KESATRIA: Jurnal Penerapan Sistem Informasi (Komputer & Manajemen)
Terakreditasi Nomor 204/E/KPT/2022 | Vol. 6, No. 1, Januari (2025), pp. 330-342

Enhancing Real Time Crowd Counting Using YOLOv8 Integrated with Microservices Architecture for Dynamic Object Detection in High Density Environments

Prihandoko[1,], Faisal Zufari[2], Yuhandri[3], Yuda Irawan[4]*
[1]Department of Information Technology, Universitas Gunadarma, Depok, Indonesia
[2]Department of Information System, Universitas Gunadarma, Depok, Indonesia
[3]Department of Computer Science, Faculty of Informatics Engineering, Universitas Putra Indonesia "YPTK" Padang, Indonesia
[4]Department of Computer Science, Universitas Hang Tuah Pekanbaru, Pekanbaru, Indonesia
E-mail: [1]pri@staff.gunadarma.ac.id, [2]faisalzufari@gmail.com, [3]yuyu@upiyptk.ac.id, [4]yudairawan89@gmail.com

Abstract

This study presents the implementation of the YOLOv8 algorithm to enhance real-time crowd counting on the ngedatedotid application, which aims to provide accurate crowd density information at various locations. The proposed model leverages the advanced capabilities of YOLOv8 in detecting and localizing head-people objects within crowded environments, even in complex visual conditions. The model achieved a mAP of 85%, outperforming previous models such as YOLO V8'S (78.3%) and YOLO V7 (81.9%), demonstrating significant improvements in detection accuracy and localization capabilities. The custom-trained model further exhibited a detection accuracy of up to 95% in specific scenarios, ensuring reliable and real-time feedback to users regarding crowd conditions at various locations. By implementing a microservices architecture integrated with RESTful API communication, the system facilitates efficient data processing and supports a modular approach in system development, enabling seamless updates and scalability. This architecture allows for independent deployment of services, thereby minimizing system downtime and optimizing performance. The integration of YOLOv8 and the custom-trained model has proven to be effective in enhancing real-time monitoring and detection of crowd density, making it a suitable solution for diverse applications that require dynamic and accurate crowd information. The results indicate that the proposed model and system architecture can provide a robust framework for real-time crowd management, which is crucial for business owners, event organizers, and public safety monitoring. Future research should consider exploring newer versions of YOLO, such as YOLO V9-S, and expanding the dataset to address challenges related to varying lighting conditions, occlusions, and object orientations. Optimizing these factors will further improve the model's accuracy and reliability, setting a new standard for crowd detection systems in public spaces and enhancing the overall user experience.

Keywords: *YOLOv8; Crowd Counting; Microservices Architecture; Object Detection; Deep Learning*

1. Introduction

The ngedatedotid app is a digital platform that provides recommendations for gathering places such as coffee shops, coworking spaces, and places to break the fast. The app is designed to meet the needs of Indonesia's younger generation who are looking for places for social interaction. However, the monolithic architecture-based features are not able to provide real-time crowd information, so users often do not have a clear picture of

ISSN: 2720-992X

the actual conditions of the location and risk visiting places that are too crowded and not according to their preferences[1].

Monolithic architectures are prone to dependencies between components, where a fault in one part can affect the entire application [2]. This dependency makes system development and maintenance difficult, as well as less resilient to technological changes and disruptions[3]. When a problem occurs with one component, the entire system is impacted, degrading performance. For a constantly evolving application like ngedatedotid, a transition to a more modular and independent microservices architecture is required[4]. Microservices divide the system into small stand-alone services, making it easier to develop, and maintain[5]. In contrast to monolithic architectures, microservices architectures allow each service to have its own responsibilities, be more flexible, and efficiently handle changes in technology and system scale[6]. This makes microservices the new standard in modern application development that demands flexibility and reliability. [7].

This research implements the YOLOv8 algorithm to support a real-time crowd counting system on the ngedatedotid application. YOLOv8 is a Convolutional Neural Network (CNN)-based deep learning algorithm capable of detecting objects quickly and accurately, even in high-density environmental conditions[8]. This algorithm was chosen for its superiority in overcoming the problem of overlap or occlusion between objects, as well as its ability to detect and localize head-people objects effectively[9]. The use of YOLOv8 enables public place managers and business owners to obtain more accurate visitor count information, which can be optimized to manage resources and ensure public safety[10]. In addition in another study YOLOv8 was able to provide real-time feedback that was previously unachievable with a monolithic architecture, making it an ideal solution in providing a better user experience and being more responsive to changing conditions[11].

In addition to detection capabilities, YOLOv8 has advantages in terms of efficiency and flexibility over other algorithms such as Faster R-CNN[12]. The YOLOv8 architecture integrates candidate boxes extraction, feature extraction, and classification in one network, allowing object detection to be performed in one efficient process[13]. Another study found that detection speed and accuracy make YOLOv8 superior in real-time situations, even on devices with limited resources[14]. In addition, the algorithm is able to provide high accuracy results, even in complex environments with different orientations and lighting variations[15]. The analysis shows that the application of YOLOv8 to the ngedatedotid application can overcome the weaknesses of conventional methods that have difficulty detecting objects in high-density conditions, making it the ultimate solution in a more accurate and responsive crowd detection system.

To support the integration of YOLOv8 in the ngedatedotid application, this research uses a microservices architecture with RESTful API as the communication standard between services. The microservices architecture divides the system into small services that can be developed and deployed independently, thus minimizing the risk of downtime when a problem occurs in one of the modules. RESTful API integration enables efficient data communication between the crowd detection service and the frontend application, supporting scalability and flexibility in system development. The results show that the combination of YOLOv8 and microservices architecture can overcome the major bottlenecks of legacy systems, providing novelty value by improving the speed, accuracy, and flexibility of the system. It is hoped that this research can be a reference for developers and researchers in creating a resilient, adaptive, and better crowd detection system in providing real-time crowd information.

2. Research Methodology

The research method applied in this paper is the waterfall model, an approach from the Software Development Life Cycle (SDLC) [16]. This method includes several steps as shown in Figure 1 below:

Figure 1. SDLC Approach

The planning stage begins with problem identification and information gathering related to the YOLOv8 algorithm, the TensorFlow platform, and the use of Roboflow for the implementation of machine learning models on the back-end of the ngedatedotid application. Analysis is carried out to determine problem boundaries, objectives, and research needs. System design includes creating a real-time crowd counting system using the CRISP-DM framework, as well as creating flowcharts and UML. Implementation is done with data preparation, model training, model export, and integration of crowd counting services into the microservices architecture. The final stage is testing using black box testing to ensure all functions work according to specifications.

High Level Design (HLD) includes a description of the entire system along with its database and a brief description of the services, systems, platforms used, and relationships between modules. The HLD design can be seen in Figure 2 below:

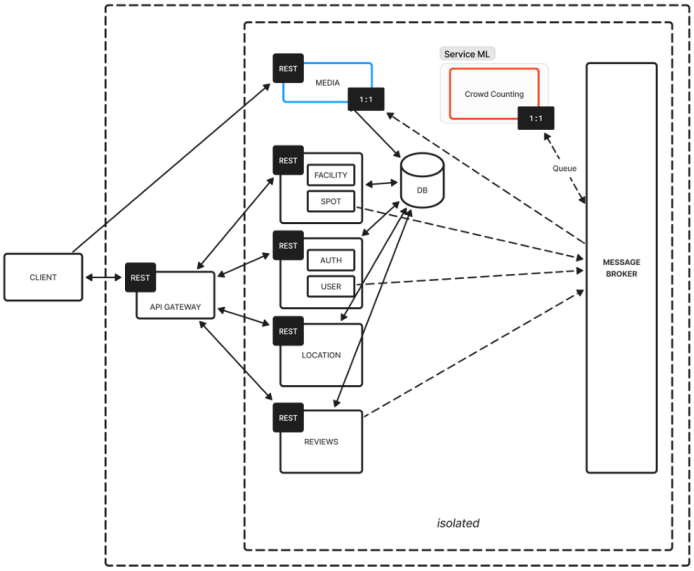

Figure 2. High Level Design

In Figure 2 is an overview of the High Level Design architecture of the ngedatedotid application, there are 4 services, namely, services reviews that function to serve requests about reviews of a place, services location that functions to serve requests about locations, auth services and users that function to serve authentication and authorization requests, services facility and spot that serve requests for places and facilities, services media that

function to serve video requests, and images, and finally crowd counting services that function to count crowds from a place in realtime.

The method used in designing this model is CRISP-DM, which is a framework that can turn business problems into data mining tasks and allow the implementation of data mining projects without being tied to the application area and technology used. There are several stages involved in building a deep learning model using the CRISP-DM framework. To make it easier to understand these stages, it can be seen in the flowchart in Figure 3.

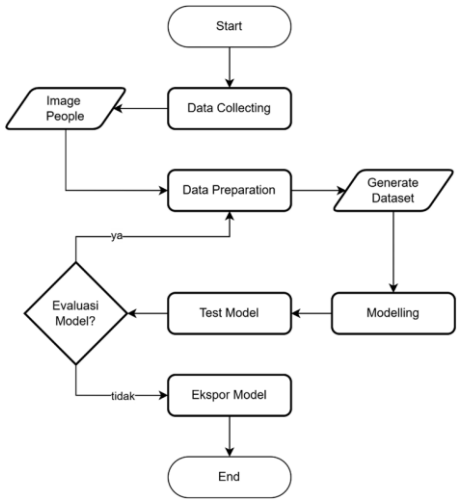

Figure 3. Flowchart for Model Creation

Figure 3 shows the flowchart of deep learning modeling using the YOLOv8 algorithm. The initial stage is data collection in the form of similar crowd images, followed by an annotation process to label the images. The annotated data is then divided into three parts: train, validation, and test for training, validation, and model testing. After data division, preprocessing is done by resizing the image to 640x640 pixels, followed by dataset generation. Next, the model training process is performed to generate metrics such as Precision, Recall, mAP50, mAP50-90, result graph, and F1-curve graph. If the training results are not optimal, model evaluation is performed by adding data or adjusting training parameters. After achieving satisfactory results, the model is exported into a suitable format, such as Tensorflow (TFJS), for implementation on a web platform to run efficiently and optimally. The flowchart in Figure 4 below shows how the model created with the YOLO algorithm works in detecting objects.

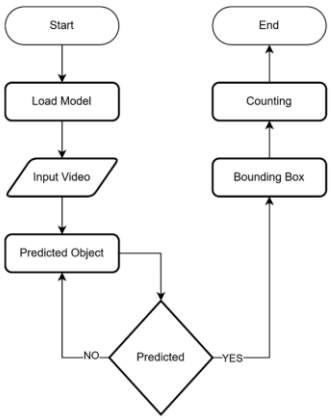

Figure 4. Flowchart of Using the YOLO Model

Figure 4 is a flowchart of the object detection process with the YOLOv8 algorithm. In this case, YOLOv8 will get the bounding box presence value and its confidence score. Confidence score is the probability value of the existence of an object in the bounding box. After the bounding box is mapped based on the resulting probability value, YOLO will predict the class of the object contained in the bounding box along with its probability, thus forming a class probability map. Of the many bounding boxes generated, to get a bounding box with a high probability, then of all the prediction results, only those that exceed the threshold will be displayed. If there is duplication in the bounding box, then Non-max Suppresion (NMS) will play a role to eliminate the duplication.

3. Results and Discussion

3.1. Data Understanding

At this stage collect and analyze the quality of the data to be used. The raw data used to train the crowd are 200 images of human crowds wearing hats, hijabs, and masks. The image data was taken directly by the researcher using a smartphone. The identified data elements are created using roboflow. If more data is obtained, the results of accuracy will also be maximized.

3.2. Data Preparation

The data processing stage includes several steps, starting from the annotation process where labels are applied to the images using a bounding box around the head object via the Roboflow platform. The dataset was then split into three subsets: train (133 images), valid (18 images), and test (10 images) to organize the data used in training and testing the YOLOv8 model. A resize step is performed to homogenize the image dimensions to 640x640 pixels, which helps with computational efficiency and processing consistency. The Auto Orient feature is applied to adjust the orientation of objects in the image, while the Flip Horizontal augmentation is used to add variety to the dataset by mirroring the image horizontally. In addition, the image is converted to grayscale to simplify the pixel values for analysis. All these stages aim to improve the quality of the dataset and ensure the model can accurately recognize objects under varying environmental conditions.

3.3. Modelling

In this modeling step the main focus is on developing and training an object detection model capable of identifying humans. In this research, the YOLOv8 architecture and the yolov8s.pt model are used as the basis for training. The model is built with the aim of recognizing objects. This process involves a model training stage utilizing a dataset that has undergone a number of pre-processing steps, including augmentation and size adjustment. The choice of using the yolov8s.pt model provides sufficient ability to detect and recognize objects in various positions and scales. Here is Figure 5 on inference with the custom model:

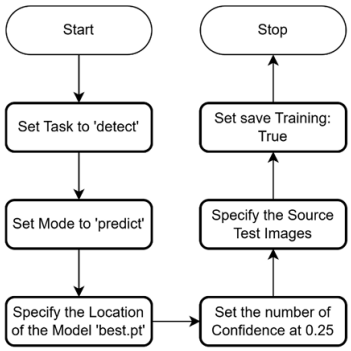

Figure 5. Inference Custom Model

In figure 5 the program starts by specifying the task to be performed by the model. In this case, the task is crowd detection. Then, determine the predict model, determine the location of the model to be evaluated by naming best.pt which is the best model, determine the confidence threshold of 0.25, determine the data source for prediction as images, and determine the storage of prediction results. Figure 6 below shows a sequence diagram that illustrates the process flow to view location details (spots) and the implementation of real-time crowd detection features.

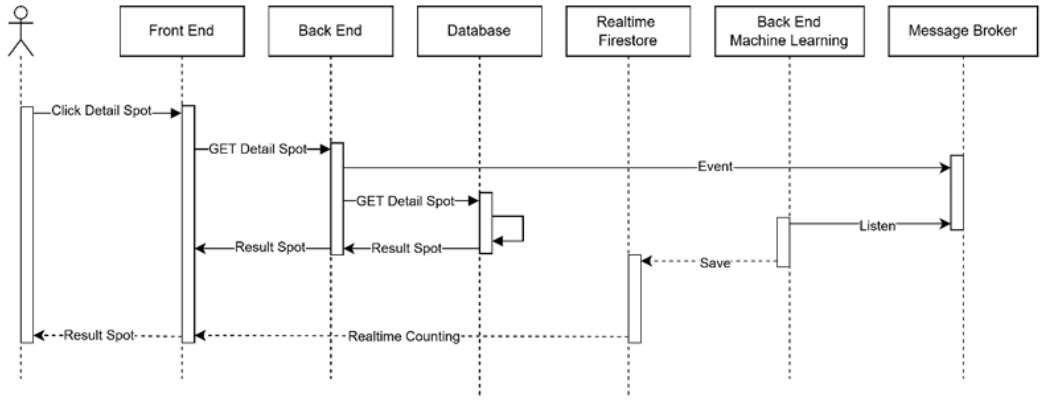

Figure 6. Sequence Diagram

The diagram above illustrates the interaction flow between the front-end and back-end of the system connected with machine learning. The process starts when a user clicks on a spot detail on the front-end, which sends a request to the back-end to retrieve information from the database. The results are sent back to the front-end and stored in the Realtime Firestore to initiate real-time crowd counting. The data is then passed to the Back-End Machine Learning component through the Message Broker, where the Deep Learning model performs object detection and calculates the crowd count. The detection results are sent back to the Realtime Firestore for display to the user, ensuring the crowd information is always up-to-date and accurate.

3.4. Evaluation

The evaluation process is used to see and measure the performance of the model in detecting objects. In this study, the authors used three evaluation methods, namely Confusion Matrix, mAP (Mean Average Precision) Metric, and using additional custom data. Here is Figure 7 about confusion matrix visualization:

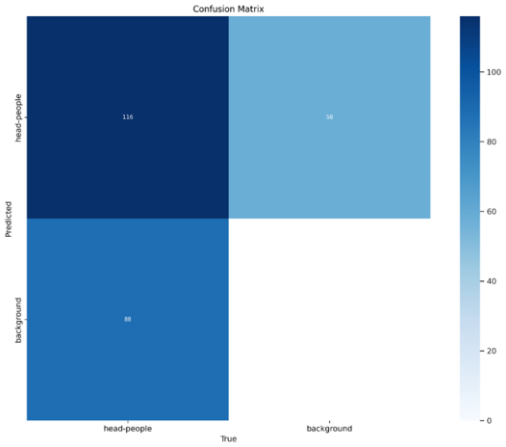

Figure 7. Confusion Matrix

Based on the matrix image above, the performance of the model can be seen from several key elements. True Positives show that the model successfully detected the head-people object correctly in 116 samples, while True Negatives show that the model also detected the background correctly in 88 samples. However, there were 58 samples of False Positives, where the model misclassified the background as head-people, indicating a detection error. The model did not record any False Negatives, meaning that the model was able to recognize all head-people objects well. Although the detection rate of head-people is quite good, the high number of False Positives indicates that the model is too sensitive to some areas that should be empty. Therefore, parameter adjustments or additional training data are required to improve accuracy and reduce detection errors. Further evaluation by visualizing it in the mAP graph presented in Figure 8.

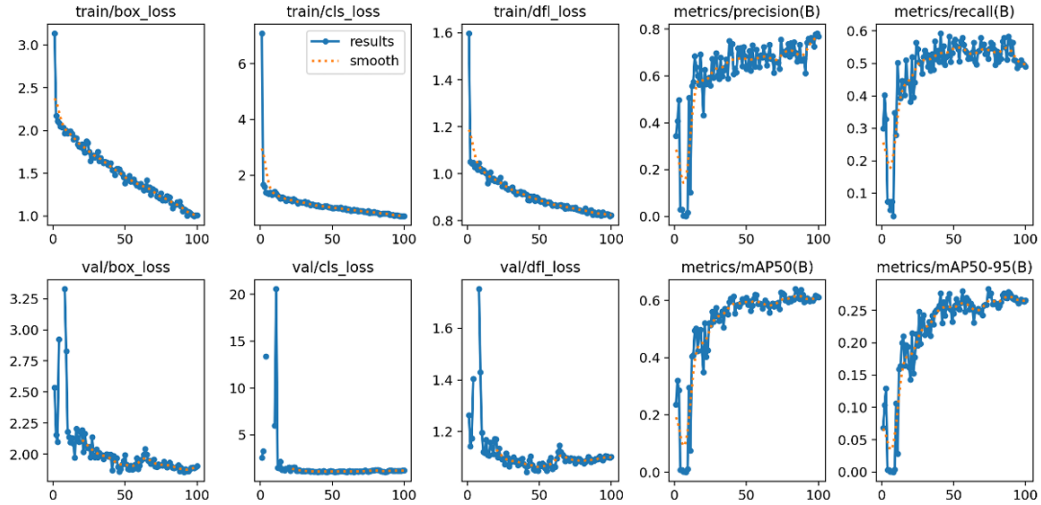

Figure 8. Evaluation results using mAP

Figure 8 above shows the results of training the model with various evaluation metrics. In the train box loss graph, there is a decrease from the initial value of 3.0 to about 1.0 at the end of training. Train classification loss also decreased significantly from a value of 6.0 to about 0.8. Meanwhile, the train distribution focal loss (DFL) decreased from an initial value of 1.5 to 0.4, indicating that the model is getting better at detecting objects that are difficult to recognize. The validation loss graph also shows a significant decrease, with validation box loss dropping from 3.25 to 1.5, validation classification loss dropping from 20.0 to 1.0, and validation DFL loss dropping from 1.6 to 0.4. The precision metrics graph increased to reach around 0.80, and recall reached the highest value of around 0.7. Meanwhile, mAP50 increased to about 0.85, and mAP50-95 reached about 0.3. These values show the improved performance of the model in detecting and classifying objects with high accuracy at various thresholds.

Furthermore an evaluation is carried out using a custom method which includes the process of prediction custom model, validate custom model, and inference with custom model. This evaluation aims to measure the performance of the model in predicting objects with varying accuracy results based on adjusted parameters. The evaluation results show that this custom model is able to provide prediction results with different levels of accuracy, which reflects the model's ability to recognize and classify objects more specifically according to research needs. The prediction results of the custom model can be seen in Figure 9 below:

Figure 9. Custom Model Prediction Results

The figure above shows the prediction results of the custom model for detecting head-people objects in outdoor environments. The model successfully identifies the position of a human head with a bounding box and varying confidence values (0.3 to 0.9). Although the model performs well in dense and complex areas, there are some false positives on objects with low confidence values, which are caused by the similarity between the head object and the background. These results show that the model is sufficiently accurate, but still needs improvement to increase accuracy in areas with high visual similarity. The experimental test results were carried out with 4 different images with the aim of seeing the success rate of the model in detecting and classifying. The test results can be seen in Figure 10 below:

Figure 10. Model Test Results

The Figure 10 displays the results of the custom model detection test on four outdoor scenarios that show successful detection of head-people objects. This detection success is influenced by several factors, such as the position of the object in the image, high image resolution, and adequate lighting levels. In images where the distance between objects is too close, the model has difficulty in detecting objects accurately due to overlapping bounding boxes. Although the model is able to recognize objects with varying confidence, the detection results show that areas with optimal lighting and object spacing provide better detection accuracy compared to areas with visual distractions or objects that are close together. Table 1 provides a comparison with previous research:

Table 1. Comparison with previous research

No	Researcher	Model	mAP
1	M. Muthumari[17]	YOLO V2	78%
2	Rui Shi [18]	YOLO V3	83.4%
3	Alexey Bochkovskiy [19]	YOLO V4	74.3%
4	Xin Zhang [20]	YOLO V5	80.88%
5	Hoang Tran Ngoc[21]	YOLO V5	82.1%
6	Tian-Hao Wu [22]	YOLO V5'S	83.36%
7	Yang Wang [23]	YOLO V7	81.9%
8	Ade Syahputra [24]	YOLO V8'S	78.3%
9	Proposed Model		85%

The table above shows that the proposed model achieves an mAP value of 85%, which is higher than the previous versions of YOLO models such as YOLO V7 (81.9%), YOLO V2 (78%), and YOLO V5 (82.1%). The model also outperforms YOLO V3 (83.4%) and YOLO V4 (74.3%), which shows an improvement in object detection and classification accuracy. This performance improvement indicates that the proposed model has a competitive advantage in more precise object detection over previous versions of YOLO.

Furthermore, testing of the ngedatedotid application is carried out to evaluate system performance after the application of the YOLOv8 algorithm that has been integrated into the microservices architecture. The testing process includes validation of the crowd detection feature and real-time data processing performed on the application backend. Testing is done using black-box testing to ensure each service functions according to specifications without checking the internal code. The results obtained from blackbox testing are shown in Table 2 below:

Table 2. Input File Testing

No	File	Input	Expected results	Output	Testing Results
1	Model.json	Insert files in the model folder on the back-end	Load model configuration	File executable	Success
2	serviceAccount Key.json	Insert files in the firebase folder on the back-end	Load firebase configuration	File executable	Success

From the black box testing table testing input files, it can be concluded that this application is running well and all input files run as expected and successfully run the input files. The following is table 3 which describes the function testing of the results of the integration of YOLO V8 with microservices architecture in ngedatedotid backend.

Table 3. Function Testing

No	Fungsi	Input	Expected results	Output	Testing Results
1	Extraction	Videos with objects	Generate images	Image per second	Success
2	Detection	Images that contain objects	Generates coordinates and accuracy	Object detected	Success
3	Drawing	Images that contain objects	Generate bounding box with class label and accuracy value	Bounding box detected	Success
4	Counting	Class and accuracy	Display the total calculation value	Amount increased	Success

Table 3 above shows the test results of the integration of the YOLOv8 algorithm with microservices architecture on the backend of the ngedatedotid application which includes four main functions, namely extraction, detection, drawing, and counting. In the extraction function, the model successfully processes videos containing objects and generates images per second. The detection function shows that the model can detect the coordinates and accuracy of objects in the image well. Next, the drawing function displays the bounding box along with the class label and accuracy value of the detected object. Finally, the counting function successfully displays the total count of objects in the image with results according to the accuracy of the model. All test functions show successful results, indicating that the crowd detection and counting system using YOLOv8 has been well integrated into the backend of the ngedatedotid application and operates according to the expected specifications.

The next test is to test the results of human detection, classification, and calculation. At this stage, the results of the three main processes of this web application are tested, namely detecting, classifying, and counting the number of humans using the YOLOv8 algorithm and the object calculation process takes place when humans are detected. The results of this experiment use 1 video extracted 1 second per frame with a duration of 1 minute 49 seconds which produces 109 images to see the success rate of this application in detecting, classifying, and counting the number of humans in realtime. The percentage of success rate is calculated using the following formula.

$$k = \frac{n}{m}\, x\, \% \qquad (1)$$

Where k = percentage of success rate, n = number of successful trial data, m = number of observation data. The following figure 11 is the result of the detection, classification, and calculation of the 109 images.

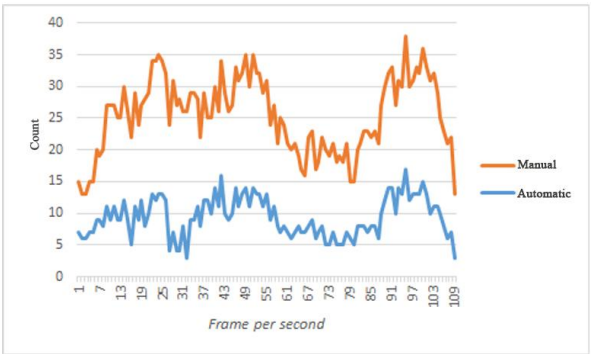

Figure 11. Comparison of Manual and Automatic calculation

The figure above shows a comparison of the number of people count results between the manual and automatic methods at 109 frames per second. The manual calculation (orange line) has a higher variation, with the number of detections ranging from 10 to 35 objects per frame. Meanwhile, the automatic calculation (blue line) has a lower and more stable number of detections, ranging from 5 to 20 objects per frame. This significant difference indicates that manual detection is more sensitive to changes in the number of objects, while the automatic model still needs to be improved to achieve accuracy close to the manual results, especially on frames with high object density. This proves that this application works well and as desired by the researcher. The following table 4 shows the results of the human detection, classification, and calculation of the number of people:

Table 4. Results of Detection, Classification, and calculation of the number of people test

No	Image	Amount Calculated in the App	Total Manual Calculation	Percentage of Success
1		17	19	89%
2		18	19	95%

Table 4 above shows the results of detection, classification, and headcount tests conducted using the app as well as the manual method. In the first image, the app successfully detected 17 people out of a total of 19 people counted manually, with a success percentage of 89%. In the second image, the app successfully detected 18 out of 19 people, with a higher success percentage of 95%. These results show that the app has a fairly good detection capability and is close to the manual calculation, although there is a small difference in the number of detections. This high percentage of success indicates that the detection model has worked optimally in identifying “head-people” objects under stable environmental conditions. In future research, we can consider using YOLO V9-S which has been further developed so as to improve the accuracy in moving object detection[25].

4. Conclusion

The conclusion of this study shows that the application of the YOLOv8 algorithm to the ngedatedotid application has significantly improved the accuracy of crowd detection. The proposed model achieves a mean Average Precision (mAP) value of 85%, higher than previous versions of YOLO models such as YOLO V8'S (78.3%) and YOLO V7 (81.9%). This performance improvement demonstrates the superiority of the YOLOv8

integration in detecting people objects in high-density environments, despite complex visual distractions. In addition, testing of the custom model shows detection success of up to 95% in certain scenarios, with high accuracy in identifying objects and providing real-time crowd information. This proves that the proposed model is superior to previous conventional models. This research also emphasizes the importance of using microservices architecture integrated with RESTful APIs to facilitate communication between services, thereby increasing the flexibility and efficiency of data processing in applications. The combination of YOLOv8, custom model, and microservices architecture proved to be able to produce a system that is more responsive and adaptive to changing environmental conditions. For further research, it is recommended to explore the latest YOLO versions such as YOLO V9-S, as well as expand the variety of datasets to overcome challenges in lighting conditions and more complex variations in object positions. Optimization of model training and testing parameters in real-world environments also needs to be done to ensure the model can achieve better performance in real-world scenarios.

References

[1] K. Gos and W. Zabierowski, "The Comparison of Microservice and Monolithic Architecture," *IEEE Access*, vol. 1109, no. June, pp. 150–153, 2020, doi: 10.1109/MEMSTECH49584.2020.9109514.

[2] Y. Abgaz *et al.*, "Decomposition of Monolith Applications Into Microservices Architectures : A Systematic Review," *IEEE Trans. Softw. Eng.*, vol. 49, no. 8, pp. 4213–4242, 2023.

[3] F. Tapia, M. Á. Mora, W. Fuertes, H. Aules, E. Flores, and T. Toulkeridis, "From Monolithic Systems to Microservices: A Comparative Study of Performance," *Appl. Sci.*, vol. 10, no. 17, pp. 1–35, 2020, doi: 10.3390/app10175797.

[4] G. Liu, B. Huang, and Z. Liang, "Microservices : architecture , container , and challenges," *IEEE Access*, vol. 1109, no. December, pp. 629–635, 2020, doi: 10.1109/QRS-C51114.2020.00107.

[5] N. Singh *et al.*, "Load balancing and service discovery using Docker Swarm for microservice based big data applications," *J. Cloud Comput.*, vol. 12, no. 1, p. 4, 2023, doi: 10.1186/s13677-022-00358-7.

[6] S. Li *et al.*, "Understanding and Addressing Quality Attributes of Microservices Architecture : A Systematic Literature Review," *Inf. Softw. Technol.*, vol. 131, no. March, pp. 1–30, 2020, doi: 10.1016/j.infsof.2020.106449.

[7] S. Ben Atitallah, M. Driss, and H. Ben Ghzela, "Microservices for Data Analytics in IoT Applications: Current Solutions, Open Challenges, and Future Research Directions," *Procedia Comput. Sci.*, vol. 207, no. June, pp. 3938–3947, 2022, doi: https://doi.org/10.1016/j.procs.2022.09.456.

[8] Z. Khan, H. Liu, Y. Shen, and X. Zeng, "Deep learning improved YOLOv8 algorithm: Real-time precise instance segmentation of crown region orchard canopies in natural environment," *Comput. Electron. Agric.*, vol. 224, no. September, p. 109168, 2024, doi: https://doi.org/10.1016/j.compag.2024.109168.

[9] B. Lin, "Safety Helmet Detection Based on Improved YOLOv8," *IEEE Access*, vol. 12, no. February, pp. 28260–28272, 2024, doi: 10.1109/ACCESS.2024.3368161.

[10] M. Safran, A. Alajmi, and S. Alfarhood, "Efficient Multistage License Plate Detection and Recognition Using YOLOv8 and CNN for Smart Parking Systems," *J. Sensors*, vol. 2024, no. 1, p. 4917097, 2024, doi: https://doi.org/10.1155/2024/4917097.

[11] J. Farooq, M. Muaz, K. Khan Jadoon, N. Aafaq, and M. K. A. Khan, "An improved YOLOv8 for foreign object debris detection with optimized architecture for small objects," *Multimed. Tools Appl.*, vol. 83, no. 21, pp. 60921–60947, 2024, doi: 10.1007/s11042-023-17838-w.

[12] R. Sapkota, D. Ahmed, and M. Karkee, "Comparing YOLOv8 and Mask R-CNN for instance segmentation in complex orchard environments," *Artif. Intell. Agric.*, vol. 13,

no. September, pp. 84–99, 2024, doi: https://doi.org/10.1016/j.aiia.2024.07.001.
[13] G. Yang, J. Wang, Z. Nie, H. Yang, and S. Yu, "A Lightweight YOLOv8 Tomato Detection Algorithm Combining Feature Enhancement and Attention," *Agronomy*, vol. 13, no. 7, pp. 1–14, 2023, doi: 10.3390/agronomy13071824.
[14] M. Talib, A. H. Y. Al-Noori, and J. Suad, "YOLOv8-CAB: Improved YOLOv8 for Real-time Object Detection," *Karbala Int. J. Mod. Sci.*, vol. 10, no. 1, pp. 56–68, 2024, doi: 10.33640/2405-609X.3339.
[15] G. Oh and S. Lim, "One-Stage Brake Light Status Detection Based on YOLOv8," *Sensors*, vol. 23, no. 17, pp. 1–18, 2023, doi: 10.3390/s23177436.
[16] Y. Irawan, "Decision Support System For Employee Bonus Determination With Web-Based Simple Additive Weighting (SAW) Method In PT. Mayatama Solusindo," *J. Appl. Eng. Technol. Sci.*, vol. 2, no. 1, pp. 7–13, 2020, doi: https://doi.org/10.37385/jaets.v2i1.162.
[17] M. Muthumari, V. Akash, K. P. Charan, P. Akhil, V. Deepak, and S. P. Praveen, "Smart and Multi-Way Attendance Tracking System Using an Image-Processing Technique," in *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 2022, pp. 1805–1812. doi: 10.1109/ICSSIT53264.2022.9716349.
[18] R. Shi, T. Li, and Y. Yamaguchi, "An attribution-based pruning method for real-time mango detection with YOLO network," *Comput. Electron. Agric.*, vol. 169, no. February, p. 105214, 2020, doi: https://doi.org/10.1016/j.compag.2020.105214.
[19] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv*, vol. 2004, no. Apr, pp. 1–17, 2020, doi: https://doi.org/10.48550/arXiv.2004.10934.
[20] X. Zhang *et al.*, "Inspection and Classification System of Photovoltaic Module Defects Based on UAV and Thermal Imaging," in *2022 7th International Conference on Power and Renewable Energy (ICPRE)*, 2022, pp. 905–909. doi: 10.1109/ICPRE55555.2022.9960506.
[21] H. T. Ngoc, K. H. Nguyen, H. K. Hua, H. V. N. Nguyen, and L. Da Quach, "Optimizing YOLO Performance for Traffic Light Detection and End-to-End Steering Control for Autonomous Vehicles in Gazebo-ROS2," *Int. J. Adv. Comput. Sci. Appl.*, vol. 14, no. 7, pp. 475–484, 2023, doi: 10.14569/IJACSA.2023.0140752.
[22] T. H. Wu, T. W. Wang, and Y. Q. Liu, "Real-Time Vehicle and Distance Detection Based on Improved Yolo v5 Network," *2021 3rd World Symp. Artif. Intell. WSAI 2021*, vol. 978, no. June, pp. 24–28, 2021, doi: 10.1109/WSAI51899.2021.9486316.
[23] Y. Wang, H. Wang, and Z. Xin, "Efficient Detection Model of Steel Strip Surface Defects Based on YOLO-V7," *IEEE Access*, vol. 10, no. November, pp. 133936–133944, 2022, doi: 10.1109/ACCESS.2022.3230894.
[24] A. Syahputra, Yaddarabullah, M. F. Azhary, A. B. A. Rahman, and A. Saad, "Occupancy Measurement in Under-Actuated Zones: YOLO-based Deep Learning Approach," *Int. J. Adv. Comput. Sci. Appl.*, vol. 15, no. 2, pp. 757–769, 2024, doi: 10.14569/IJACSA.2024.0150277.
[25] A. Febriani, R. Wahyuni, Y. Irawan, and R. Melyanti, "Improved Hybrid Machine and Deep Learning Model for Optimization of Smart Egg Incubator," *J. Appl. Data Sci.*, vol. 5, no. 3, pp. 1052–1068, 2024.