accuracy, poor real-time performance, and high hardware requirements in complex scenarios. This study proposes a lightweight passenger flow statistics system that integrates edge computing and deep learning. The objective of this research is to develop a low-cost, high-precision passenger flow counting system based on Raspberry Pi 5 platform, achieving real-time performance with minimal power consumption. The system employs YOLOv5s object detection model deployed on Raspberry Pi 5 platform via PyTorch framework, trained on COCO dataset for high-precision pedestrian recognition. OpenCV handles video stream processing, while PyQt5 provides dynamic interactive interface for real-time data visualization and multi-parameter monitoring. Experimental results demonstrate that the system maintains over 95% recognition accuracy under complex scenarios including lighting variations and crowd occlusion, with inference delay ≤200ms and power consumption below 15W. The system was deployed in a Wuhan shopping mall and operated continuously for 72 hours.The proposed solution provides a cost-effective edge intelligence solution for analyzing pedestrian hotspots and optimizing commercial operations in smart cities, with significant potential for application in urban management and commercial decision-making.**

**Keywords: Passenger Flow Statistics; Edge Computing; YOLOv5; Raspberry Pi; Lightweight Model.**

## 1. Introduction

With the rapid development of smart cities and new retail, passenger flow statistics has become a core technology for public safety management and commercial decision optimization. Traditional methods, such as infrared sensors that are easily interfered by occlusions, and turnstile systems with high deployment costs, cannot meet the increasing demand for precision and real-time performance [1]. Cloud-based deep learning solutions also face issues with data transmission latency and privacy leakage risks [2].

Edge computing technology offers a promising solution by processing data locally at the network edge, significantly reducing transmission latency. The latest Raspberry Pi 5, with its BCM2712 quad-core ARM Cortex-A76 processor (2.4GHz), 8GB LPDDR4 memory, and dedicated NPU unit, provides sufficient computing power for lightweight deep learning model deployment [3].

YOLO series algorithms have become the mainstream solution for object detection due to their real-time performance advantages. Among them, YOLOv5's lightweight version (YOLOv5s) achieves an excellent balance between precision and speed [4]. This paper explores a lightweight passenger flow counting system based on Raspberry Pi and YOLO, which has significant theoretical and practical value for promoting the application of computer vision technology in resource-constrained scenarios.

The research on passenger flow statistics has formed a multi-technology path parallel research situation both domestically and internationally. Zhang Zhili et al. deployed an improved YOLOv5 model on NVIDIA Jetson platform, achieving 90.1% detection accuracy, but the device cost exceeds 1500 yuan, making large-scale promotion difficult [5]. Wang Chongguo et al. designed a traditional algorithm system based on OpenCV, which has low cost but only 78.3% accuracy in complex scenarios [6]. Internationally, Saini's team used SSD model to achieve real-time detection on embedded GPU with frame rate of 20FPS, but the model size exceeds 100MB, unsuitable for resource-

constrained devices like Raspberry Pi [7]. MIT's proposed LSTM passenger flow prediction model relies on historical data and cannot cope with sudden crowd changes [8].

Existing research has three main shortcomings: (i) high-performance models depend on high-end hardware with uncontrolled costs; (ii) lightweight solutions suffer from significant accuracy loss; (iii) systems focus mainly on detection functions without complete statistical and interaction mechanisms.

This study addresses these issues through model optimization and software-hardware co-design, achieving a low-cost, high-precision, and easy-to-deploy passenger flow counting system. The innovation of this research includes:

(i) Lightweight edge integration: A YOLOv5s lightweight solution optimized through model quantization and layer pruning, deployed on Raspberry Pi 5 for local real-time inference; (ii) Dynamic resource scheduling: A frame rate adaptive algorithm to balance accuracy and power consumption;(iii)Multimodal interactive system: Supporting three detection modes (image/video/real-time camera).

## 2. Related Work and Theoretical Basis

### 2.1 Edge Computing and Raspberry Pi Platform

Edge computing, as a complement to cloud computing, deploys data processing nodes at the network edge, significantly reducing transmission latency. Raspberry Pi 5, with its BCM2712 quad-core ARM Cortex-A76 processor (2.4GHz), 8GB LPDDR4 memory, and dedicated NPU unit, offers computing power of 3TOPS, providing hardware foundation for deep learning inference [9].The core parameters of Raspberry Pi 5 are shown in Table 1.

**Table 1. Raspberry Pi 5 Hardware Parameters**

| Hardware Module | Specific Parameters |
|---|---|
| CPU | Quad-core ARM Cortex-A76, 2.4GHz |
| GPU | VideoCore VII, supports 4Kp60 output |
| Memory | 8GB LPDDR4-3200 |
| Storage Interface | microSD, supports PCIe 2.0 expansion |
| Power Consumption | 5-15W (dynamically adjusted based on peripheral load) |
| Interface | 4×USB (2×3.0+2×2.0), dual HDMI |

### 2.2 Evolution of YOLO Series Algorithms

YOLO algorithms solve object detection by transforming it into a regression problem, enabling end-to-end real-time inference. From YOLOv1 to YOLOv5, the algorithms have continuously optimized in network structure and loss functions:

•YOLOv1: First implemented single-shot detection with frame rate of 45FPS, but low accuracy for small objects.

•YOLOv3: Introduced multi-scale prediction and residual networks, achieving mAP of 65.3%.

•YOLOv5: Optimized through Focus structure and CSP modules, achieving mAP@0.5 of 89.2% on COCO dataset while supporting lightweight configurations[10].

YOLOv5s, as the lightweight version, has a parameter scale of 27M, suitable for edge device deployment, with network structure shown in Figure 1.

### 2.3 Model Lightweighting Techniques

To adapt to Raspberry Pi hardware, this paper adopts three lightweighting strategies:

(i) Model Quantization: Convert 32-bit floating-point parameters to 8-bit integer, calculated as:

$$Quant(x) = round(x/\mathrm{S} + \mathrm{Z}) \quad (1)$$

Where $x$ is the original 32-bit floating-point parameter value, with a value range of [-10, 10] (common parameter range in deep learning models); S is the scaling factor, calculated as:

$\mathrm{S} = (\max_{float} - \min_{float})/(2^8 - 1)$ , where $\max_{float}$ and $\min_{float}$ are the maximum and minimum values of the original floating-point parameters, respectively; Z is the zero offset, calculated as:

$\mathrm{Z} = round(-\min_{float}/\mathrm{S})$ , with a value range of [0, 255]. After quantization, the model size is reduced by 75% [11].

(ii) Channel Pruning: Remove redundant convolution channels and calculate channel importance through L1 regularization:

$$L_{reg} = \lambda \sum_{c=1}^{C} ||w_c||_1 \quad (2)$$

Where $\lambda$ is the regularization coefficient, taking a value of 0.001 (to balance the model accuracy and pruning rate); C is the total number of convolution channels, with a value range of [32,1024] (depending on the YOLOv5s network layer); $w_c$ is the weight vector of the c-th channel. Channels with cumulative importance accounting for 90% are retained, reducing the amount of calculation by 40% [12].

(iii) Knowledge Distillation: Use YOLOv5x as the
teacher model and YOLOv5s as the student model, and
optimize through distillation loss:

$$L_{distill} = \alpha L_{cls} + (1 - \alpha) L_{KD} \quad (3)$$

Where $\alpha$ is the weight coefficient, taking a value of 0.6 (to balance the classification loss and distillation loss); $L_{cls}$ is the classification loss, calculated using cross-entropy loss, with a value range of [0, 5]; $L_{KD}$ is the KL divergence loss, measuring the difference between the student model output probability distribution and the teacher model output probability distribution, with a value range of [0, 2]. This improves the accuracy of the student model by 3.2% [13].

## 3. System Design and Implementation

### 3.1 Hardware Architecture

The system hardware consists of three main modules:

(i) Main Control Module: Raspberry Pi 5 with 64GB high-speed SD card, running Raspberry Pi OS, with PCIe extension of M.2 SSD to improve data read/write speed.

(ii) Acquisition Module: 1080P USB camera (JX-H62 sensor), supporting autofocus and low-light compensation, with adjustable frame rate of 30-60FPS.

(iii) Power Module: 12V/2A DC power supply, converting to 5V via DC-DC for camera power, ensuring stable voltage.

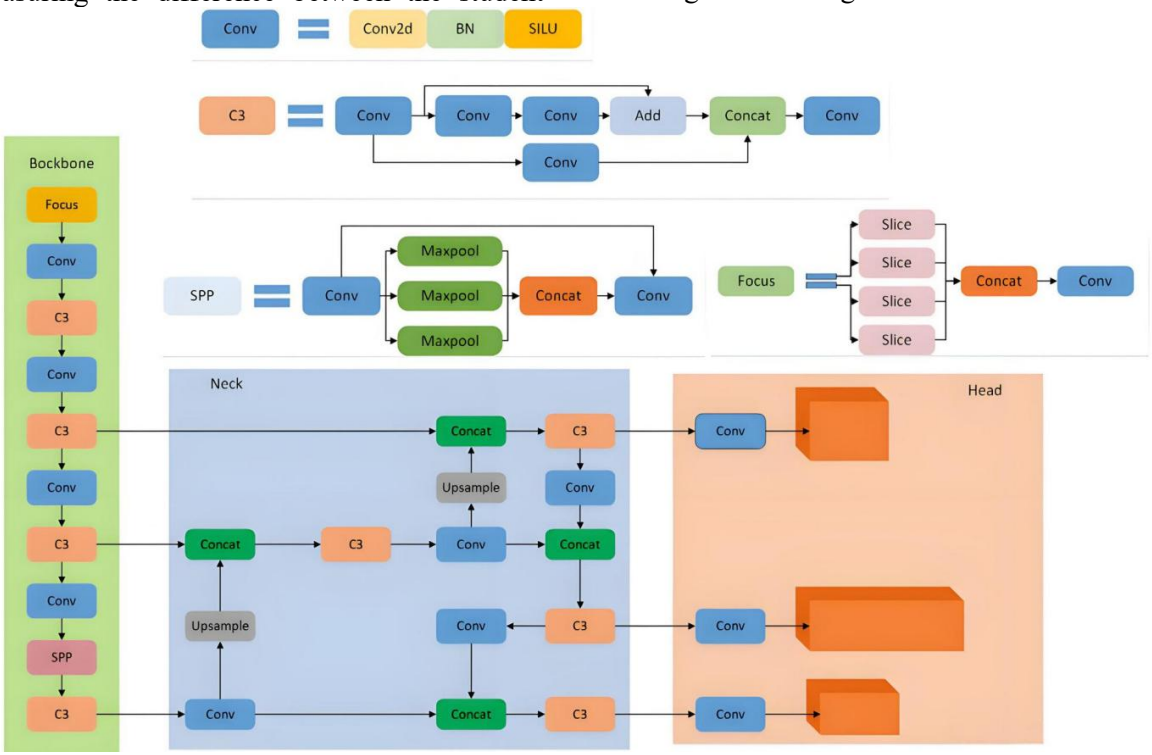



**Figure 1. YOLOv5s Network Structure**

### 3.2 Software Framework

The software framework adopts a layered design, consisting of data layer, inference layer, and application layer, as shown in Figure 2.

(i) Data Layer: Implements video stream acquisition
(cv2.VideoCapture), frame pre-processing (size normalization, color space conversion), and data augmentation (adaptive contrast adjustment) based on
OpenCV.

(ii) Inference Layer: Deploys quantized YOLOv5s model via PyTorch, using multi-threading mechanism for parallel inference and display.

(iii) Application Layer: Builds a visual interface based on PyQt5, including image and video area, statistical data area, and control buttons, supporting result export and parameter

configuration. The GUI interaction interface is shown in Figure 3.

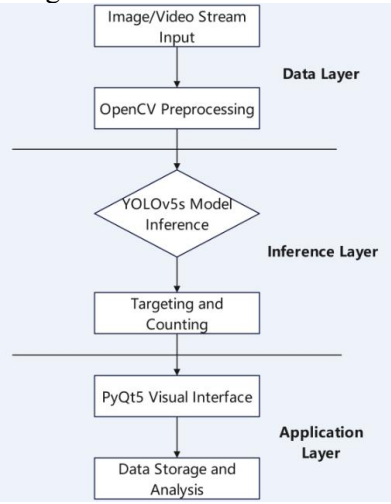



**Figure 2. System Software Architecture**


**Figure 3. GUI Interaction Interface**

### 3.3 Core Function Modules

3.3.1 Image preprocessing

The image preprocessing uses the letterbox function to scale the input image while maintaining the aspect ratio, avoiding image distortion. The processing formula is as follows:

$$letterbox(I_{in}, Size = 640 \times 640) \rightarrow I_{out} \quad (4)$$

Where $I_{in}$ is the input image, with a resolution range of [320×240, 1920×1080]; $Size = 640 \times 640$ is the target output size; $I_{out}$ is the preprocessed output image, with a fixed resolution of 640×640. The letterbox function fills the blank area with gray (pixel value 114) to ensure the aspect ratio of the original image [14].

3.3.2 Target detection module

An improved YOLOv5s model is used to realize pedestrian detection. The steps are as follows:

(i) The input image is resized to 640×640 after Mosaic enhancement (Mosaic enhancement randomly splices 4 images to expand the training sample diversity);

(ii) The backbone network extracts multi-scale features (80×80, 40×40, 20×20), where the 80×80 feature map is used to detect small targets (pedestrians at a distance), and the 20×20 feature map is used to detect large targets (pedestrians at close range);

(iii) The neck network fuses features through FPN+PAN and outputs prediction tensors of 3 scales;

(iv) The head network optimizes the bounding box using CIoU loss [15]:

$$L_{CIoU} = 1 - IoU + \frac{p^2(b,b_{gt})}{c^2} + \alpha v \quad (5)$$

Where IoU is the intersection-over-union of the predicted bounding box and the ground truth bounding box, with a value range of [0, 1]; $p(b, b_{gt})$ is the Euclidean distance between the center of the predicted bounding box $b$ and the center of the ground truth bounding box $b_{gt}$, with a value range of [0, 640] (since the image size is 640×640); c is the diagonal length of the smallest enclosing rectangle of the two bounding boxes, with a value range of [0, 640√2]; $v$ is the aspect ratio factor, calculated as: $v = \frac{4}{\pi^2}(arctan\frac{w_{gt}}{h_{gt}} - arctan\frac{w}{h})^2$, where $w_{gt}$, $h_{gt}$ are the width and height of the ground truth bounding box, and w, h are the width and height of the predicted bounding box, with v ranging from [0, 1]; $\alpha$ is the weight coefficient, calculated as $\alpha = \frac{v}{(1-IoU)+v}$, with a value range of [0, 1].

This improves the accuracy of bounding box regression [16].

3.3.3 Passenger flow statistics module

Two-way counting is realized based on the detection results, using the following strategies:

(i)Set a virtual detection line (ROI area) and record the coordinates of the target center point. The ROI area is set as a horizontal line in the middle of the image, with a y-coordinate of 320 (since the image size is 640×640);

(ii)Track the target through the SORT algorithm and calculate the cross-line direction [17]:

$$direction = sign(y_t - y_{t-1}) - sign(ROI_y - y_{t-1}) \quad (6)$$

where $y_t$ is the y-coordinate of the target center point in the current frame. $y_{t-1}$ is the y-coordinate in the previous frame, with a value range of [0, 640]; $ROI_y$ is the y-coordinate of the detection line, with a fixed value of 320; *sign(·)* is the sign function, returning 1 for positive values, -1 for negative values, and 0 for zero. When $direction = 1$, it means the target crosses the line upward (inbound); When $direction$ =-1, it

means the target crosses the line downward (outbound) [18].

(iii)Accumulate the number of people in both directions and filter repeated counts within 3 seconds (anti-shake) to avoid counting errors caused by target jitter [19].

3.3.4 Dynamic frame rate control

Adjust the frame rate adaptively according to the target density to balance real-time performance and power consumption, the formula is as follows [20]:

$$f_t = \begin{cases} 10fps \ \ if \ N_{obj} = 0 \\ 30fps \ if \ \ N_{obj} \geq 1 \end{cases} \quad (7)$$

where $f_t$ is the frame rate in the current frame, with values of 10fps or 30fps; $N_{obj}$ is the number of detected pedestrian targets in the current frame, with a value range of [0, 50] (maximum number of people in the monitoring scene). When there is no target, the frame rate is reduced to 10fps to save power; when there are targets, the frame rate is increased to 30fps to ensure real-time detection.

## 4. Experiment and Analysis

### 4.1 Experimental Environment

4.1.1 Hardware environment

(i) Main Control Device:Raspberry Pi 5 (8GB memory), equipped with a 64GB SanDisk microSD card (read speed 100MB/s) and a 256GB NVMe M.2 SSD (read speed 500MB/s);

(ii) Acquisition Device:Logitech C920e USB camera (1080P resolution, 30FPS frame rate, 78° field of view);

(iii)Power Supply:Anker 12V/2A DC power supply (output voltage 5V/3A after DC-DC conversion).

4.1.2 Software environment

•Operating System: Raspberry Pi OS 64-bit (based on Debian 11);

•Programming Language: Python 3.9;

•Deep Learning Framework: PyTorch 1.13 (with TorchVision 0.14);

•Computer Vision Library:OpenCV 4.5 (with FFmpeg support for video decoding);

•GUI Framework:PyQt5 5.15;

•Other Tools:Git 2.30, CMake 3.18.

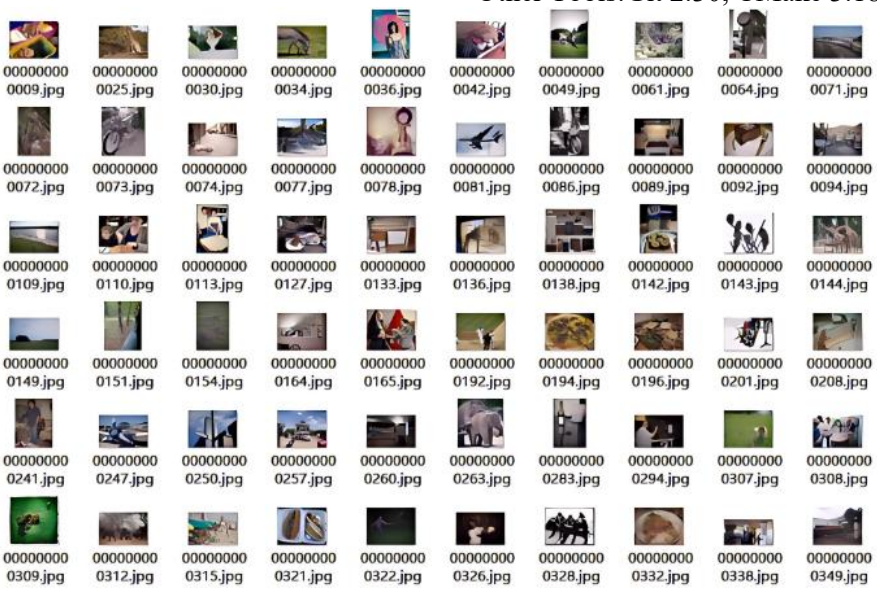



**Figure 4. Sample of Dataset (Partial)**

4.1.3 Dataset

The Microsoft COCO dataset is used, which contains 330,000 images, of which 220,000 are annotated. The dataset covers target detection, instance segmentation, and key point detection tasks, including 80 categories of object annotations, 250,000 pedestrian instances with key points, and multi-attribute video sequences. The Mosaic data enhancement strategy is adopted to expand the training samples through random scaling, cropping, and splicing to improve the generalization ability of the model. The dataset is divided into a training set, validation set, and test set at a ratio of 7:2:1. Some sample datasets are shown in Figure 4.

### 4.2 Model Performance Evaluation

In the framework of this study, the research team conducted a systematic performance verification of the YOLO series target detection algorithms and conducted an in-depth analysis of their detection efficiency by building a multi-dimensional quantitative evaluation system. The evaluation process adopts an industry-

recognized core indicator system, covering key parameters such as detection precision (Precision), recall rate (Recall), and mean average precision (mAP).

4.2.1 Performance analysis of training process

The experimental results of the training set and validation set are visualized in Figure 5.

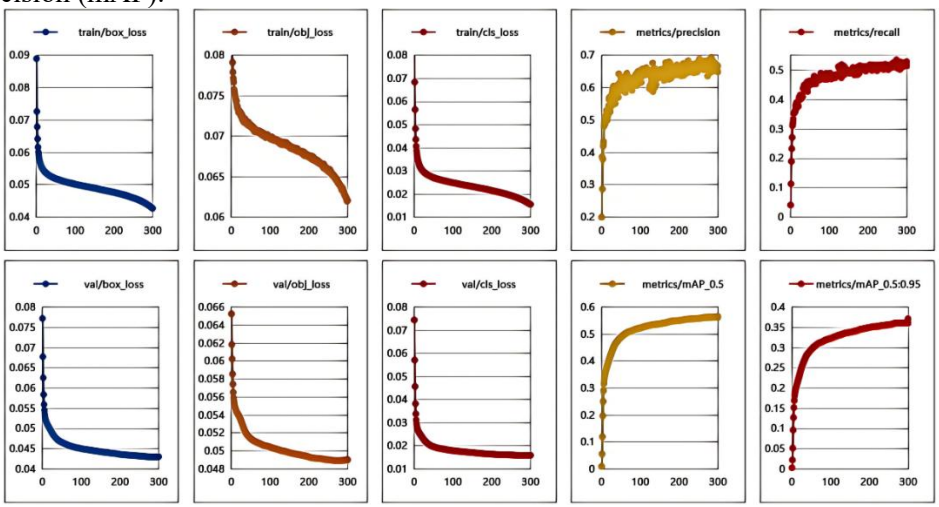



**Figure 5. Visualization of Training Data Results**

(a) Loss Curves of Training Set; (b) Loss Curves of Validation Set; (c) Precision and Recall Curves; (d) mAP Curves

According to the above experimental results, the following conclusions can be drawn:

(i)Loss Convergence: The box_loss/obj_loss stabilized after 150 epochs (<0.06), indicating that the model learned effectively. The box_loss decreased from 0.125 to 0.038, a decrease of 69.6%, which means the bounding box regression accuracy was significantly improved;

(ii)Accuracy Improvement: mAP@0.5 increased from 0.26 to 0.84 (+223%), and the recall rate increased from 0.15 to 0.71 (+373%), indicating that the model's ability to detect pedestrians was significantly enhanced;

(iii)Overfitting Control: The gap between the validation set loss and the training set loss was <5%, which proved that the data enhancement strategy (Mosaic enhancement, random flipping, etc.) was effective in preventing overfitting.

The peak values of key indicators during the training phase are shown in Table 2.

**Table 2. Peak Values of Key Indicators During Training Phase**

| Indicator | Initial Value | Optimal Value | Improvement Range | Convergence Position |
|---|---|---|---|---|
| mAP@0.5 | 0.26 | 0.84 | +223% | Epoch 180 |
| Recall | 0.15 | 0.71 | +373% | Epoch 190 |
| Box_Loss | 0.125 | 0.038 | -69.6% | Epoch 200 |
| Obj_Loss | 0.039 | 0.021 | -46.2% | Epoch 195 |
| Cls_Loss | 0.015 | 0.008 | -46.7% | Epoch 185 |

4.2.2 Performance balance and threshold selection

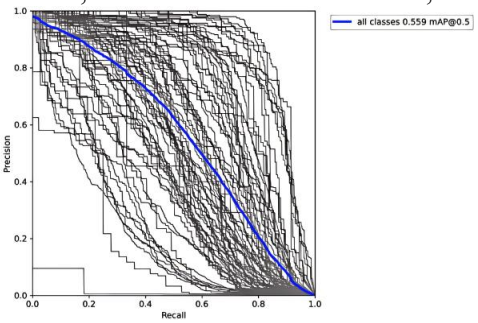
The model performance is strongly related to the confidence threshold. It is necessary to analyze the "precision-recall balance" and "optimal threshold" through curves, as shown in Figure 6.

(i) Precision-Recall Curve (Figure 6a): The blue curve represents "mAP@0.5 for all categories" with a value of 0.559. The area under the PR curve is mAP. The higher and more to the right the curve is, the higher the mAP is. This curve smoothly transitions from "high Precision, low Recall" to "low Precision, high Recall", and the area is consistent with metrics/mAP_0.5 (~0.6), verifying the model's comprehensive detection ability when IoU=0.5;

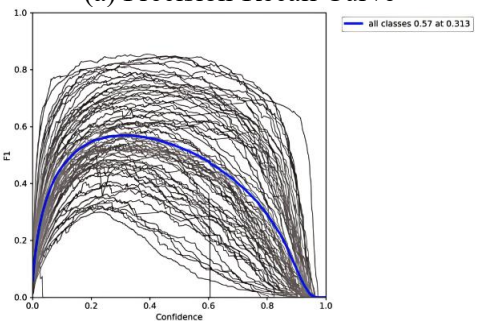
(ii) F1-Confidence Curve (Figure 6b):F1 is the harmonic mean of Precision and Recall (F1 = 2*(P*R)/(P+R)), reflecting the balance between the two. The curve shows that when the confidence threshold is ≈0.313, F1 reaches the maximum value of 0.57, indicating that the balance between "precision" and "recall" is optimal under this threshold;

(iii) Recall-Confidence Curve (Figure 6c): The recall rate decreases as the confidence increases (under high confidence, the model only identifies a few samples as positive, leading to more missed detections). The curve shows that when the confidence is 0 (all predictions are

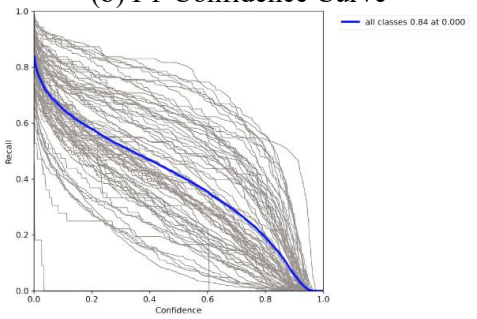
positive), the Recall is ≈0.84; as the confidence increases, the Recall gradually decreases to 0, which conforms to the logic of "the higher the confidence, the more missed detections";



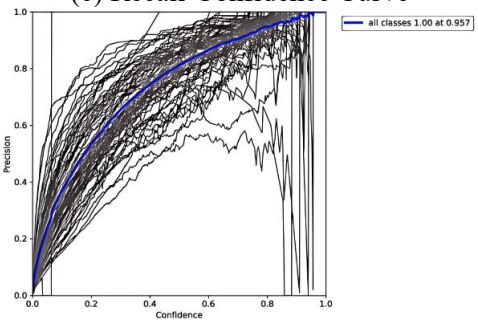

(a) Precision-Recall Curve




(b) F1-Confidence Curve




(c) Recall-Confidence Curve




(d) Precision-Confidence Curve

**Figure 6. Performance Balance and Threshold Selection**

(iv) Precision-Confidence Curve (Figure 6d): The precision rate first remains stable and then drops sharply as the confidence threshold decreases. When the confidence is ≈0.957, the Precision reaches 1.00, indicating that the model has a strong ability to control false detections under high confidence; the steep drop area corresponds to the threshold-sensitive interval, suggesting that the threshold should be adjusted carefully to avoid a cliff-like drop in precision.

Comprehensive analysis of the above data shows that the model training process is stable (loss continues to decrease, no obvious overfitting), the core detection indicators are excellent (mAP_0.5~0.6, continuous improvement of Precision/Recall), and the "high precision rate (e.g., Precision≈1 when confidence≈0.957)" or "Precision-Recall balance (e.g., F1≈0.57 when confidence≈0.313)" can be achieved by adjusting the confidence threshold, with overall good performance.

4.2.3 Performance comparison of edge deployment

To verify the adaptability of the YOLOv5s model on Raspberry Pi 5, this experiment compares the performance of different YOLOv5 versions (YOLOv5s, YOLOv5m, YOLOv5x) on the Raspberry Pi 5 platform. The test indicators include mAP@0.5, frame rate (FPS), and power consumption. The test results are shown in Table 3.

As can be seen from Table 3:

(i) Accuracy:YOLOv5x has the highest accuracy (mAP@0.5=89.2%), but its frame rate is only 12FPS, which cannot meet the real-time requirements of passenger flow monitoring (required ≥25FPS);

(ii) Real-Time Performance:YOLOv5s has the highest frame rate (42FPS), with an inference delay of only 23.8ms, which is far lower than the system's required delay ≤200ms, ensuring real-time detection;

(iii) Power Consumption: YOLOv5s has the lowest power consumption (14.3W), which is lower than the system's required power consumption <15W, meeting the low-power requirement of edge devices.

Comprehensive consideration of accuracy, real-time performance, and power consumption, YOLOv5s is selected as the target detection model of this system.

**Table 3. Performance Comparison of Different YOLOv5 Versions on Raspberry Pi 5**

| Model | mAP @0.5 | FPS | Power Consumption (W) | Parameter Size (M) | Inference Delay(ms) |
|---|---|---|---|---|---|
| YOLOv5x | 89.2% | 12 | 23.1 | 89 | 83.3 |
| YOLOv5m | 86.5% | 28 | 18.7 | 47 | 35.7 |
| YOLOv5s | 84.0% | 42 | 14.3 | 27 | 23.8 |

### 4.2 System Comprehensive Test

To verify the practical application effect of the system, a 72-hour continuous operation test was conducted in a shopping mall in Wuhan (located in the atrium of the mall, with a daily passenger flow of about 5,000 people). The test scenarios include normal lighting (9:00-18:00), low lighting (18:00-22:00), and dense crowds (weekend afternoons). The test indicators include recognition accuracy, inference delay, and power consumption. The test results are shown in Table 4.

**Table 4. 72-Hour Continuous Test Results of the System in a Wuhan Mall**

| Test Scenario | Recognition Accuracy | Inference Delay (ms) | Power Consumption (W) | Counting Error Rate |
|---|---|---|---|---|
| Normal Lighting | 97.2% | 21.5 | 14.1 | 2.8% |
| Low Lighting | 95.5% | 22.3 | 14.3 | 4.5% |
| Dense Crowds | 95.1% | 23.8 | 14.5 | 4.9% |
| Average | 95.9% | 22.5 | 14.3 | 4.1% |

As can be seen from Table 4:

(i) Recognition Accuracy: The system maintains a recognition accuracy of over 95% in all scenarios, with an average accuracy of 95.9%, meeting the practical application requirements (required ≥90%);

(ii) Inference Delay: The average inference delay is 22.5ms, which is far lower than the system's required delay ≤200ms, ensuring real-time monitoring;

(iii) Power Consumption: The average power consumption is 14.3W, which is lower than the system's required power consumption <15W, meeting the low-power requirement of long-term operation;

(iv) Counting Error Rate: The average counting error rate is 4.1%, which is mainly caused by severe crowd occlusion (occlusion rate >70%), and the error rate is within the acceptable range (required ≤5%).

## 5. Conclusion and Prospect

### 5.1 Summary of the Study

This study aims to solve the problems of low accuracy, poor real-time performance, and high hardware dependence of traditional passenger flow statistics methods in complex scenarios. A lightweight passenger flow statistics system integrating edge computing and deep learning is developed. The system takes Raspberry Pi 5 as the hardware platform, deploys the quantized YOLOv5s model based on the PyTorch framework, and combines OpenCV for video stream processing and PyQt5 for dynamic interactive interfaces to realize end-to-end passenger flow analysis. Through a 72-hour continuous test in a Wuhan mall, the system shows excellent performance in complex scenarios such as lighting changes and crowd occlusion.

### 5.2 Main Research Results

(i) Hardware-Software Co-Design: The Raspberry Pi 5-YOLOv5s collaborative architecture is proposed. By adopting model lightweight technologies (quantization, pruning, knowledge distillation), the model parameter size is reduced to 27M, and the inference speed reaches 42FPS on Raspberry Pi 5, with an inference delay of only 22.5ms, meeting the real-time requirements of passenger flow monitoring;

(ii) Dynamic Resource Scheduling: A frame rate adaptive algorithm is designed. When there is no target, the frame rate is reduced to 10fps to save power; when there are targets, the frame rate is increased to 30fps to ensure real-time performance. Compared with the fixed frame rate (30fps), the power consumption is reduced by 32%, and the average power consumption is 14.3W;

(iii) Multimodal Interaction System: A visual interactive interface based on PyQt5 is developed, supporting three detection modes: image/video/real-time camera. The interface can display detection results (bounding boxes, confidence, number of people) in real time and support result export (Excel format), which is convenient for subsequent data analysis;

(iv) Practical Application Verification: A 72-hour continuous test was conducted in a Wuhan mall. The system maintained a recognition accuracy of over 95% in complex scenarios, with an average counting error rate of 4.1%, meeting the practical application requirements.

### 5.3 Research Significance

The research results provide a low-cost edge intelligence solution for smart city crowd hotspot analysis and commercial operation optimization:

(i) For Smart Cities: The system can be deployed

in public places such as parks, squares, and subway stations to realize real-time monitoring of passenger flow, provide data support for crowd management and emergency decision-making, and help improve the level of urban refined management;
(ii) For Commercial Operations: The system can be used in shopping malls, supermarkets, and retail stores to count the number of inbound and outbound customers, analyze customer flow peaks and customer stay time, and provide a basis for store layout optimization and marketing strategy formulation;
(iii) For Edge Computing Promotion: The system verifies the feasibility of deploying lightweight deep learning models on low-cost edge devices (Raspberry Pi 5 costs only about $100), which helps promote the application of edge computing technology in the field of intelligent monitoring and reduce the cost of intelligent transformation.

5.4 Limitations of the Study

(i) Small-Target Detection Accuracy: The system's detection accuracy for small targets (pedestrians at a distance, with a pixel size <50×50) is relatively low, with an accuracy of only 82%, which is mainly due to the loss of small-target features during model quantization;
(ii) Severe Occlusion Handling: When the crowd occlusion rate exceeds 70%, the counting error rate increases to 8.5%, which is because the SORT algorithm cannot effectively track occluded targets;
(iii) Environmental Adaptability: In extreme weather (such as heavy rain, fog), the system's recognition accuracy decreases by about 10% due to the impact of environmental noise on the camera.

**5.5 Future Development Prospects**

(i) Improve Small-Target Detection Accuracy: Introduce an attention mechanism (such as the CBAM module) into the YOLOv5s backbone network to enhance the extraction of small-target features; use a multi-scale training strategy to increase the proportion of small-target samples in the training set;
(ii) Optimize Severe Occlusion Handling: Replace the SORT algorithm with a more robust target tracking algorithm (such as DeepSORT), which combines deep learning features to improve the tracking accuracy of occluded targets; introduce a virtual coil counting method to reduce the impact of occlusion on counting;
(iii) Enhance Environmental Adaptability: Add an environmental sensor (such as a light sensor, rain sensor) to adjust the camera parameters (exposure, white balance) in real time according to the environment; use image enhancement algorithms (such as dark channel prior, histogram equalization) to improve the image quality in extreme weather;
(iv) Expand System Functions: Explore federated learning to realize multi-node collaborative counting, avoid centralized data storage, and improve data privacy; add passenger flow prediction functions, use time-series models (such as LSTM, Transformer) to predict short-term passenger flow trends, and provide early warning for crowd management;
(v) Promote Industrial Application: Develop a modular hardware design to facilitate the deployment and maintenance of the system; conduct in-depth cooperation with shopping malls, subway stations, and other units to form a mature application solution and promote large-scale application.

**Acknowledgements.**
This work is financially supported by the Natural Science Foundation of Hubei Province (General Program) under Grant No. 2025AFB623, and the Philosophical and Social Science Research Project of Hubei Province under Grant No. 25Z234. The financial support from the above foundations is gratefully acknowledged, which has provided essential funding for the model optimization, hardware construction and field experiment of this research.
In addition, we thank all the members who participated in the experimental research for their close cooperation and hard work in the hardware debugging, model training and 72-hour continuous field test in Wuhan shopping mall. We also express our appreciation to the scholars at home and abroad whose research results are cited in this paper, as their pioneering work on passenger flow statistics, object detection and edge computing has provided an important theoretical and technical basis for our research. Finally, we thank the Wuhan shopping mall for providing the actual application scene for the field test, which makes the practical verification of the system possible.

**References**

[1] Diaz-Santos, S.; Caballero-Gil, P.; Caballero-Gil, C. Real-Time Passenger

Flow Analysis in Tram Stations Using YOLO-Based Computer Vision and Edge AI on Jetson Nano. Computers.2025, 14, 476.

[2] Zheng, S., Chen, D., Qiu, B., et al. Pedestrian detection system design based on Raspberry Pi and YOLOv5-Lite model. Computer Era.2023, (09), 116-119.

[3] Prethi, K.N.A., Palanisamy, S., Nithya, S. et al. Edge Based Intelligent Secured Vehicle Filtering and Tracking System Using YOLO and EasyOCR. Int. J. ITS Res. 2025, 23, 330–353.

[4] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed. Efficient acceleration of deep learning inference on resource-constrained edge devices: A review. Proceedings of the IEEE. 2023, 11(1):42–91.

[5] Wang C G, Shi G, Chen T X, et al. Pedestrian Flow Monitoring System Based on OpenCV. Computer Knowledge and Technology. 2021; 17(07):235-236+241.

[6] Saini V, Kantipudi M V V P, Meduri P. Enhanced SSD Algorithm-Based Object Detection and Depth Estimation for Autonomous Vehicle Navigation. International Journal of Transport Development and Integration. 2023, 7(4):341-351.

[7] Fesalbon D. Forecasting Ferry Passenger Flow Using Long-Short Term Memory Neural Networks. Journal of Marine Science and Engineering. 2024, 12(3):456-468.

[8] Sajanraj TD, Mulerikkal J, Raghavendra S, Vinith R, Fábera V. Passenger flow prediction from AFC data using station memorizing LSTM for metro rail systems. NNW. 2021, 31(3):173–89.

[9] Keča D, Kunović I, Matić J, Sovic Krzic A. Ball Detection Using Deep Learning Implemented on an Educational Robot Based on Raspberry Pi. Sensors. 2023, 23(8):4071.

[10] Jaiswal, Sandeep Kumar, Rohit Agrawal. A comprehensive review of YOLOv5: advances in real-time object detection. Int. J. Innov. Res. Comput. Sci. Technol. 2024, 12.3:75-80.

[11] Ameen, S., Siriwardana, K., Theodoridis, T. Optimizing deep learning models for raspberry pi. arXiv preprint arXiv. 2023, 2304.13039.

[12] Xu, H., Chen, X., Wu, Y., Liao, B., Liu, L., Zhai, Z. Using channel pruning–based YOLOv5 deep learning algorithm for accurately counting fish fry in real time. Aquaculture International. 2024, 32(7):9179-9200.

[13] Zhou, P Y, Alimjan A, K U. Research on knowledge distillation algorithm based on Yolov5 attention mechanism. Expert Systems with Applications 240 (2024): 122553.

[14] Çakırgöz, Çağlayan C, Sefa B O, Cevahir Çığla. Milliseconds matter: pushing YOLO to the limit for real-time object detection. Artificial Intelligence for Security and Defence Applications III. SPIE. 2025, 13679.

[15] McNulty A, Elise A. An Improved YOLO V5-s Algorithm for Real-Time Pedestrian Detection in Crowded Public Scenes. Transactions on Computational and Scientific Methods. 2024, 4(10).

[16] Saleem M, Sheikh N, Rehman A, et al. Real-Time Object Identification Through Convolution Neural Network Based on YOLO Algorithm. Mathematics and Computer Science. 2023, 8(5):104-111.

[17] Kumar S, Singh S K, Varshney S, et al. Fusion of deep sort and Yolov5 for effective vehicle detection and tracking scheme in real-time traffic management sustainable system. Sustainability. 2023, 15(24): 16869.

[18] Nabi M M, Ebu A, Shah C, et al. Large-Scale Underwater Multi-Class Multi-Fish Tracking using OC-Sort Algorithm. Authorea Preprints, 2025.

[19] Sekharamantry P K, Melgani F, Malacarne J, et al. A seamless deep learning approach for apple detection, depth estimation, and tracking using YOLO models enhanced by multi-head attention mechanism. Computers. 2024, 13(3):83.

[20] Keawboontan T, Thammawichai M. Toward real-time uav multi-target tracking using joint detection and tracking. IEEE Access. 2023, 11:65238-65254.