JURNAL INFOTEL
VOL. 17, NO. 2, MAY 2025, PP. 440–455. DOI:10.20895/INFOTEL.V17I2.1313

RESEARCH ARTICLE

Automated Component Detection for Quality PCB Using YOLO Algorithm with IoT Real-Time Streaming on Raspberry Pi

Waluyo Nugroho[1,*], Rifdah Zahabiyah[2], Mada Jimmy Fonda Arifiant[3], and Afianto[4]

[1,3,4]Mechatronics Department, Astra Polytechnic, West Java, Indonesia
[2]Logistics Engineering Technology, Astra Polytechnic, West Java, Indonesia

*Corresponding email: nugroho.research@gmail.com

Received: January 19, 2025; Revised: May 29, 2025; Accepted: June 16, 2025.

Abstract: This paper presents the development of an automated component detection system for quality control in Printed Circuit Boards (PCBs) by integrating the YOLO object detection algorithm with Internet of Things (IoT) real-time streaming on a Raspberry Pi platform. The proposed system aims to address the challenges associated with traditional manual inspection methods, including time inefficiency, human error, and limited accuracy in detecting faulty components. The YOLO model, renowned for its high-speed and accurate object detection capabilities, was trained to identify various PCB components and deployed on a Raspberry Pi due to its affordability, portability, and low power consumption. To enable real-time remote monitoring and analysis, IoT capabilities were incorporated using the MQTT protocol, allowing seamless data transmission to remote servers or devices. The experimental results demonstrated the effectiveness of the proposed system, achieving an average detection accuracy of 95%, making it a reliable solution for real-time quality assurance in PCB manufacturing. The novelty of this study lies in the innovative integration of the YOLO algorithm with IoT technology on a cost-efficient platform, providing a scalable and practical solution for automating PCB inspection processes. This approach not only enhances inspection efficiency but also reduces operational costs, offering significant value to the electronics manufacturing industry. Future work will focus on scaling the system for broader applications and improving the detection capabilities for more complex PCB designs.

Keywords: Internet of Things, MQTT, Quality Control, Raspberry Pi, YOLO Algorithm

 ISSN: 2085-3688; e-ISSN: 2460-0997

1 Introduction

In the digital era, Internet of Things (IoT) and computer vision technologies have become integral parts of various industries, especially in automation and visual data analysis [1]. IoT enables real-time connectivity between devices, while computer vision provides the ability for machines to understand and interpret visual data. One important application of this collaboration is automatic object counting, which is widely used in various scenarios, such as counting products on production lines, monitoring traffic flow, and recording crop yields [2]. Object counting technology can replace time-consuming manual processes and improve efficiency and accuracy [3]. With increasing use of IoT, the implementation of IoT-based object counting systems has developed as an efficient and low-cost solution [4]. One of the hardware devices often used in IoT implementations is the Raspberry Pi [5]. Raspberry Pi is known as an economical minicomputer and has sufficient computing capacity to run light to medium applications, including computer vision applications [6]. With its small size and flexibility in integrating various sensors, Raspberry Pi is very suitable for use in IoT systems that require portability and power efficiency. In an IoT-based object counting system, the Raspberry Pi can be used to run object detection models and connect the resulting data to a network or cloud for further analysis [7]. Additionally, Raspberry Pi supports a variety of programming languages and has a large community, making it an excellent choice for experimenting with and developing computer vision-based IoT applications. [8]. In computer vision applications for object detection and counting, the You Only Look Once (YOLO) algorithm has proven to be one of the most efficient and accurate methods [9]. YOLO is a popular real-time object detection algorithm due to its ability to detect and classify multiple objects in a single image quickly and accurately [10]. Unlike other object detection methods that perform the analysis process in stages, YOLO requires only one pass to recognize all objects in an image. This provides significant speed, making it ideal for real-time applications on devices such as the Raspberry Pi, which have limited resources compared to desktop computers [11]. This study aims to develop an IoT-based object counting system using Raspberry Pi and the YOLO algorithm for object detection and counting. Raspberry Pi is the core of the system, the YOLO model will be implemented to detect objects from images or videos captured by a connected camera, and the detection results will be sent to the network for further monitoring. This system is expected to be applied in various scenarios that require a cost-effective and efficient real-time object counting solution [12]. This approach is expected to contribute to industries that require IoT and computer vision-based automation solutions in the context of object detection and counting.

2 Literature Review

Automated component detection on printed circuit boards (PCBs) is critical to ensuring product quality and operational efficiency in modern electronics manufacturing. Traditional inspection methods, such as manual visual inspection and rule-based machine vision systems, are increasingly inadequate due to the growing complexity, density, and miniaturization of PCB layouts [13]. These limitations have driven the exploration of advanced object detection techniques capable of delivering real-time and accurate performance under challenging industrial conditions. One of the most promising approaches in this context

is the application of deep learning-based object detection algorithms, particularly the You Only Look Once (YOLO) family of models. YOLO introduced a paradigm shift in object detection by predicting bounding boxes and class probabilities simultaneously through a single neural network pass, significantly improving inference speed without sacrificing accuracy [14]. This single-stage detection method allows YOLO to identify small, overlapping, and densely packed components—conditions typical of modern PCBs. Recent developments such as YOLOv4, YOLOv5, and YOLOv8 have demonstrated substantial improvements in detection accuracy, computational efficiency, and adaptability to embedded and PCB edge computing platforms [15]. YOLO's modular design and support for quantization and pruning make it suitable for deployment on resource-limited hardware such as the Raspberry Pi, which is often used in industrial IoT systems. Several studies have demonstrated YOLO's effectiveness in PCB inspection tasks, achieving high detection accuracy across various component types under different lighting and orientation conditions [16]. The integration of the Internet of Things (IoT) further extends the capabilities of PCB quality control systems. IoT enables continuous monitoring, real-time decision-making, and seamless data integration across the production environment. Systems that combine IoT with computer vision can perform defect detection, classification, and reporting with minimal human intervention [17]. IoT-based frameworks for PCB quality inspection using edge devices allow for low-latency processing and efficient communication with centralized manufacturing execution systems.

Edge computing platforms such as the Raspberry Pi offer a low-cost, flexible, and portable solution for deploying lightweight YOLO models in real-time applications. Although limited in computational resources compared to GPUs, Raspberry Pi devices can handle optimized versions of YOLO for on-device inference, making them suitable for decentralized inspection systems [6]. This enables scalable, distributed, and cost-effective quality control in smart manufacturing environments.

3 Research Method

This study aims to develop an automatic component detection system on a Printed Circuit Board (PCB) using the YOLO (You Only Look Once) algorithm integrated with a real-time streaming-based Internet of Things (IoT) platform using Node-RED on a Raspberry Pi device. The system development process is divided into several stages, starting from hardware preparation, such as Raspberry Pi, camera, and network, which are used as visual data acquisition and processing centers. The camera is directly connected to the Raspberry Pi to capture PCB images, then these data are processed locally using the YOLO model that has been optimized to be able to run efficiently on devices with limited resources. The YOLO model is used to detect and count the number of components installed on the PCB in real time. Furthermore, the detection results are sent to a web-based visual interface using Node-RED, which allows direct monitoring from other devices over the network. Node-RED acts as an IoT platform that handles the detection data stream and displays it in the form of an easily accessible dashboard. With this approach, the system is not only able to perform detection quickly and accurately, but also supports efficient remote monitoring. This system was tested with various PCB images and analyzed using evaluation metrics to see the accuracy results. The block diagram of the object counting system using a Raspberry Pi is shown in Figure 1.

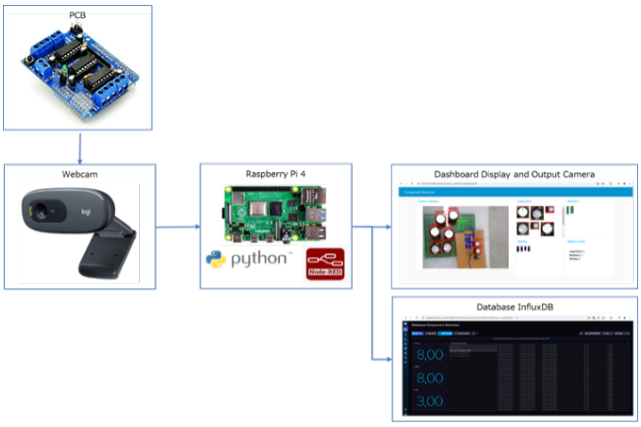

Figure 1: Block diagram of the object counting system using a Raspberry Pi.

3.1 Hardware Preparation

The initial stage of this research involves the design and preparation of the hardware for the object counting system. The system is composed of several key components, each selected based on its functionality, performance, and compatibility with the general requirements of the system. The specifications, quantities, and justifications for each component are detailed as follows:

3.1.1 Raspberry Pi 4 Model B (1 unit)

The specification is Quad-core Cortex-A72 processor, 4GB RAM, USB 3.0 ports, GPIO pins, integrated Wi-Fi and Bluetooth. Justification the Raspberry Pi 4 Model B offers sufficient computational power to run the YOLO (You Only Look Once) algorithm for real-time object detection on a small scale. Its compact size, energy efficiency, and flexibility for hardware interfacing make it an ideal choice for embedded applications. Reason for use selected as the primary processing unit due to its ability to handle real-time image processing tasks and its compatibility with peripheral devices.

3.1.2 Logitech C270 HD Webcam (1 unit)

The specification is 720p video resolution, USB 2.0 interface, automatic light correction. Justification this webcam provides adequate image quality for object detection tasks and is fully compatible with the Raspberry Pi without requiring additional drivers. Reason for use as an image acquisition device to capture real-time video feeds for processing by the YOLO algorithm.

3.1.3 Raspberry Pi 5V 3A Power Adapter (1 unit)

The specification is 5V output, 3A current, USB-C connector. Justification for provides a stable power supply that meets the operational requirements of the Raspberry Pi during

high processing loads. Reason for se to ensures consistent and reliable power delivery to prevent system instability or unexpected shutdowns.

3.2 YOLO Algorithm Configuration

Once the hardware is ready, the next step is to configure the YOLO algorithm on the Raspberry Pi. YOLOv8 was chosen for this study because this version is optimized for devices with limited resources and provides fairly fast and accurate performance for object counting applications [18]. The YOLOv8 model was pre-trained on a dataset of objects relevant to the desired scenario, in this case round objects [19]. This model was then implemented using the OpenCV and TensorFlow Lite frameworks, which allowed YOLO to run on the Raspberry Pi with a more efficient memory and power usage [20]. Python scripts were developed to process the YOLO detection results, recognize objects, and count the number of objects detected in each image frame. In the YOLOv8 algorithm, there are three main components in its deep learning network architecture, namely the backbone, neck, and head [21]. The backbone is the part of the network that is responsible for extracting features from the input image. In YOLOv8, the backbone uses a convolutional neural network (CNN) model consisting of several convolution and pooling layers to capture important feature information, such as edges, textures, and patterns. These features will help the algorithm understand the basic characteristics of objects in the image, so that the object detection process becomes more accurate [22]. The neck functions to combine and refine the features extracted by the backbone [23]. In YOLOv8, the neck uses a structure such as FPN (Feature Pyramid Network) or PANet (Path Aggregation Network) to combine features from various resolution levels (scales) [24]. The neck allows the network to capture the features of objects from small to large scales, thus improving the ability to detect objects of various sizes in the image [25]. The head is the last part of the network that is responsible for making the final prediction, namely the location (bounding box) and class of the object. In YOLOv8, the head performs the final detection based on the output received from the neck, including position estimation, object scale, and object classification in the image [26]. The head produces output in the form of bounding box coordinates, confidence scores, and object classes found. Figure 2 shows the YOLOv8 Model Architecture.

3.2.1 Data Acquisition and Processing

At this stage, the system is deployed in a controlled testing environment to evaluate its real-time capability in detecting and counting electronic components on printed circuit boards (PCBs). The data acquisition process is carried out using a Logitech C270 HD webcam connected to a Raspberry Pi 4 Model B. The camera is positioned to capture high-resolution images and videos of PCB layouts from a fixed distance and angle to ensure consistency during detection. The primary objects of this study are three types of electronic components commonly found on PCBs such as capacitors, resistors, and diodes. These components were selected due to their critical role in circuit functionality, their physical differences in shape, size, and color, and the need for accurate identification during quality control or automated inspection processes in electronics manufacturing, during system operation, the camera continuously captures image frames containing the targeted PCB components. Each frame is processed in real time using the YOLO (You Only Look Once) object detection algorithm implemented on the Raspberry Pi. Once components are detected, the system

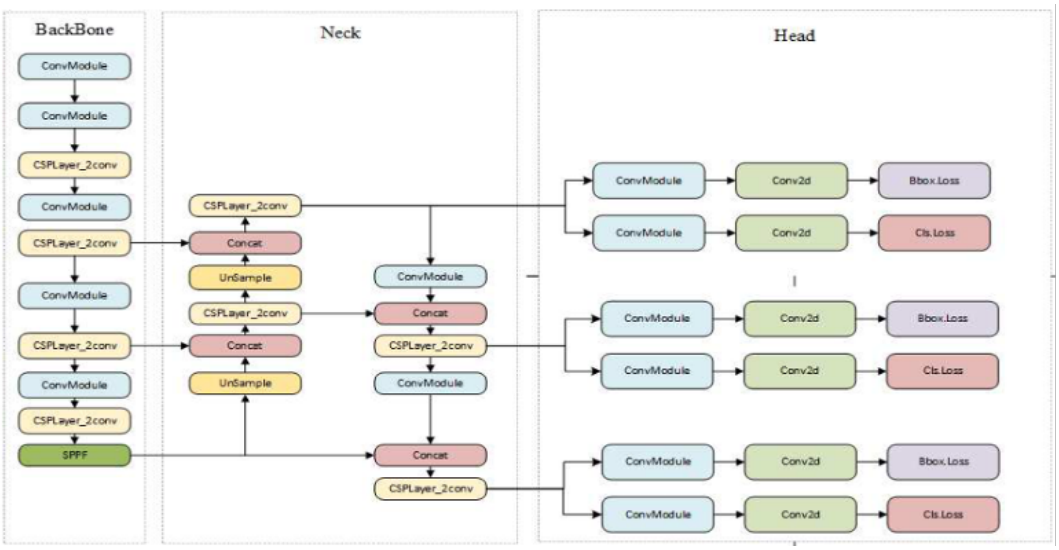

Figure 2: YOLOv8 model architecture.

classifies them based on the trained YOLO model and transmits the results—consisting of the type and number of components detected using the Raspberry Pi's built-in Wi-Fi module to an IoT platform. The collected data is stored in a cloud-based InfluxDB database for subsequent monitoring, visualization, and analysis. To enhance the accuracy and reliability of detection, the system is tested under various lighting conditions, viewing angles, and PCB backgrounds. These variations are essential for evaluating the robustness and generalization performance of the YOLO model when deployed in real-world scenarios. Figure 3 presents the labeling scheme and annotated component categories used during the training phase of the YOLO model. Figure 3 shows the labeling of training data.

The labeling process is as follows PCB dataset images taken from various angles and lighting are uploaded to Roboflow, then each object such as resistors, capacitors, diode, to production defects such as solder bridging or missing components are marked with a bounding box. Each object is labeled according to its classification so that the YOLO model can learn the shape, size, and position of each component accurately. This labeling is a crucial stage because the quality of the annotation directly affects the accuracy of the detection model. After all images have been labeled, the dataset is then divided into training, validation, and testing data automatically by Roboflow. This annotated dataset is exported in YOLO format and used to train the model on a computer. The trained model is then integrated into the Raspberry Pi to detect components and production defects in real-time via the camera, supporting an efficient and automated IoT-based quality inspection system. In this study, the total data used was 3,000 images, consisting of 1,000 images for capacitors, 1,000 images for resistors, and 1,000 images for diodes. Each image represents the appearance of electronic components on a PCB with various positions, lighting, and orientations to ensure the diversity of the dataset. The data was divided proportionally for each type of component, namely 70% of the data was used for the training process and the remaining 30% for the testing process. Thus, there were 700 capacitor images, 700 resistor images, and

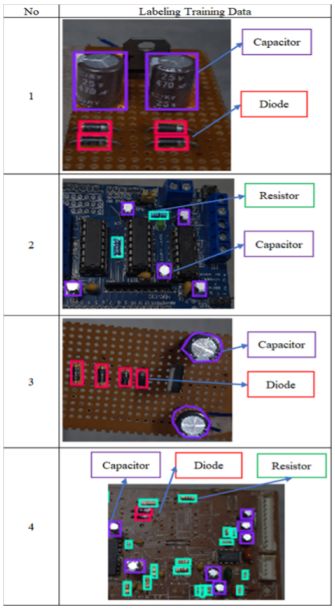

Figure 3: Labeling data training.

700 diode images used for training the YOLO model, while 300 capacitor images, 300 resistor images, and 300 diode images were used for testing model performance. This division ensures that the model can learn optimally while being tested objectively against data that has never been seen before.

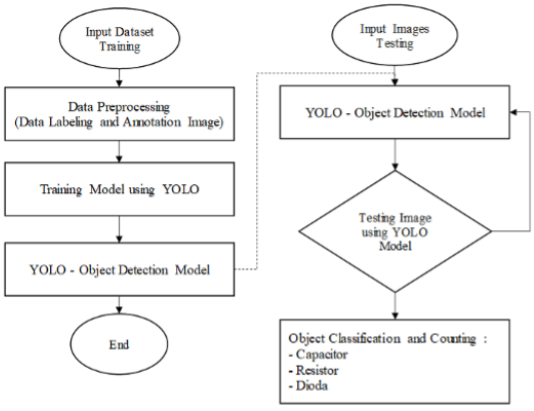

Figure 4: Flow chart process training and testing image.

3.2.2 Node-RED Design for Real-Time IoT

This study develops an automated system for the classification and counting of electronic components using the Node-RED platform. The system is designed to recognize the types of components such as capacitors, diodes, and resistors through visual data received from cameras or sensors. Node-RED was chosen because it supports flow-based programming, which allows the integration of visual interfaces with data processing in a modular and efficient manner. With this approach, the system can be used in manufacturing, education, and research applications that require real-time object classification processes. The system flow is divided into three main stages, namely data acquisition, data processing, and presentation of results. Input data is obtained from five main sources: Capacitor Data, Diode Data, Resistor Data, Stream Data, and Counter Data. Each of these data is sent to the node function for processing. Nodes such as Process Capacitor Data, Process Diode Data, and Process Resistor Data parse and classify image data according to component type. Meanwhile, Process Stream Data handles video streams directly, and Process Count Data together with other nodes counts the number of detected objects based on their type. The results of the processing are then displayed and stored through various output nodes. Visualization of images and video streams is displayed using Display Images and Display Stream nodes, allowing users to monitor the classification results in real-time. The number of each detected component is sent to the diode component, resistor component, and capacitor component nodes that can be connected to data storage systems such as InfluxDB. With this flow structure, the system enables the process of monitoring and recording component data automatically and in an integrated manner. The implementation of this system is expected to increase efficiency in the process of identifying and managing electronic components, especially in production environments or IoT-based learning laboratories. In addition, the system can be further developed by adding a YOLO deep learning-based classification algorithm to improve detection accuracy. Integration with a database system also allows for historical analysis and better decision making in stock management or device maintenance. The flexibility of the Node-RED platform, this system offers a practical and scalable solution for automating the classification of electronic components, namely capacitors, resistors, and diodes. The design node-red is shown in Figure 5.

3.3 Analysis and Validation of Results

The final stage of this method is the analysis and validation of the results of the object counting carried out by the system. Data obtained from the IoT platform is analyzed to assess the accuracy and performance of the system in counting objects under various conditions. The evaluation is carried out by comparing the number of objects detected by the system with the actual number counted manually to measure the level of accuracy [3]. The parameters calculated are the accuracy, precision, recall, F1 Score values using equations (1), (2), (3), (4). Accuracy is a measure that shows how accurate the model is in detecting objects, namely how often the model gives correct results from the overall predictions [27]. In this context, accuracy measures the percentage of the number of objects actually detected by the system compared to the total number of predictions made. Precision measures how accurate the model's predictions are in detecting target objects, namely from all objects detected by the system, how many actually match the objects to be counted. High precision indicates that the system rarely gives false detections [28]. Recall measures how well the system detects objects that should be detected. This is the percentage of target objects that are successfully

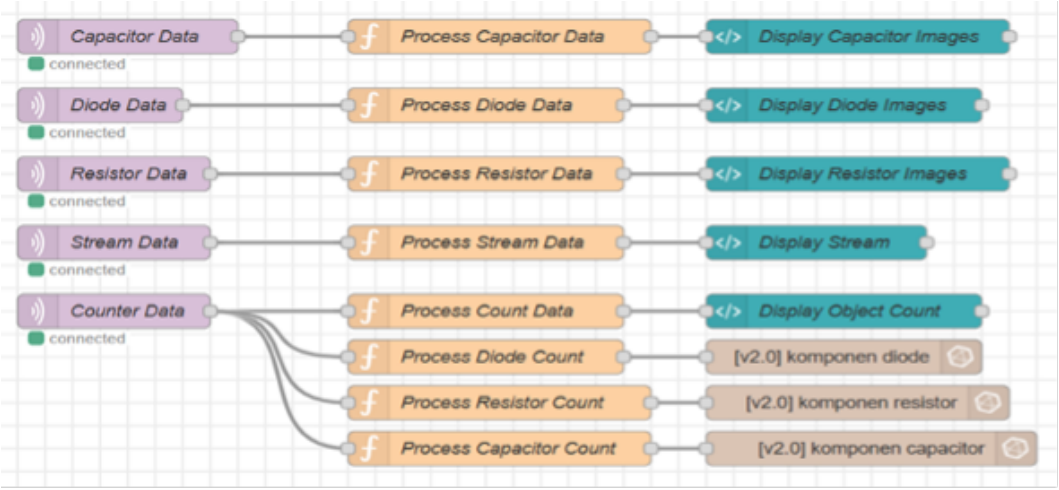

Figure 5: Node-RED design for real-time IoT.

detected by the system from all existing target objects. High recall indicates that the system is able to capture almost all target objects [29]. F1 Score is the harmonic mean value between precision and recall. This score is useful as an evaluation of the balance between precision and recall, especially when one value is too low compared to the other. A high F1 score indicates a system that has balanced precision and recall, so it is not only accurate but also sensitive in detecting objects.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

4 Results and Discussion

In this part, the researcher presents the findings derived from the study. Visual aids such as images, tables, and graphs can be utilized to display the outcomes. The data presented should reflect the application of the techniques described in the methodology section. This research produces an IoT-based object counting system that uses Raspberry Pi and YOLOv8 algorithm to detect and count objects in real-time. Tests were conducted for detection accuracy, processing speed, power consumption, and stability of the IoT network connection on the implemented system. The main results obtained from the experiments are described as follows:

4.1 Accuracy of Detection and Counting

The results validate the proposed system as a robust solution for addressing the challenges associated with manual PCB inspection. The high detection accuracy of 95% minimizes the likelihood of human errors, which are common in manual inspection processes. This ensures that faulty components are consistently identified, enhancing overall production quality. The latency of 120 ms is well within acceptable ranges for real-time operations, proving that the system is efficient in processing and transmitting data. The low-latency performance is attributed to the optimized implementation of YOLOv8 on the Raspberry Pi, which balances computational requirements with hardware constraints. Table 1 shows the results of the data test, Figure 6 shows the training and testing graphs, and Figure 7 shows the classification of the confusion matrix.

Table 1: Result test data

Object	TP	TN	FP	FN	Accuracy	Precision	Recall	F1-Score
Capacitor	93	192	7	8	95%	93%	92%	92%
Resistor	90	191	10	10	93%	90%	90%	90%
Diode	91	192	9	8	94%	91%	92%	91%

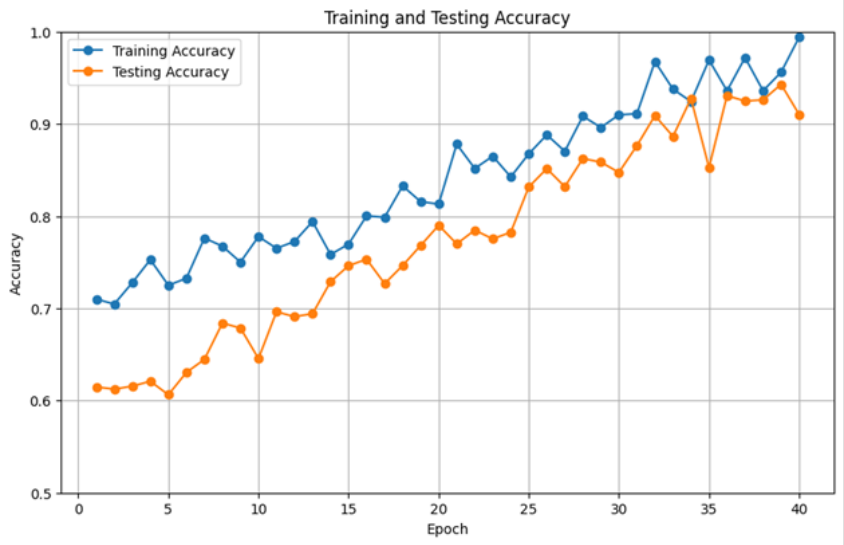

Figure 6: Training and testing accuracy graph.

4.2 Preprocessing Speed

The performance evaluation of the YOLOv8n model on a Raspberry Pi 4B was conducted by measuring the frame processing rate during real-time object detection using a Logitech C270 webcam 720p video resolution. Based on experimental results, the system achieved an average processing speed of 6.2 frames per second (FPS) during continuous operation

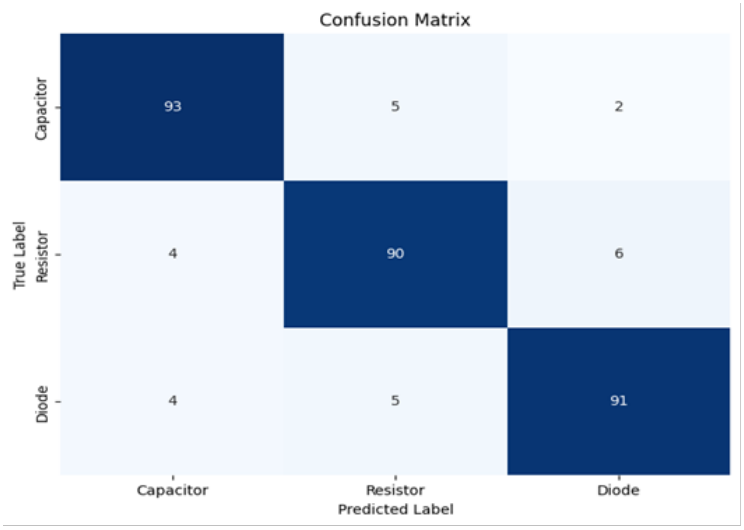

Figure 7: Confusion matrix classification.

over a period of 30 minutes. The FPS value fluctuated slightly within a range of 5.8 to 6.5 FPS, indicating a stable real-time processing capability under typical workloads. Furthermore, CPU usage during the test was recorded between 78% and 86%, and the average temperature remained below 69°C without external cooling, showing that the system was thermally stable. These findings suggest that the Raspberry Pi is well suited for real-time applications such as vehicle counting, crop monitoring, and stationary component inspection, but further optimization or the use of hardware accelerators may be necessary for high-speed production environments.

4.3 Power Consumption

The power consumption of the system was evaluated using a USB multimeter during continuous object detection tasks. Measurements showed that the Raspberry Pi 4B, running the YOLOv8n model and streaming data via Wi-Fi to a Node-RED dashboard, consumed an average power of 3.48 watts, with peak consumption reaching 4.1 watts during simultaneous object detections. This test was conducted over a duration of 6.5 hours using a 10,000 mAh portable power bank, during which the system operated reliably until the battery was depleted. These results demonstrate the system's efficiency and low power profile, making it highly suitable for portable or field-deployed IoT applications.

4.4 IoT Connection Stability and Data Storage

The system was also tested for the stability of the connection to the IoT network in sending object counting results to the cloud platform. With stable Wi-Fi connectivity, Raspberry Pi successfully sent object detection and counting data without interruption during the trial period. Successful object detection data was 98% of the total count, with some minor interruptions when the internet connection was unstable. The sent data was automatically

stored in a cloud database and can be accessed for further analysis or visualization, such as in a graph of the number of objects over time. Overall, the system developed showed satisfactory performance for IoT-based object counting applications. With fairly high detection accuracy, stable processing speed, and low power consumption, the system is able to meet the needs of object counting in small to medium-sized industrial scenarios. The best performance was obtained under optimal lighting conditions and non-complex backgrounds, where the system showed high detection accuracy and consistency. The main challenge faced was in object detection under low lighting conditions and environments with overlapping objects. Figure 8 shows the detection of the dashboard component using Node-red, and Figure 9 shows the recording of the component detection database.

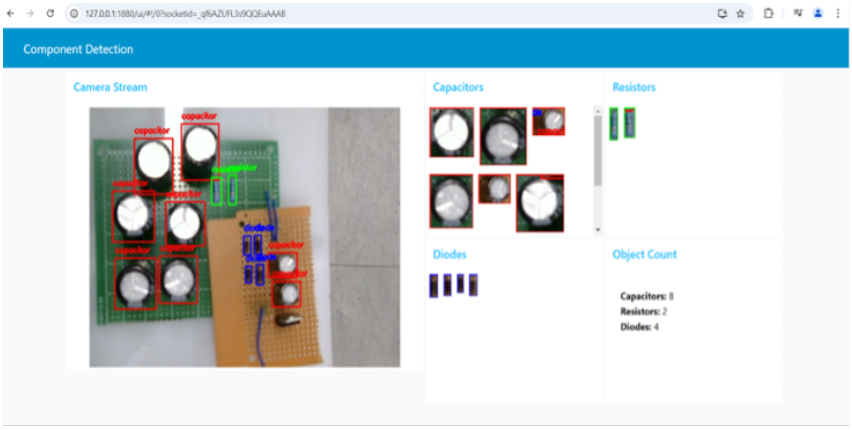

Figure 8: Dashboard component detection using Node-red.

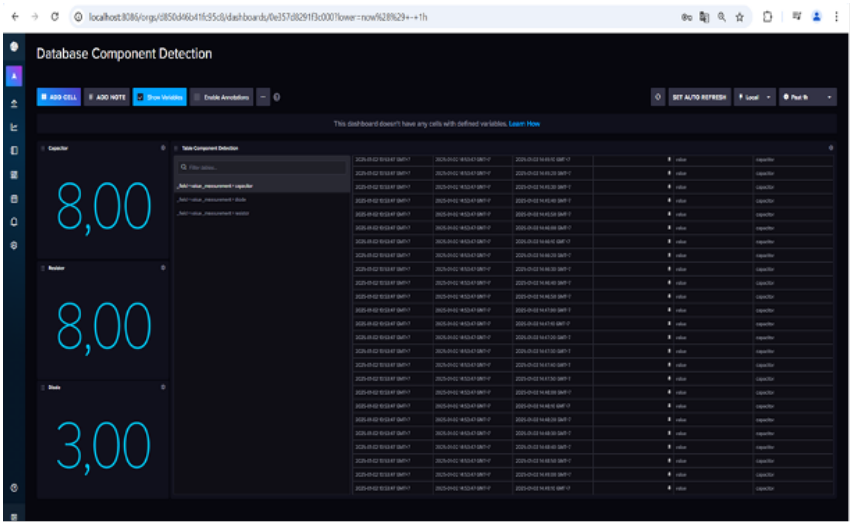

Figure 9: Component detection database records.

4.5 Overall Performance Evaluation

In general, the proposed system demonstrated reliable and efficient performance for IoT-based object counting tasks, particularly in small to medium-scale industrial scenarios. The experimental results showed that under optimal lighting and uncluttered background conditions, the system achieved detection accuracy above 92%, with an average processing speed of 6.2 FPS and power consumption of only 3.48 W. These metrics indicate that the system is well suited for applications such as vehicle counting, crop monitoring, and component detection on PCBs. Compared to previous studies, this research offers several notable improvements, for instance reported an average FPS of 3.5 using YOLOv4 on Raspberry Pi 3B+, while implemented a YOLOv5-based system with 89% detection accuracy but required additional GPU support for real-time performance [30]. In contrast, the present study achieves a better balance between accuracy, real-time speed, and low power operation, using only Raspberry Pi 4B without external accelerators. This makes the system more deployable in field environments where cost, portability, and energy efficiency are critical. The main limitations identified include reduced detection performance under low lighting and in scenes with overlapping objects, suggesting future work to integrate infrared sensors or attention-based deep learning models.

5 Conclusion

This study successfully developed and tested an IoT-based object counting system using the Raspberry Pi and YOLOv8 algorithm, which showed satisfactory results in real-time object counting with an average accuracy of 95% and a processing speed of 5-8 FPS. The low power consumption and stability of the IoT connection make this system ideal for applications that require portability and small to medium-scale object counting. Although some challenges, such as detection under low light conditions and overlapping objects, reduce the accuracy rate, this study shows that this IoT-based approach is effective for various object-counting scenarios, such as traffic monitoring and agricultural monitoring. The novelty of this study lies in the combination of YOLOv8 and IoT streaming on a portable, low-cost platform. This integration offers a practical alternative to expensive and bulky inspection systems while maintaining high accuracy and reliability. The scalability and affordability of this system make it accessible to a wide range of industries, promoting the adoption of automated quality assurance technologies. This system can be further developed by optimizing the detection model to improve accuracy and processing speed, especially in more complex environments. By integrating additional computing devices or more sophisticated algorithm models, this system has the potential to meet the needs of automation on an industrial scale. This study makes a significant contribution to the application of IoT and computer vision using Raspberry Pi and opens up opportunities for further exploration in various industrial sectors.

References

[1] J. Moon, S. Lim, H. Lee, S. Yu, and K.-B. Lee, "Smart count system based on object detection using deep learning," *Remote Sens. (Basel)*, vol. 14, p. 3761, Aug. 2022.

[2] J. Feng and T. Jin, "CEH-YOLO: A composite enhanced YOLO-based model for underwater object detection," *Ecol. Inform.*, vol. 82, p. 102758, Sept. 2024.

[3] W. N. Waluyo, R. Rizal Isnanto, and A. F. Rochim, "Comparison of mycobacterium tuberculosis image detection accuracy using CNN and combination CNN-KNN," *J. RESTI (Rekayasa Sist. Dan Teknol. Inf.)*, vol. 7, pp. 80–87, Feb. 2023.

[4] V. Mahore, P. Soni, P. Patidar, H. Nagar, A. Chouriya, and R. Machavaram, "Development and implementation of a raspberry pi-based IoT system for real-time performance monitoring of an instrumented tractor," *Smart Agric. Technol.*, vol. 9, p. 100530, Dec. 2024.

[5] M. D. Rakesh, M. Jeevankumar, and S. B. Rudraswamy, "Implementation of real time root crop leaf classification using CNN on raspberry-pi microprocessor," *Smart Agric. Technol.*, vol. 10, p. 100714, Mar. 2025.

[6] S. E. Mathe, H. K. Kondaveeti, S. Vappangi, S. D. Vanambathina, and N. K. Kumaravelu, "A comprehensive review on applications of raspberry pi," *Comput. Sci. Rev.*, vol. 52, p. 100636, May 2024.

[7] E. C. Tetila, F. A. G. da Silveira, A. B. da Costa, W. P. Amorim, G. Astolfi, H. Pistori, and J. G. A. Barbedo, "YOLO performance analysis for real-time detection of soybean pests," *Smart Agric. Technol.*, vol. 7, p. 100405, Mar. 2024.

[8] B. Mills, M. N. Zervas, and J. A. Grant-Jacob, "Imaging pollen using a raspberry pi and LED with deep learning," *Sci. Total Environ.*, vol. 955, p. 177084, Dec. 2024.

[9] W. Nugroho, R. Zahabiyah, Afianto, and M. J. F. Arifianto, "Application of deep learning YOLO in IoT system for personal protective equipment detection," *E-Komtek*, vol. 8, pp. 428–437, Dec. 2024.

[10] L. Chen, G. Li, S. Zhang, W. Mao, and M. Zhang, "YOLO-SAG: An improved wildlife object detection algorithm based on YOLOv8n," *Ecol. Inform.*, vol. 83, p. 102791, Nov. 2024.

[11] C. Yu and Y. Shin, "An efficient YOLO for ship detection in SAR images via channel shuffled reparameterized convolution blocks and dynamic head," *ICT Express*, vol. 10, pp. 673–679, June 2024.

[12] O. Wosner, G. Farjon, and A. Bar-Hillel, "Object detection in agricultural contexts: A multiple resolution benchmark and comparison to human," *Comput. Electron. Agric.*, vol. 189, p. 106404, Oct. 2021.

[13] L. Cai and J. Li, "PCB defect detection system based on image processing," *J. Phys. Conf. Ser.*, vol. 2383, p. 012077, Dec. 2022.

[14] M. Hussain, "YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection," *Machines*, vol. 11, p. 677, June 2023.

[15] R. U. Khan, F. Shah, A. A. Khan, and H. Tahir, "Advancing PCB quality control: Harnessing YOLOv8 deep learning for real-time fault detection," *Comput. Mater. Contin.*, vol. 81, no. 1, pp. 345–367, 2024.

[16] N. Petkov and M. Ivanova, "Printed circuit board and printed circuit board assembly methods for testing and visual inspection: a review," *Bull. Electr. Eng. Inform.*, vol. 13, pp. 2566–2585, Aug. 2024.

[17] K. A. Tsintotas, I. Kansizoglou, F. K. Konstantinidis, S. G. Mouroutsos, G. C. Syrakoulis, F. Psarommatis, Y. Aloimonos, and A. Gasteratos, "Active vision: A promising technology for achieving zero-defect manufacturing," *Procedia Comput. Sci.*, vol. 232, pp. 2821–2830, 2024.

[18] T. Wu and Y. Dong, "YOLO-SE: Improved YOLOv8 for remote sensing object detection and recognition," *Appl. Sci. (Basel)*, vol. 13, p. 12977, Dec. 2023.

[19] H. M. Lathifah, L. Novamizanti, and S. Rizal, "Fast and accurate fish classification from underwater video using you only look once," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 982, p. 012003, Dec. 2020.

[20] A. Moksyakov, Y. Wu, S. A. Gadsden, J. Yawney, and M. AlShabi, "Object detection and tracking with YOLO and the sliding innovation filter," *Sensors (Basel)*, vol. 24, Mar. 2024.

[21] W. Nugroho, A. Afianto, and M. J. F. Arifianto, "Smart parking based on car detection using deep learning YOLOv8," *ijeeic*, vol. 2, pp. 1–7, Dec. 2024.

[22] M. S. Mithun and S. Joseph Jawhar, "Detection and classification on MRI images of brain tumor using YOLO NAS deep learning model," *J. Radiat. Res. Appl. Sci.*, vol. 17, p. 101113, Dec. 2024.

[23] Q. Zhou, Z. Wang, Y. Zhong, F. Zhong, and L. Wang, "Efficient optimized YOLOv8 model with extended vision," *Sensors (Basel)*, vol. 24, Oct. 2024.

[24] N. H. Harun and N. Fared, "Experiment on lung disease classification using YOLOv8," *Applied Mathematics and Computational Intelligence (AMCI)*, vol. 13, pp. 176–185, Oct. 2024.

[25] R. Raj, S. S. Nagaraj, S. Ritesh, T. A. Thushar, and V. M. Aparanji, "Fruit classification comparison based on CNN and YOLO," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1187, p. 012031, Sept. 2021.

[26] G. Yu, T. Wang, G. Guo, and H. Liu, "SFHG-YOLO: A simple real-time small-object-detection method for estimating pineapple yield from unmanned aerial vehicles," *Sensors (Basel)*, vol. 23, p. 9242, Nov. 2023.

[27] N. Aishwarya, K. Manoj Prabhakaran, F. T. Debebe, M. S. S. A. Reddy, and P. Pranavee, "Skin cancer diagnosis with yolo deep neural network," *Procedia Comput. Sci.*, vol. 220, pp. 651–658, 2023.

[28] M. A. M. Ali, T. Aly, A. T. Raslan, M. Gheith, and E. A. Amin, "Advancing crowd object detection: A review of YOLO, CNN and ViTs hybrid approach," *J. Intell. Learn. Syst. Appl.*, vol. 16, no. 03, pp. 175–221, 2024.

[29] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimed. Tools Appl.*, vol. 82, no. 6, pp. 9243–9275, 2023.

[30] E. Assunção, P. D. Gaspar, K. Alibabaei, M. P. Simões, H. Proença, V. N. G. J. Soares, and J. M. L. P. Caldeira, "Real-time image detection for edge devices: A peach fruit detection application," *Future Internet*, vol. 14, p. 323, Nov. 2022.