SMATIKA: STIKI Informatics Journal
Accredited Sinta 4 Ministry of Higher Education, Science and Technology Republic of Indonesia Decree No. 79/E/KPT/2023
E-ISSN: 2580-6939 || P-ISSN: 2087-0256

Real-Time Crowd Counting for Smart City Park Monitoring Using YOLOv11 and a Microservices Architecture

M.Sultonun Naim[1*], Salamun Rohman Nudin[2]

[1,2] *Surabaya State University, Vocational, Informatics Management, Prof. Moch. Yamin Street, Ketintang, Gayungan District, Surabaya, East Java 60231, Indonesia*

Keyword

Computer Vision; Crowd Counting; Deep Learning; Microservices; YOLOv11

Corresponding Author:
msultonun.22065@mhs.unesa.ac.id

Abstract

The provision of accurate and easily accessible public information regarding the condition of urban parks remains a challenge in the management of public open spaces. People still do not have a system that allows them to monitor park crowd levels directly, making real-time observation of park conditions difficult. Previous studies on crowd counting have mainly focused on improving object detection performance, while the implementation of crowd monitoring systems as public information services remains limited. Therefore, this research integrates the YOLOv11 algorithm with a microservices architecture to provide real-time crowd information through a public park monitoring website. This research develops a park visitor counting information system based on computer vision using the YOLOv11 algorithm to detect and count crowds from CCTV video streams. The system is designed using a microservices architecture with the FastAPI framework to support real-time detection and data integration into the Surabaya park monitoring website. The research process involves several stages, including dataset preparation, data labeling using Roboflow, YOLOv11 model training, selection of the most optimal optimizer, and implementation of the system on the detection backend. The results show that the YOLOv11m model with the SGD optimizer achieved the best performance, obtaining an mAP@50 score of 92.76%, a recall value of 89.75%, and an F1-score of 90.07%. In addition, the system successfully performed real-time crowd detection and counting under various crowd density levels, lighting conditions, and CCTV camera angles.

1. Introduction

Urban parks are public spaces in urban areas that serve as accessible and welcoming recreational spaces for the community, and offer numerous benefits in mitigating the negative impacts of urban growth. In addition, these parks can be enjoyed by people from all walks of life and serve as significant symbols of social interaction[1].To support the smart city initiative, it is necessary to enhance the convenience and efficiency of public services. This includes providing information about city parks, such as their locations, available services, and accurate information on park occupancy[2].

Crowd counting is a field within computer vision used to estimate the number of individuals in a given observation area. Therefore, this study uses YOLOv11, which was selected for its balance between inference

speed and detection accuracy, making it suitable for the needs of CCTV-based monitoring of crowds in city parks [3].Deep learning-based systems often require collaboration with other platforms to function properly. Therefore, a microservices architecture was chosen to separate each element of the system, making it more structured, easier to develop, and enabling the integration of deep learning models without the need to overhaul the entire system. This approach also aligns with modern system development practices that emphasize scalability through the separation of independent services and efficiency in maintenance by reducing dependencies between elements, thereby enabling rapid adaptation to changes without disrupting the entire operation[4].

Recent studies have demonstrated the effectiveness of YOLOv11 for object detection and crowd monitoring tasks. A real-time human detection and counting system based on YOLOv11 has been shown to automatically monitor human movement in a smart campus environment; however, the study primarily focused on detection performance and did not address the integration of the model into a scalable information system [5]. Similarly, YOLOv11 has been successfully applied for crowd counting and behavior analysis in crowded environments, demonstrating effective real-time monitoring performance. Nevertheless, the proposed framework was limited to surveillance analysis and did not provide crowd information services that could be accessed directly by the public [6]. In addition, recent studies have shown that YOLOv11 introduces architectural improvements that enhance feature extraction, detection accuracy, and computational efficiency compared with previous YOLO versions [7]. Despite these advancements, existing studies have mainly emphasized model performance evaluation, while limited attention has been given to integrating YOLOv11-based crowd counting with a microservices architecture to support real-time public information services. To address this gap, the present study develops a YOLOv11m-based crowd counting system integrated with a microservices architecture for providing real-time crowd information in urban parks in Surabaya.

2. Research methods

2.1 Research Stages

This study used the Software Development Life Cycle (SDLC) method in the development of a YOLOv11-based crowd counting system. This method is used to ensure a structured system development process, from planning, analysis, design, implementation, and testing[8].

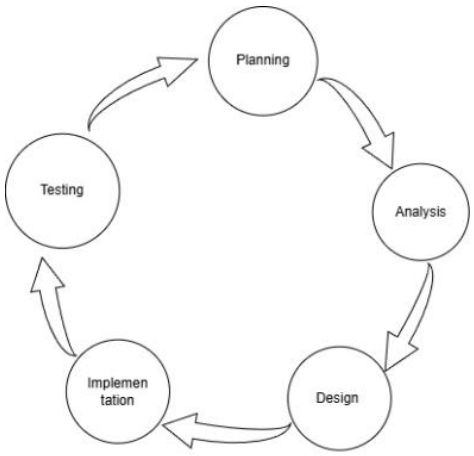

Figure 1.SDLC

At the planning stage, information was collected regarding the YOLOv11 algorithm[9], the use of FastAPI in microservices architecture[10], as well as the use of Roboflow for the dataset labeling process[11]. Next, the analysis phase is carried out to determine system requirements, dataset requirements, communication flows between services, and model training parameters to be used. The design phase includes designing the microservices system architecture, system flowcharts, crowd detection flows, and API design for data

communication between services and frontend monitoring. Then, the implementation phase is carried out through the process of extracting video frames into a dataset, preprocessing and data augmentation, training the YOLOv11 model, selecting the best optimizer, and integrating the model into the FastAPI backend. The final stage is system testing using the blackbox testing method to ensure the system can run according to requirements[12]. This is followed by the YOLOv11 model development process for crowd detection and counting. The model development flow is shown in Figure 2.

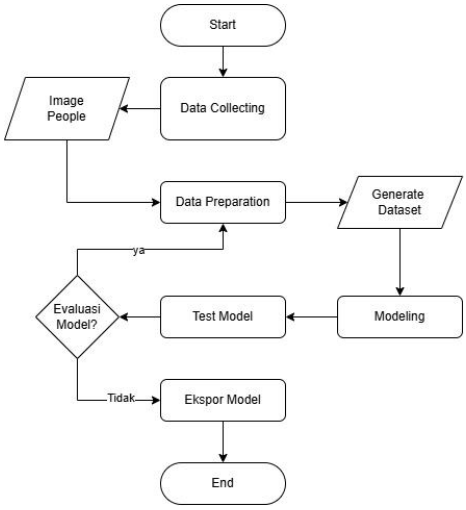

Figure 2. Flowchart of model development

The model development process begins with data collection, followed by data preparation and dataset labeling. The dataset is then used to train the YOLOv11 model, followed by model testing and evaluation. If the model results are not optimal, the process returns to the data preparation stage. However, if the model results are satisfactory, the model is exported and used in a real-time crowd detection system[13].

2.2 Dataset Processing

2.2.1 Dataset Preparation

The dataset used in this study comes from two sources, namely the results of CCTV video frame extraction and pedestrian dataset as supporting data for model training[14].The dataset contains images of people with varying levels of crowding, lighting, camera angles, and backgrounds.

Table1. Number of Datasets

Dataset Sharing	Amount	Sample dataset
Padestrian Dataset	349	
CCTV Video	534	

The dataset consists of 883 images sourced from two sources: 349 images from a pedestrian dataset and 534 images extracted from CCTV video. Two dataset sources were used to obtain a wider variety of data so that the model could learn the characteristics of human objects under various environmental conditions. The dataset used covers variations in crowd density, lighting conditions, backgrounds, and different camera angles corresponding to real-world conditions in urban parks. The combination of the pedestrian dataset and CCTV frames was chosen to increase sample diversity and more specifically represent the system's operational conditions. Additionally, the dataset covers conditions ranging from low to high crowd density, allowing it to be used to evaluate the model's ability to detect human objects at various crowd density levels. Thus, the dataset serves not only as training data for the model but also as a representation of the real-world conditions the system will face upon implementation.

2.2.2 Labeling Dataset

The dataset labeling process was carried out using the Roboflow platform using the bounding box method to mark each human object in the data. This study used one object class, namely "person."[15] After labeling, the dataset was processed through a preprocessing stage consisting of auto-orientation, resizing to 640×640 pixels, and filtering out null annotations. Next, data augmentation was performed using horizontal flip, rotation (−10° to +10°), saturation (±20%), brightness (±20%), exposure (±15%), blur (up to 1.1 pixels), and noise (up to 1.01%) to increase data variation and reduce the potential for overfitting[16]. This process increased the dataset size from 883 to 1,776 images. Since only a single object class was used,

Figure 3. Data with bounding box

there was no class distribution imbalance. Additionally, a combination of pedestrian datasets and CCTV frames with crowd levels ranging from low to high was used to improve the model's generalization ability. The dataset was then split into training, validation, and testing sets in a 70:20:10 ratio before being exported to the YOLO format (.txt).

2.3 YOLOv11 Architecture Model

You Only Look Once (YOLO) is a method for detecting objects directly by utilizing a Convolutional Neural Network. YOLO applies a single neural network to recognize objects in a single image. This network utilizes features from the entire image to predict each bounding box and probability directly in a single evaluation stage [13]. In this study, several variants of the YOLOv11 model were tested, namely YOLOv11n, YOLOv11s, and YOLOv11m to compare the performance of human object detection[17].The training process was carried out using a previously prepared dataset with an input size of 640×640 pixels and a total of 100 epochs.

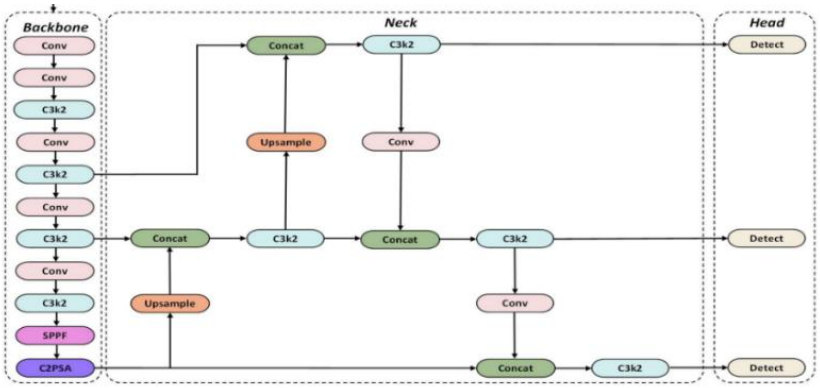

Figure 4. YOLOv11 architecture

After obtaining the optimal model variant, several optimizers, namely Adam, SGD, and RMSprop, were tested to improve model training performance. The optimizer with the best evaluation results was then used as the final model implemented in a real-time crowd detection system.

Table 2. Optimizer Parameters

Parameter	Value/Amount used
Epoch	100
Batch Size	16
Initial Learning Rate	0.0001
Optimization	Adam, RMSProp, SGD

In this step, the previously developed system will test the separated data. The results will be analyzed using a confusion matrix, which aims to obtain the results of the model created in the form of accuracy, precision, recall, and F1-Score values, as seen in Equations (1), (2), (3), and (4).

$$Accuracy = \frac{\text{TP}}{\text{TP+FP+FN}} \times 100\% \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \times 100\% \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \times 100\% \quad (3)$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \times 100\% \quad (4)$$

The Confusion Matrix is used to assess the performance of a classification model, while the Accuracy Matrix shows how precise or accurate the model is in classifying. The Precision Matrix is used to measure the proportion of data predicted as positive that is truly positive compared to the total data predicted as positive. The Recall Matrix is used to assess the proportion of data predicted as positive that is truly positive from all data that is truly positive. The F1-Score Matrix is the average between precision and recall, which aims to find a balance between the two matrices.[18].

2.4 System Implementation

In the implementation phase, the trained YOLOv11 model is integrated into a real-time crowd detection and counting system. The system is built using a microservices architecture, a collection of processes that interact with each other to create a complex application based on any API language. Microservices are an evolution of Service-oriented Architecture because microservices are a system consisting of small, separate, and task-

focused service-block components or operating through modular and independent methods for their respective purposes, but still connected to each other regularly[19][20].In this research, FastAPI is used as the main backend in a microservices architecture to connect crowd detection processes, location data management, and API gateway communication in real-time[21].

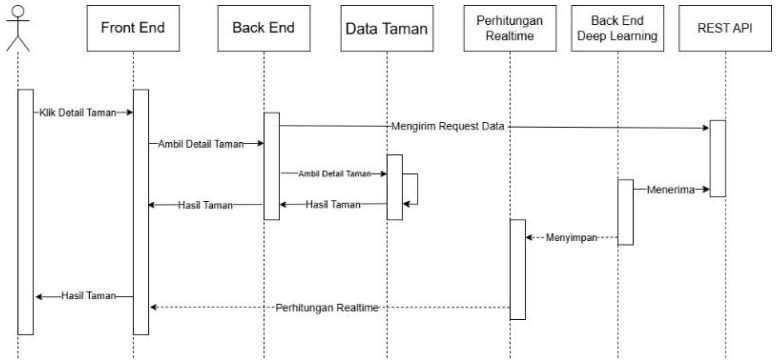

Figure 5. Sequence Diagram

The diagram explains the communication flow between services in a microservices-based crowd detection system. The frontend monitoring requests data from the backend service to obtain location information and real-time crowd count results. The backend then communicates with the location service and the YOLOv11

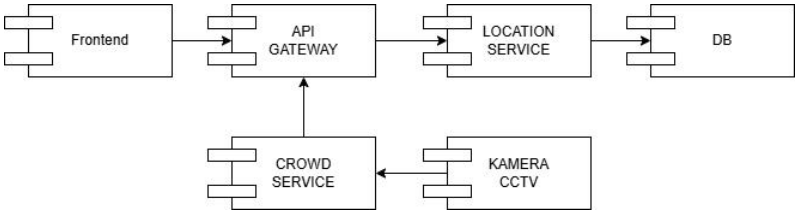

Figure 6. Component Diagram

detection service via the REST API to process the CCTV stream and generate crowd count data to be displayed on the frontend monitoring. This approach allows for separate detection processes, making the system easier to develop and capable of handling multiple CCTV sources simultaneously[13].

The system's main components consist of CCTV cameras, an API gateway, a crowd service (YOLOv11), a location service, a database, and a frontend website. Video from the CCTV cameras is processed by the crowd service using YOLOv11 to detect and count the number of visitors. The detection results are then stored in the database and displayed in real time on the website.

3. Results and Discussion

3.1 YOLOv11 Training and Validation

Training was carried out by testing several model variants, namely YOLOv11n, YOLOv11s, and YOLOv11m to compare detection performance based on mAP, precision, recall and F1-score values before being implemented in a real-time crowd detection system.

3.1.1 YOLOv11 Model Variant Comparison

In the initial stage, several YOLOv11 model variants, namely YOLOv11n, YOLOv11s, and YOLOv11m, were tested to compare their performance in human object detection. All models were trained using the same training parameters for 100 epochs. Model performance was compared based on mAP@50, mAP@50-95, precision, and recall. The test results for each model variant are shown below.

Table 3. YOLOv11 Model Training

Model	Results			
	MAP@50	MAP@50-95	Precision	Recall
Nano	83.91	62.35	83.28	76.93
Small	85.10	64.93	85.51	78.29
Medium	85.01	66.70	83.84	79.10

Each YOLOv11 model variant was tested using the same training parameters with 100 epochs to ensure consistent comparisons. The YOLOv11s model achieved the highest mAP@50 and precision values, but YOLOv11m was chosen as the primary model due to its better recall and mAP@50-95 values compared to the other variants. A higher recall value indicates the model's ability to optimally detect human objects in various crowd conditions, while a higher mAP@50-95 value indicates more stable detection performance at various IoU levels. Therefore, YOLOv11m is considered more suitable for implementing a real-time crowd detection system.

3.1.2 Optimizer Test Results

In this study, three optimizers were used: Adam, SGD, and RMSprop with the same training parameters to compare the performance of the YOLOv11m model. Evaluation was conducted using mAP@50, mAP@50-95, precision, recall, and F1-score to determine the best optimizer for human object detection. The results of the optimizer testing are shown below.

Table 4. Optimizer Testing

Optimizer	Results				
	MAP@50	MAP@ 50-95	Precision	Recall	F1 score
Adam	93.23	66.17	89.88	89.08	89.48
SGD	92.76	66.66	90.40	89.75	90.07
RMSprop	93.08	66.60	90.54	89.47	90.00

All three optimizers performed well, with mAP@50 scores above 92%. The Adam optimizer achieved the highest mAP@50 score of 93.23%, while RMSprop achieved the highest precision score of 90.54%. Nevertheless, the SGD optimizer was selected as the primary optimizer because it produced the highest recall value of 89.75%, the highest mAP@50-95 of 66.66%, and the highest F1-score of 90.07%. In crowd detection and counting systems, the ability to detect as many human objects as possible is a critical aspect for minimizing uncounted objects. Additionally, a high F1-score indicates a good balance between precision and recall. With a relatively small difference in mAP@50 compared to Adam just 0.47% SGD is considered to offer more balanced performance and is suitable for real-time crowd detection and counting system implementation.

3.1.3 Training and Validation Results

After obtaining the best optimizer, the training and validation process was analyzed using the SGD optimizer. The model training results graph is shown below.

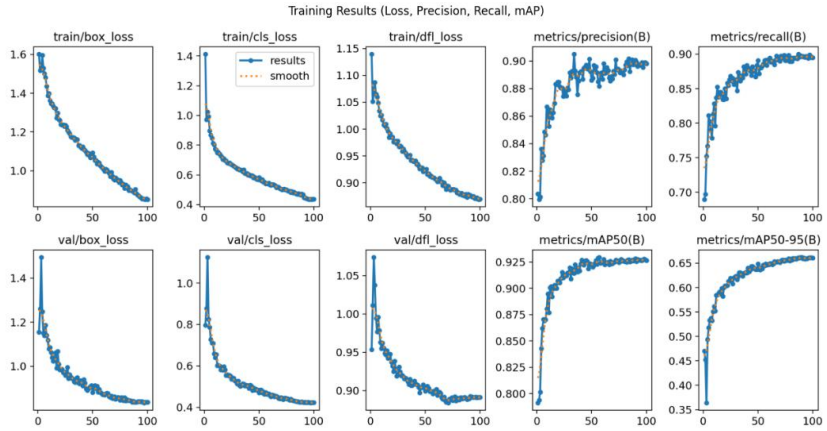

Figure 7. Training Results

The model training results show a gradual decrease in loss values during the training and validation processes. The train box loss decreased from approximately 1.6 to 0.85, the train classification loss decreased from 1.4 to approximately 0.44, and the train DFL loss decreased to approximately 0.87. During the validation process, the validation box loss decreased to approximately 0.84, the validation classification loss decreased to approximately 0.42, and the validation DFL loss reached approximately 0.89. In addition, the precision value increased to approximately 0.90 and the recall reached approximately 0.89. The mAP@50 value also increased to approximately 0.92, while the mAP@50-95 value reached approximately 0.66 at the end of training. These results indicate that the model is able to learn data patterns well and has stable human object detection performance for application in real-time crowd detection and counting systems.

3.1.4 Confusion Matrix Evaluation

The evaluation process is used to observe and measure the model's performance in detecting objects. In this study, a Confusion Matrix was used. The following is a visualization of the Confusion Matrix:

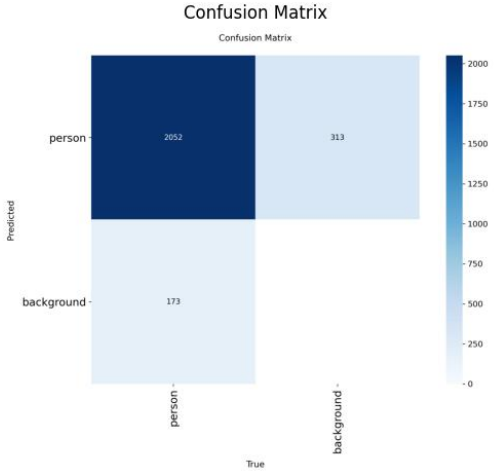

Figure 8. Confusion Matrix

The confusion matrix results show 2052 True Positive values. This indicates that the model correctly recognized most of the human objects in the dataset. However, there were 313 False Positive values, indicating that some background areas were detected as human objects. Furthermore, there were 173 False Negative values, indicating that some human objects were not detected. The results indicate that the model has fairly stable performance in detecting human objects in various crowd conditions.

3.1.5 Comparison with Previous Research

Table 5. Research Comparison on Human Detection and Counting

Researchers	Model	MAP@50
[22]	YOLOv2	90,73%
[23]	YOLOv3	83.4%
[24]	YOLOv4	74.3%
[25]	YOLOv5	82.1%
[26]	YOLOv8s	78.3%
[5]	YOLOv11	85%
Proposed model		92.76%

Previous studies have applied various YOLO architectures for human detection and crowd counting. YOLOv2, YOLOv3, YOLOv4, and YOLOv5 have achieved mAP@50 values of 90.73%, 83.4%, 74.3%, and 82.1%, respectively, demonstrating the effectiveness of successive YOLO architectures for real-time human detection tasks [22], [23], [24], [25]. More recent approaches based on YOLOv8s and YOLOv11 have also reported competitive mAP@50 values of 78.3% and 85%, confirming the continued advancement of YOLO-based methods for human detection and crowd counting [5], [26]. Despite these improvements, existing studies have primarily focused on detection accuracy and real-time monitoring, while limited attention has been given to integrating crowd counting results into scalable public information systems. To address this gap, the present study integrates a YOLOv11m-based crowd counting model with a microservices architecture to provide real-time crowd information services. The proposed system achieves an mAP@50 of 92.76%, demonstrating that the integration of an advanced object detection model with a scalable microservices architecture can support both high detection performance and real-time public information delivery.

3.2 Realtime System Implementation

The YOLOv11 model was then implemented in a real-time CCTV-based crowd detection and counting system. The system implementation was conducted to determine the model's ability to detect human objects and display crowd level information in real-world environments.

3.2.1 System Implementation Results

The YOLOv11 model is capable of detecting human objects by providing a bounding box for each detected object and displaying the crowd category directly on the monitoring system.

Figure 9. System Implementation Results

Detection results are displayed in real time, allowing users to monitor crowd conditions directly through the frontend. In addition to displaying bounding boxes around each human object, the system also displays crowd size information to facilitate more effective monitoring of CCTV areas.

3.2.2 System Testing

Testing includes crowd levels, lighting conditions, CCTV camera angles, and system function testing using the blackbox testing method.

Table 6. Crowd Level Testing

Condition	Actual Amount	Number of Detections	Difference
Low	5	5	0
Medium	32	24	8
High	52	38	14

Based on the crowd level testing results, the system achieved good detection accuracy under low-density conditions but experienced detection gaps of 8 and 14 objects in medium- and high-density scenarios, respectively. This performance degradation can be attributed to inter-object occlusion, where extensive bounding-box overlap causes the standard Non-Maximum Suppression (NMS) algorithm in YOLOv11 to suppress true positive detections, a limitation that has also been reported in previous studies [27]. To mitigate this issue, future systems may incorporate density-aware attention mechanisms, such as the Convolutional Block Attention Module (CBAM), to improve the model's ability to focus on unoccluded human features, particularly head regions, thereby enhancing detection performance in densely crowded environments [27].

Table 7. Lighting Condition Testing

Condition		Actual Amount	Number of Detections	Difference
Bright		5	5	0
Dark		15	8	7

Based on the lighting testing results, the system achieved perfect detection in bright conditions but suffered a 7-object gap in dark settings. This drop occurs because low-light contrast reduces YOLOv11's convolutional feature extraction capability, a limitation that can be mitigated by integrating a Zero-DCE preprocessing pipeline to boost image illumination prior to inference [28].

Based on the camera angle testing results, the system achieved accurate detection at ideal positions but suffered an 11-object gap at non-ideal angles. This drop occurs because extreme perspective distortions shrink far-away objects, causing their spatial features to vanish during YOLOv11's downsampling process, which can be mitigated by incorporating multi-scale feature fusion or a homography matrix for perspective correction[29].

Table 8. Camera Angle Testing

Condition		Actual Amount	Number of Detections	Difference
High angle		5	5	0
Low angle		20	9	11

Based on the test results, the system achieved an MAE of 5.71, an MSE of 61.42, and an RMSE of 7.83. These values indicate that the system has an average calculation error of approximately 5–6 people per test. Additionally, the system is capable of near-real-time detection at a rate of 9 FPS.

Table 9. System Performance Testing

Matric	Result
Mean Absolute Error (MAE)	5,71
Mean Squared Error (MSE)	61,42
Root Mean Square Error (RMSE)	7,83
Frames Per Second (FPS)	9,00 FPS
Inference Time	111,11 ms

Table 10. Blackbox Testing

Function	Input	Expected Result	Test Results	Status
Frame Extraction	CCTV Video	The system generates video frames in real-time.	Frame successfully extracted	Succeed
Object Detection	Frames containing human objects	The system detects human objects along with confidence values.	Human object detected	Succeed
Bounding Box Visualizatio n	Detection result object	The system displays the bounding box, label, and confidence score.	Bounding box appears on the object	Succeed
Crowd Counting	Human object detected	The system displays the number of human objects automatically.	The crowd count is successfully displayed	Succeed

The blackbox testing results cover four main functions: extraction, detection, depiction, and counting. In the extraction function, the model successfully processes video containing objects and generates images per second. The detection function demonstrates that the model can detect object coordinates and accuracy well in images. Furthermore, the depiction function displays a bounding box along with class labels and accuracy values for detected objects. Finally, the counting function successfully displays the total number of objects in the image, with results consistent with the model's accuracy. All test functions showed successful results.

4. Conclusions and Future Work.

Based on the research results, the YOLOv11m model with the SGD optimizer was able to deliver good human object detection performance, with an mAP@50 of 92.76%, a recall of 89.75%, and an F1-score of 90.07%. The model was successfully implemented in a real-time CCTV-based crowd counting system using a microservices architecture and the FastAPI framework. Test results also show that the system is capable of detecting and counting crowds effectively under various conditions of crowd density, lighting, and CCTV camera angles. However, system performance still declines under conditions of high crowd density and low lighting. Future research can focus on improving detection accuracy and testing the scalability of the microservices architecture under higher loads.

5. Reference

[1] M. K. Kadri, R. Armanda, G. Purba, dan Y. Fitriani, "Kesesuaian Pengadaan Ruang Terbuka Hijau Taman Kota Berdasarkan Standar Minimal Pelayanan Penduduk di Kota Surabaya Suitability of City Park Green Open Space Procurement Based on Minimum Standards of Resident Services in the City of Surabaya," vol. 02, no. 01, hal. 95–99, 2023, doi: 10.35718/compact.v2i1.853.

[2] D. D. Saputri, "Penilaian Fungsi Taman Kota Sebagai Ruang Terbuka Publik di Kota Surabaya," vol. 13, no. 2, hal. 40–47, 2018, doi: 10.12962/j2716179X.v13i2.7113.

[3] T. Abdul, R. Shyaa, dan A. A. Hashim, "Enhancing real human detection and people counting using YOLOv8," vol. 00061, hal. 1–8, 2024, doi: https://doi.org/10.1051/bioconf/20249700061.

[4] S. Hassan, "Systematic scalability analysis for microservices granularity adaptation design decisions," no. December 2021, hal. 1378–1401, 2022, doi: 10.1002/spe.3069.

[5] F. Aldi, I. Nozomi, dan S. Rahmawati, "Real-Time Human Detection and Counting System Based on YOLOv11 for Smart Campus Environment," *JET-UMY*, vol. 9, no. 2, hal. 64–73, 2025, doi: 10.18196/jet.v9i2.29213.

[6] A. A. Lubis, A. Prasasta, dan D. A. Sari, "Towards Efficient Crowd Counting and Behavior Analysis Using," vol. 4, no. 1, hal. 35–47, 2025, doi: https://doi.org/10.63876/ijtm.v4i1.128.

[7] M. Khanam, R , Hussain, "YOLOV11: AN OVERVIEW OF THE KEY ARCHITECTURAL ENHANCEMENTS," vol. 2024, hal. 1–9, 2024, doi: 10.48550/arXiv.2410.17725.

[8] N. Dwivedi, D. Katiyar, dan G. Goel, "A Comparative Study of Various Software Development Life Cycle (SDLC) Models," *IJRESM*, vol. 5, no. 3, hal. 141–144, 2022, doi: 10.51316/ijresm.2022.1881.

[9] J. R. Terven dan D. M. Cordova-esparaza, "A COMPREHENSIVE REVIEW OF YOLO: FROM YOLOV1 TO YOLOV8 AND BEYOND," *Mach. Learn. Knowl. Extr.*, hal. 1–27, 2023, doi: 10.3390/make5040083.

[10] G. Aurelia, N. Lubis, R. Purnamasari, dan K. Saleh, "Designing a Real-Time TOXMAP Backend Based on FastAPI and Firebase for B3 Waste," vol. 8, no. 1, 2025, doi: 10.32877/bt.v8i1.2888.

[11] F. Ciaglia, F. S. Zuppichini, P. Guerrie, M. Mcquade, dan J. Solawetz, "Roboflow 100: A Rich, Multi-Domain Object Detection Benchmark," 2022, doi: 10.48550/arXiv.2211.13523.

[12] M. E. Khan, "A Comparative Study of White Box , Black Box and Grey Box Testing Techniques," *IJACSA*, vol. 3, no. 6, hal. 12–15, 2012, doi: 10.14569/IJACSA.2012.030603.

[13] F. Zufari dan Y. Irawan, "Enhancing Real Time Crowd Counting Using YOLOv8 Integrated with Microservices Architecture for Dynamic Object Detection in High Density Environments," *KESATRIA*, vol. 6, no. 1, 2025, doi: 10.30645/kesatria.v6i1.575.

[14] S. A. Rumapea dan M. F. Fawwaz, "Implementation of YOLOv8 in Object Recognition Systems for Public Area Security in Kebun Raya Bogor," *Ultim. J. Tek. Inform.*, vol. 17, no. 1, hal. 1–10, 2025, doi: 10.31937/ti.v17i1.4133.

[15] N. Sujana, M. M. Mutoffar, dan A. A. Haryanto, "ANALISIS KINERJA YOLOV8 OPTIMALISASI ROBOFLOW UNTUK DETEKSI EKSPRESI WAJAH EMOSIONAL DENGAN MACHINE LEARNING,"

NARATIF, vol. 06, no. 02, hal. 115–124, 2024, doi: 10.53580/naratif.v6i2.292.

[16] C. Shorten dan T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J. Big Data*, 2019, doi: 10.1186/s40537-019-0197-0.

[17] J. Meng, Y. Li, Z. Li, dan W. Xie, "TARGET DETECTION FOR WEEDING ROBOTS BASED ON IMPROVED YOLOv11 MODELS," *engenhariaagricola*, vol. 4430, 2025, doi: 10.1590/1809-4430-eng.agric.v45e20250032/2025.

[18] T. H. Pinem dan Z. P. Putra, "Evaluasi Kinerja Algoritma Klasifikasi Deep Learning dalam Prediksi Diabetes," vol. 17, no. 1, hal. 17–28, 2025, doi: 10.22441/fifo.2025.v17i1.003.

[19] C. S. B. Sakti dan I. Hermawan, "IMPLEMENTASI ARSITEKTUR MICROSERVICE PADA BACK END SISTEM INFORMASI I ATLANTAS BERBASIS WEBSITE," *J. Teknol. terpadu*, vol. 6, no. 2, hal. 96–104, 2020, doi: 10.54914/jtt.v6i2.281.

[20] S. Atmojo, R. Utami, S. Dewi, dan N. Widhiyanta, "Implementasi Sistem-informasi DesaBerbasisArsitektur Microservices," *SMATIKA STIKI Inform. J.*, vol. 12, no. 1, hal. 55–66, 2022, doi: https://doi.org/10.32664/smatika.v12i01.658.

[21] J. Chen, "Model Algorithm Research based on Python Fast API," vol. 3, no. 9, hal. 7–10, 2023, doi: 10.54691/fse.v3i9.5591.

[22] K. Boudjit dan N. Ramzan, "Human detection based on deep learning YOLO-v2 for real-time UAV applications Human detection based on deep learning YOLO-v2 for real-time," *J. Exp. Theor. Artif. Intell.*, vol. 00, no. 00, hal. 1–18, 2021, doi: 10.1080/0952813X.2021.1907793.

[23] R. Shi, T. Li, dan Y. Yamaguchi, "An attribution-based pruning method for real-time mango detection with YOLO network," *Comput. Electron. Agric.*, vol. 169, no. July 2019, hal. 105214, 2020, doi: 10.1016/j.compag.2020.105214.

[24] C. Wang dan H. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020, doi: 10.48550/arXiv.2004.10934.

[25] H. T. Ngoc, K. H. Nguyen, H. K. Hua, H. Vu, N. Nguyen, dan L. Quach, "Optimizing YOLO Performance for Traffic Light Detection and End-to-End Steering Control for Autonomous Vehicles in Gazebo-ROS2," vol. 14, no. 7, hal. 475–484, 2023, doi: 10.14569/IJACSA.2023.0140752.

[26] A. Syahputra, M. F. Azhary, A. Binti, A. Rahman, dan A. Saad, "Occupancy Measurement in Under-Actuated Zones : YOLO-based Deep Learning Approach," vol. 15, no. 2, hal. 757–769, 2024, doi: 10.14569/IJACSA.2024.0150277.

[27] S. Woo, J. Park, J. Lee, dan S. Kweon, "CBAM: Convolutional Block Attention Module," *Eur. Conf. Comput. Vis.*, hal. 3–19, 2018, doi: 10.1007/978-3-030-01234-2_1.

[28] C. Guo *et al.*, "Zero-Reference Deep Curve Estimation for Low-Light Image Enhancement," *Conf. Comput. Vis. Pattern Recognit.*, hal. 1780–1789, 2020, doi: 10.1109/CVPR42600.2020.00343.

[29] W. Liu, M. Salzmann, P. Fua, dan D. L. Epfl, "Context-Aware Crowd Counting," *Conf. Comput. Vis. Pattern Recognit.*, hal. 5094–5103, 2019, doi: 10.1109/CVPR.2019.00524.