ISSN 3025-3780(online)
ISSN 3025-4590(printed)

Journal of

Robotics, Automation, and Electronics Engineering

Vol. 4, No. 1, March 2026

JRAEE EDITORS

ISSN 3025-3780(online)
ISSN 3025-4590(printed)

Journal of

Robotics, Automation, and Electronics Engineering

Vol. 4, No. 1, March 2026

JRAEE REVIEWERS

ISSN 3025-3780(online)
ISSN 3025-4590(printed)

ISSN 3025-3780(online)
ISSN 3025-4590(printed)

Journal of

Robotics, Automation, and Electronics Engineering

Vol. 4, No. 1, March 2026

TABLE OF CONTENTENS

JRAEE

Journal of Robotics, Automation, and Electronics Engineering
Vol. 4, No. 1, 2026, pp. 226-236
e-ISSN 3025-3780
https://journal.uny.ac.id/v3/jraee

A Real-Time Road Damage Monitoring System using YOLOv11 on an Edge Computer

Zaki Nugraha[a,1,*], Aris Nasuha[b,2]

[a,b]Dept. of Electrical and Electronics Engineering, Vocational Faculty, Universitas Negeri Yogyakarta
[1]zakinugraha.2021@student.uny.ac.id; [2]arisnasuha@uny.ac.id
[*]Corresponding Author

ARTICLE INFO

Article History
Received Date Month Year
Revised Date Month Year
Accepted Date Month Year

ABSTRACT

This study developed a real-time road damage detection system on an edge device to address the inefficiency and subjectivity of manual monitoring. Following the Engineering Design Process (EDP), the system uses a YOLOv11 model—trained in PyTorch and converted to ONNX for optimization—to classify four types of road damage on a Raspberry Pi 5, integrated with a GPS module for automatic coordinate recording. Field evaluation showed that after optimization, the system achieved an average processing speed of 3.66 FPS with an inference latency of 277.69 ms, while maintaining efficient resource usage (45.06% RAM, 65.05% CPU load) and an mAP@50 of 0.554. These results demonstrate the feasibility of an autonomous, accurate, real-time road damage detection system on a low-cost edge computing platform, producing geospatial data ready to support operational decision-making in road maintenance.

Keywords
YOLOv11;
GPS;
Computer Vision;
ONNX;
Edge Computing.

ABSTRAK

Penelitian ini mengembangkan sistem deteksi kerusakan jalan secara real-time pada perangkat edge untuk mengatasi inefisiensi dan subjektivitas pemantauan manual. Mengikuti Engineering Design Process (EDP), sistem menggunakan model YOLOv11—dilatih dengan PyTorch dan dikonversi ke ONNX untuk optimasi—guna mengklasifikasikan empat jenis kerusakan jalan pada Raspberry Pi 5, yang diintegrasikan dengan modul GPS untuk pencatatan koordinat otomatis. Evaluasi lapangan menunjukkan bahwa setelah optimasi, sistem mencapai kecepatan pemrosesan rata-rata 3,66 FPS dengan latensi inferensi 277,69 md, sambil mempertahankan penggunaan sumber daya yang efisien (RAM 45,06%, beban CPU 65,05%) dan mAP@50 sebesar 0,554. Hasil ini membuktikan kelayakan sistem deteksi kerusakan jalan yang otonom, akurat, dan real-time pada platform edge computing berbiaya rendah, yang menghasilkan data geospasial siap pakai untuk mendukung pengambilan keputusan operasional dalam pemeliharaan jalan.

1. Introduction

Road infrastructure is a vital asset supporting mobility and the economy, yet its condition is often degraded

by damage such as cracks and potholes, which impact safety and comfort. Manual road condition monitoring has fundamental weaknesses: it is slow, expensive, and highly subjective. Although modern systems like the City Road Management System (CRMS) have been introduced in Indonesia, the core process of identifying specific damage types still heavily relies on manual visual interpretation by operators [1]. This dependence on the "human eye" perpetuates issues of scalability and objectivity, necessitating a faster, automated, and more measurable monitoring solution.

As a solution, survey automation using computer vision and deep learning has grown rapidly. Various studies have explored deep learning architecture for this task. For example, Hsieh et al. (2024) integrated YOLOv7 with IMU data on an NVIDIA Jetson Nano, but this system relies on 5G connectivity and cloud processing [2]. Mutijarsa et al. (2023) designed an IoT-based system, but the damage detection process was performed as post-processing, not real-time in the field [3]. Other studies focus on model comparison, such as Sidqi et al. (2024), who tested various YOLO variants and found YOLOv8s provided the best precision for four damage classes [4], while Wu et al. (2023) designed a lightweight model (YOLO-LWNet) for mobile devices [5]. Furthermore, Zanevych et al. (2024) achieved a high mAP with YOLOv11, but their focus was limited to only detecting potholes [6]. This state-of-the-art review reveals a gap: a lack of fully autonomous, low-cost systems capable of real-time inference directly on an edge device without cloud dependency, while simultaneously integrating GPS data.

To address this gap, this study proposes the design and implementation of an autonomous road damage detection system using the YOLOv11 model [7], optimized for a low-cost edge computing device, the Raspberry Pi 5. The contribution of this research is the development and validation of a platform capable of real-time detection of four types of damage (longitudinal cracks/D00, transverse cracks/D10, alligator cracks/D20, and potholes/D40) while simultaneously and automatically recording geospatial coordinates (GPS) [8], operating fully on the edge device without requiring cloud connectivity.

2. Method

2.1 Research Methodology

This research adopted the Engineering Design Process (EDP) as its core methodology, encompassing a structured cycle of Ask, Research, Imagine, Plan, Create, Test, and Improve. This systematic approach guided the development from the initial requirements definition and literature review to the final validated prototype, allowing for iterative optimization based on empirical test results. The cycle of this process, which guided the project from planning to improvement, is illustrated in Fig. 1.

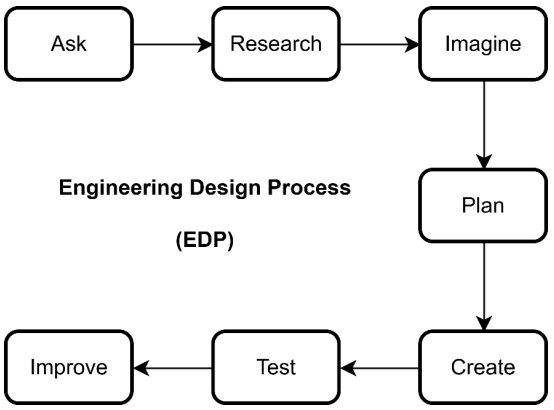


Fig.1 Engineering Design Process (EDP)

2.2 System Architecture and Hardware Design

The system's high-level architecture is designed around three core modules: Input, Processing, and Output, as illustrated in the block diagram in Fig. 2. The input module consists of a GoPro Hero 10 camera for video stream acquisition and a u-blox NEO-6MV2 GPS module for location coordinate acquisition. The Processing module is the system's core, utilizing a Raspberry Pi 5 (8GB) [10] to run the YOLOv11 model, perform object tracking, and synchronize data. The Output module provides real-time feedback to a portable monitor via the GUI and saves all resulting data (video, logs) to a MicroSD card.

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

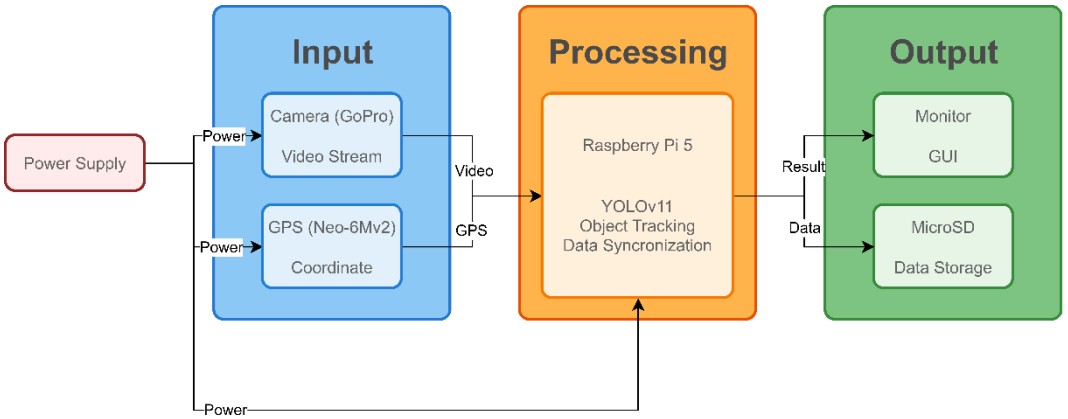


Fig.2 Block diagram system for architecture design

The detailed electronic hardware design, shown in Fig. 3, details the physical interconnection of these components. The entire system is powered from a vehicle's 12V source via a Taffware Car Power Inverter, which provides a 220V AC output. This powers a 5.1V/5A USB-C Power Supply for the Raspberry Pi 5. The Raspberry Pi 5 connects to the GoPro camera (UVC) and the u-blox GPS module via its USB ports for data transfer. Video output for the GUI is sent to the portable monitor via a micro-HDMI-to-mini-HDMI connection, while an active fan connected to the Pi's JST 4-pin header ensures thermal management.

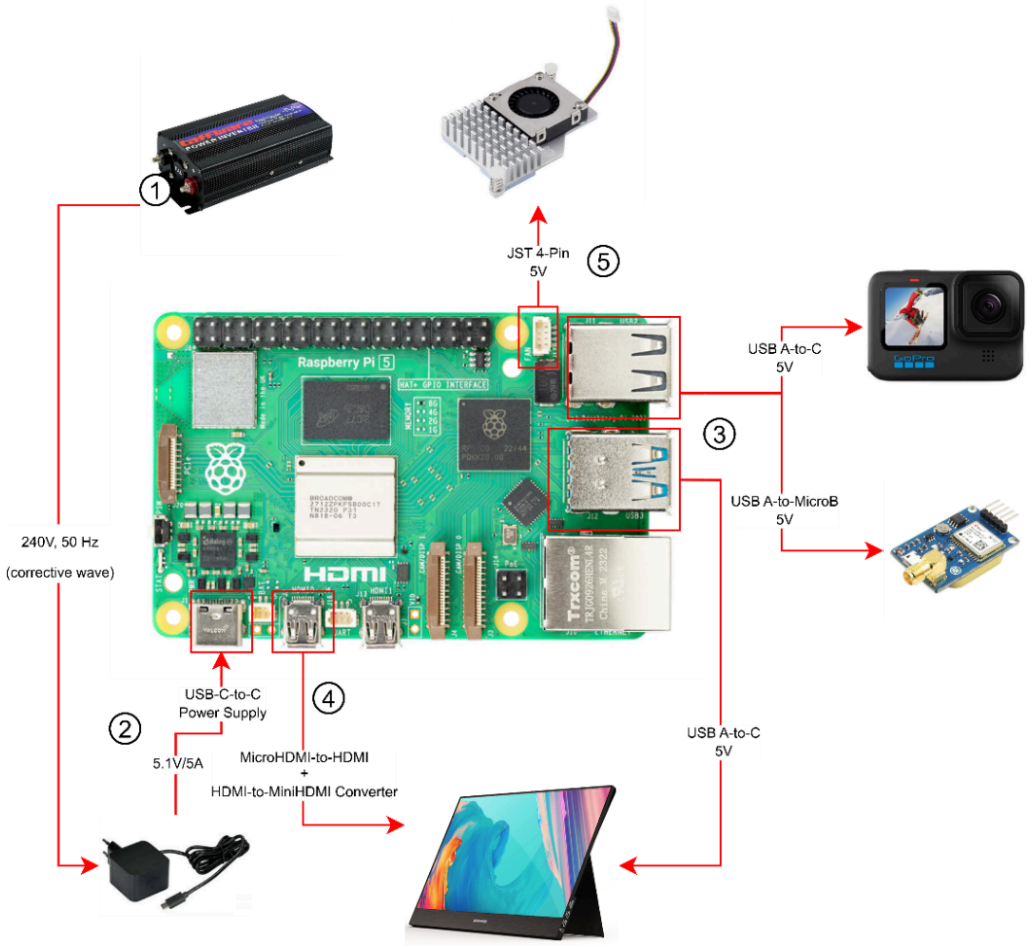


Fig.3 Hardware design

2.3 Software Design and Data Flow

The system's software, developed in Python 3.12, operates on a multi-threaded architecture to ensure real-time data synchronization. The main process initializes a custom-built GUI using `customtkinter`. Upon

starting, two parallel threads are launched:

- Inference Thread: This thread captures frames from the video stream, processes them using the YOLOv11 model (via the Ultralytics library and OpenCV), and annotates the video.
- GPS Thread: This thread continuously reads and parses NMEA sentences from the GPS module using `pynmea2` and `serial` libraries.

The main loop synchronizes the outputs from these threads, annotating video frames with both detection results (bounding boxes, class labels) and the most recent GPS coordinates (latitude, longitude, timestamp). All synchronized data is saved in structured formats (MP4 for video, JavaScript Object Notation (JSON) for detection logs, and Keyhole Markup Language (KML) / GPS Exchange Format (GPX) for geospatial tracks) for post-survey analysis. The complete software flowchart, detailing the multi-threaded logic and data flow, is shown in Fig. 4.

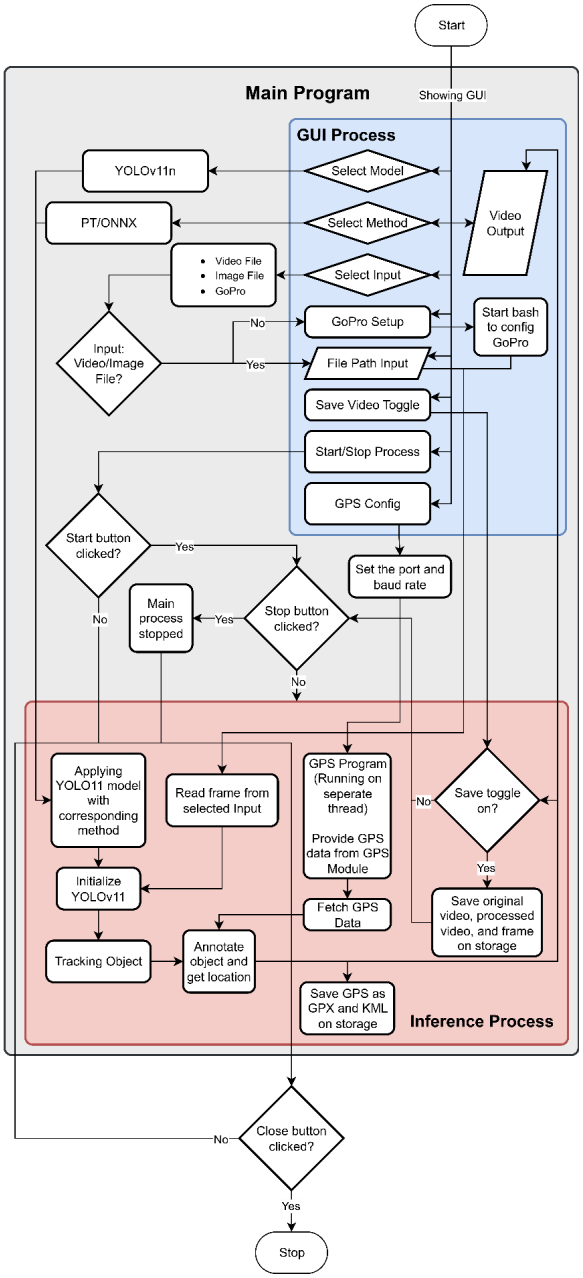


Fig.4 Software flowchart

2.4 Dataset and Model Training

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

The detection model was trained on a combined dataset sourced from the public CRDDC 2022 [9] and the 2023 Bina Marga Road Day Competition, ensuring diverse data from various countries, including Indonesia. The dataset was manually cleaned, standardized, and annotated to cover four distinct damage classes: longitudinal cracks (D00), transverse cracks (D10), alligator cracks (D20), and potholes (D40). The final dataset was split into training (70%), validation (20%), and testing (10%) sets. The YOLOv11-nano (YOLOv11n) model was trained using PyTorch on an NVIDIA RTX 3060, with a batch size of 16, targeting 1000 epochs. An early stopping mechanism with a patience of 100 epochs was implemented to prevent overfitting. The distribution of the combined dataset is shown in **Fig. 6**.

2.5 Model Optimization for Edge Computing

To achieve real-time performance on the resource-constrained Raspberry Pi 5, the trained PyTorch (.pt) model was converted to the ONNX (Open Neural Network Exchange) format. This conversion enables the use of the highly optimized ONNX Runtime, which performs static graph optimizations (e.g., layer fusion) and utilizes compute kernels optimized for ARM Central Processing Unit (CPU) cores. This step was critical for reducing inference latency and stabilizing the frame rate, aiming to move from an unstable baseline to a reliable real-time system.

2.6 System Validation and Performance Metrics

The fully integrated system was validated through field tests in a vehicle. Three distinct routes were selected to represent different operational environments: semi-rural, rural/village, and dense urban. During these tests, the system's performance was evaluated based on a set of key quantitative metrics:

- Accuracy: Mean Average Precision (mAP), the primary metric for object detection, which calculates the average Precision-Recall curve. It is defined as:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \quad (1)$$

  where AP_i is the Average Precision for class i and N is the total number of classes. We primarily use mAP@50 (Intersection over Union (IoU) threshold of 0.5).
- Speed: Frames Per Second (FPS) and inference latency (ms).
- Efficiency: CPU load (%), Random Access Memory (RAM) consumption (%), and CPU temperature (°C).

System metrics were logged using the `psutil` library and custom scripts, while inference speed was captured directly from the Ultralytics model's profiler.

3. Result and Discussion

The system's performance was evaluated in two key areas: model accuracy and operational system performance (speed, efficiency, and stability).

3.1 Model Accuracy and Training

The YOLOv11n model was trained as described in the methodology. The training process successfully converged, automatically halting at 289 epochs due to the early stopping mechanism, which prevented overfitting. The training and validation loss curves (**Fig. 5**) demonstrate stable convergence.

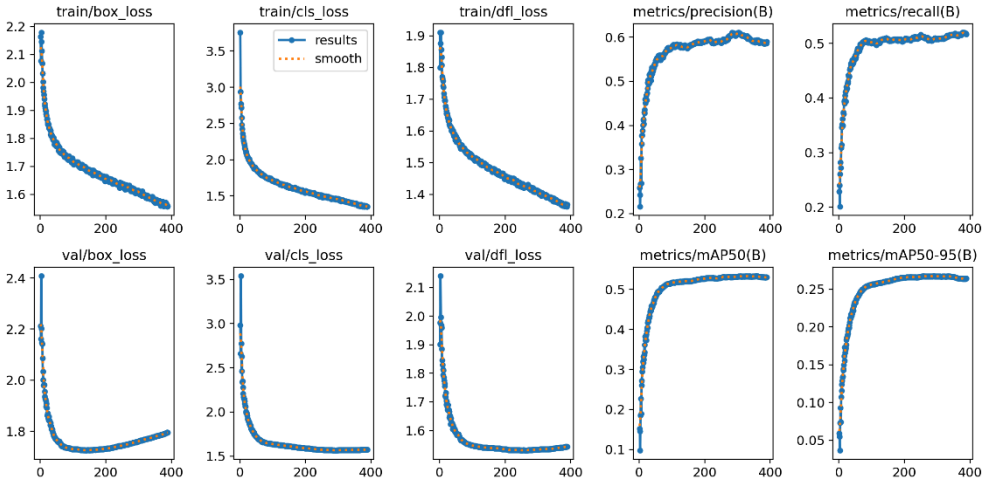


Fig.5 Training and Validation Loss Curves

The final model was evaluated on the 10% test split, achieving a mean Average Precision (mAP) at an IoU of 0.5 (mAP@50) of 0.554, and an mAP@50-95 of 0.278. The dataset distribution for this evaluation is shown in **Fig. 6**. Performance varied by class, with alligator cracks (D20) being the most accurately detected (mAP@50 = 0.679), while longitudinal cracks (D00) were the most challenging (mAP@50 = 0.493).

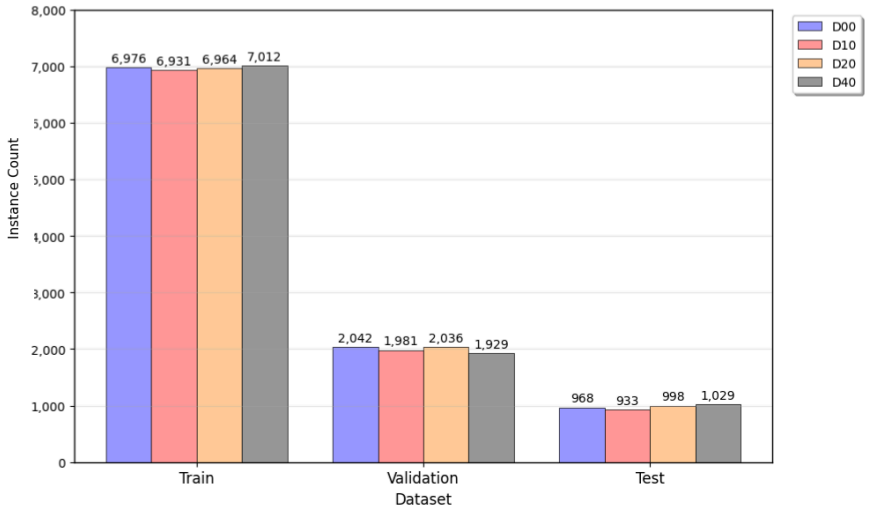


Fig.6 Instance Distribution for Each Class

A key finding from the confusion matrix (**Fig. 7**) revealed that the primary source of error was not misclassification between damage types, but a high rate of false negatives—the model often classified a real instance of damage as "background."

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

Fig.7 Confusion Matrix

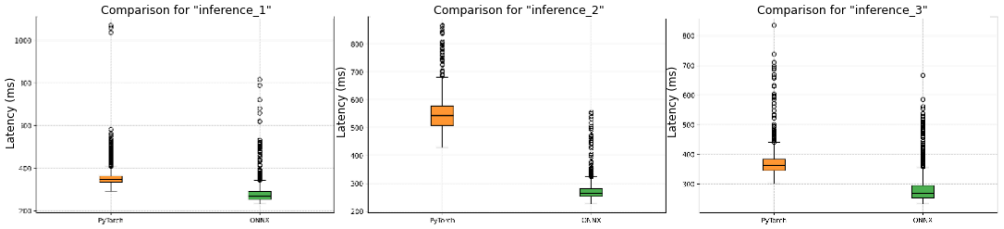
3.2 System Performance: Optimization for Real-Time Edge Computing

A critical objective of this research was to validate the feasibility of real-time processing on a low-cost edge device. To quantify this, we compared the performance of the baseline PyTorch (.pt) model against the optimized ONNX (.onnx) model on the Raspberry Pi 5.

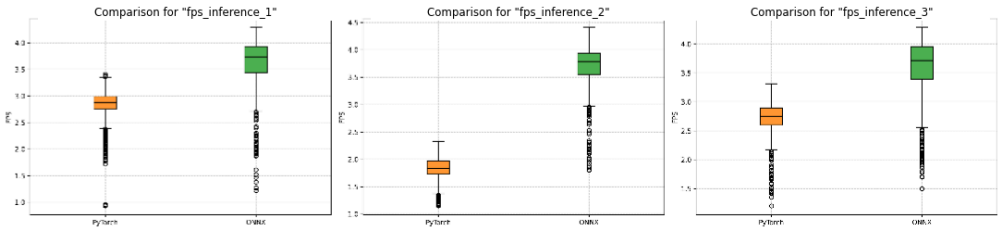
The results, summarized in **Table 1** (and detailed in **Fig. 8**), show a dramatic improvement. The baseline PyTorch model was unstable, with performance dropping as low as 1.85 FPS, failing to meet the 2 FPS target. In contrast, the ONNX model achieved a stable average processing speed of 3.66 FPS with an average latency of 277.69 ms. This optimization yielded a 47.6% increase in processing speed, successfully exceeding the system's performance target.

Table 1. PyTorch vs. ONNX Model Performance Statistics

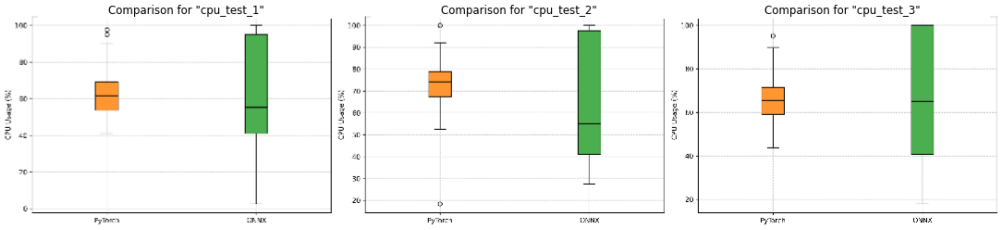
No	Parameter	Model	Average (3 Test)	Standard Deviation
1.	Latency (ms)	PyTorch (`.pt`)	424.11	49.71
		ONNX (`.onnx`)	277.69	43.01
2.	FPS	PyTorch (`.pt`)	2.48	0.23
		ONNX (`.onnx`)	3.66	0.41
3.	CPU (%)	PyTorch (`.pt`)	66.67	9.98
		ONNX (`.onnx`)	65.05	26.74
4.	RAM (%)	PyTorch (`.pt`)	42.44	2.5
		ONNX (`.onnx`)	45.06	1.81
5.	Temperature (˚C)	PyTorch (`.pt`)	65.25	0.96
		ONNX (`.onnx`)	71.79	2.15
6.	Power (Watt)	PyTorch (`.pt`)	14.6	0.9
		ONNX (`.onnx`)		



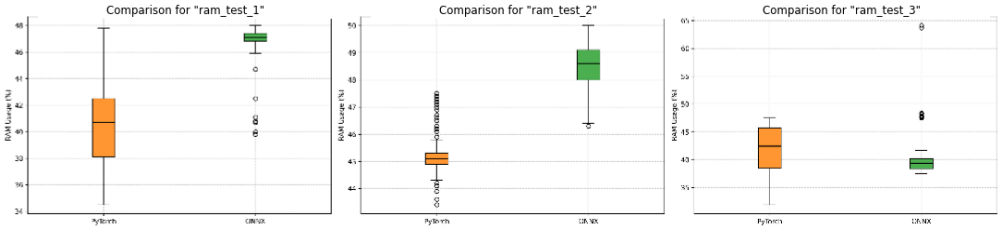
A. Latency PyTorch vs. ONNX



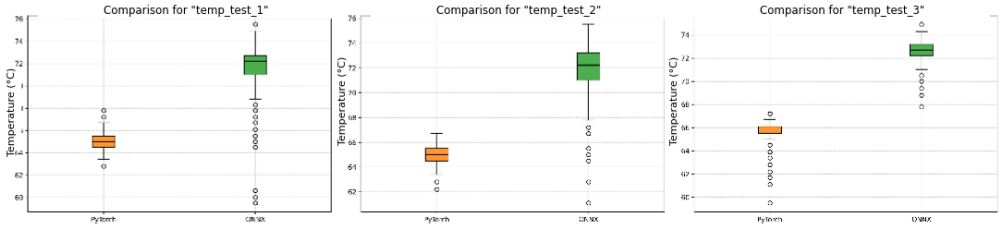
B. FPS PyTorch vs. ONNX



C. CPU Usage PyTorch vs. ONNX



D. RAM Usage PyTorch vs. ONNX



E. Temperature PyTorch vs. ONNX

F. Power Usage

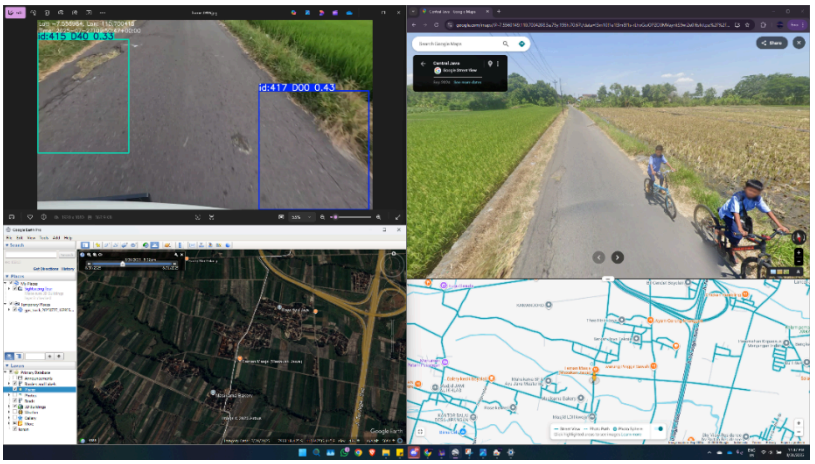
Fig.8 PyTorch vs. ONNX usage

Furthermore, all system efficiency targets were met. During operation, the system consumed an average of 45.06% of RAM and 65.05% of the CPU. The average CPU temperature was 71.79°C, well below the 85°C thermal throttle limit, and the system drew an average of only 14.6W of power. This confirms the system's ability to operate efficiently and stably on low-cost hardware.

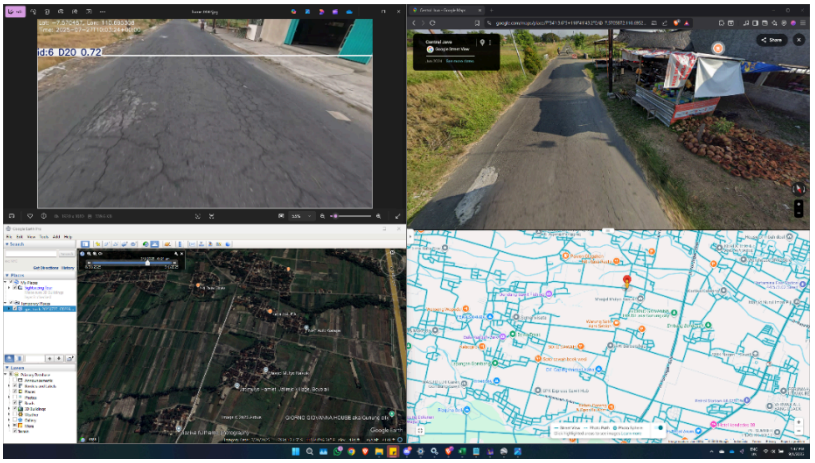
3.3 Field Validation and Geospatial Functionality

The system's geospatial functionality was validated in three distinct field tests (semi-rural, rural, and urban). In all scenarios, the GPS module successfully acquired a satellite lock and accurately recorded the vehicle's path. The system demonstrated its core capability by correctly synchronizing the detected damage instances with their precise geographical coordinates, generating a functional KML track (**Fig. 9**) for post-survey mapping and analysis.

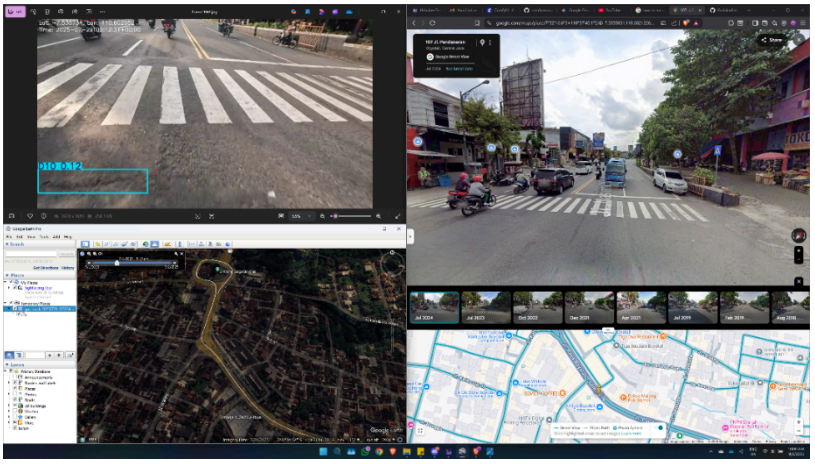
ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)



A. Semi-rural Area



B. Rural Area



C. Urban Area

Fig.9 Geospatial testing

3.4 Discussion

The findings validate that the proposed system successfully addresses the gap identified in the literature. With an mAP@50 of 0.554 and a stable 3.66 FPS on a Raspberry Pi 5, this work demonstrates a balance between accuracy and efficiency on a low-cost platform.

Unlike cloud-dependent systems [2] or post-processing-based systems [3], our platform is fully autonomous and operates in real-time. While Zanevych et al. [6] achieved a higher mAP (0.72), their model was limited

to a single class (potholes). Our system, in contrast, provides a more comprehensive 4-class detection, which is more comparable to the scope of Sidqi et al. [4]. Notably, our mAP of 0.554, validated on the actual target hardware (Pi 5), surpasses the mAP of 0.497 reported by the lightweight YOLO-LWNet model [5], which was benchmarked on a high-power desktop GPU. This confirms the contribution of this research: a functional, low-cost, and real-time edge-computing platform for multi-class road damage monitoring.

4. Conclusion

This study successfully addressed the objective stated in the introduction: to design, implement, and validate an autonomous, real-time road damage detection system on a low-cost edge platform. The results confirm that the system, based on YOLOv11 and optimized with ONNX, is fully functional on a Raspberry Pi 5. It achieved reliable real-time performance, exceeding the target with 3.66 FPS, and successfully integrated precise GPS data. The model demonstrated its capability to detect four damage classes with an mAP@50 of 0.554, confirming the feasibility of this approach.

Based on the findings, the primary limitation identified was the model's high rate of false negatives (undetected damage). Future work should therefore focus on improving model accuracy by retraining the model with a more varied and augmented dataset. Furthermore, the system's prospects can be expanded by developing advanced functionalities, such as classifying damage severity and creating a web-based visualization dashboard to enhance data usability for maintenance stakeholders. Validating the system's performance at different vehicle speeds and exploring AI accelerators for further optimization also remain valuable avenues for future research.

Author Contribution:

All authors contributed equally as main contributors to this paper. All authors read and approved the final paper.

Funding:

This research received no external funding.

Conflicts of Interest:

The authors declare no conflict of interest.

References

[1] D. J. B. M. Kementerian Pekerjaan Umum, “Pedoman Sistem Pemeliharaan Jalan Kota (City Road Management System),” Feb. 2025. [Online]. Available: https://binamarga.pu.go.id/index.php/nspk/detail/02pbm2025-pedoman-sistem-pemeliharaan-jalan-kota-city-road-management-system

[2] C.-C. Hsieh, H.-W. Jia, W.-H. Huang, and M.-H. Hsih, “Deep learning-based road pavement inspection by integrating visual information and IMU,” *Information*, vol. 15, no. 4, p. 239, 2024, doi: 10.3390/info15040239.

[3] K. Mutijarsa, F. A. Alunjati, F. Hidayat, R. Kurnia, and Z. Arsyad, “Smart road damage survey system using machine vision and IoT technology,” *2023 3rd International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)*, pp. 391–396, 2023, doi: 10.1109/icicyta60173.2023.10428992.

[4] A. J. Sidqi, F. A. Alunjati, U. Elviani, and F. Hidayat, “Development of longitudinal and transversal crack detection feature in automated road pavement condition survey system using YOLO algorithm,” *2024 International Conference on ICT for Smart Society (ICISS)*, pp. 1–5, 2024, doi: 10.1109/iciss62896.2024.10751400.

[5] C. Wu, M. Ye, J. Zhang, and Y. Ma, “YOLO-LWNet: A lightweight road damage object detection network for mobile terminal devices,” *Sensors*, vol. 23, no. 6, p. 3268, 2023, doi: 10.3390/s23063268.

[6] Y. Zanevych, V. Yovbak, O. Basystiuk, N. Shakhovska, S. Fedushko, and S. Argyroudis, “Evaluation of pothole detection performance using deep learning models under low-light conditions,” *Sustainability*, vol. 16, no. 24, p. 10964, 2024, doi: 10.3390/su162410964.

[7] R. Sapkota, M. Flores-Calero, R. Qureshi, C. Badgujar, U. Nepal, A. Poulose, P. Zeno, U. B. P.

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

Vaddevolu, S. Khan, M. Shoman, H. Yan, and M. Karkee, "YOLO Advances to Its Genesis: A Decadal and Comprehensive Review of the You Only Look Once (YOLO) Series," *Artificial Intelligence Review*, vol. 58, Art. no. 274, 2025, doi: 10.1007/s10462-025-11253-3.

[8] S. Jana, A. I. Middya, and S. Roy, "Mobile Crowd Sensing for Road Anomaly Detection Using Deep Learning," *Spatial Information Research*, vol. 33, Art. no. 51, 2025, doi: 10.1007/s41324-025-00651-y.

[9] D. Arya, H. Maeda, S. K. Ghosh, D. Toshniwal, and Y. Sekimoto, "RDD2022: A Multi-National Image Dataset for Automatic Road Damage Detection," *Geoscience Data Journal*, 2024, doi: 10.1002/gdj3.260.

[10] D. Minott, S. Siddiqui, and R. J. Haddad, "Benchmarking Edge AI Platforms: Performance Analysis of NVIDIA Jetson and Raspberry Pi 5 with Coral TPU," in *2025 IEEE SoutheastCon*, 2025, pp. 1384–1389, doi: 10.1109/SoutheastCon56624.2025.10971592.

JRAEE

Journal of Robotics, Automation, and Electronics Engineering
Vol. 4, No. 1, 2026, pp. 237-249
e-ISSN 3025-3780
https://journal.uny.ac.id/v3/jraee

Prototype of a Coffee Bean Weight Measuring Device Using a Webcam with the Convolutional Neural Network (CNN) Method at Roetin Coffee Shop

Nurul Afifah[a,1], Faris Yusuf Baktiar[b,2,*]

[a,b]Department of Electrical and Electronics Engineering, Vocational Faculty, UNY
[1]nurulafifah.2021@student.uny.ac.id; [2]farisyusufbaktiar@uny.ac.id
[*]Corresponding Author

ARTICLE INFO

Article History
Received Date Month Year
Revised Date Month Year
Accepted Date Month Year

Keywords
CNN;
Image Processing;
Mobile Net;
Coffee Beans.

ABSTRACT

The advancement of Artificial Intelligence (AI) and Computer Vision has enabled new opportunities for automation within the coffee industry, particularly in weight measurement of coffee beans, which is still performed manually and becomes inefficient at large scale. This study proposes an automatic weight estimation system using images captured by a webcam and processed through a Convolutional Neural Network (CNN) employing MobileNet as a lightweight regression model. The developed system analyzes visual features to estimate weight autonomously, offering an efficient, contactless alternative to conventional weighing tools and supporting stock monitoring for coffee industries and small enterprises.

ABSTRAK

Perkembangan kecerdasan buatan (AI) dan pengolahan citra digital (Computer Vision) membuka peluang baru dalam otomasi industri kopi, termasuk proses pengukuran berat biji kopi yang selama ini masih dilakukan secara manual dan kurang efisien dalam skala besar. Penelitian ini merancang sistem estimasi berat biji kopi dalam toples transparan menggunakan citra dari webcam yang diproses dengan Convolutional Neural Network (CNN) arsitektur MobileNet sebagai model regresi untuk memprediksi berat berdasarkan fitur visual. Sistem yang dikembangkan bersifat otomatis dan berpotensi menjadi alternatif timbangan konvensional serta mendukung monitoring stok pada industri kopi maupun UMKM.

1. Introduction

The coffee shop industry in Indonesia is experiencing rapid growth, making raw material inventory management a vital component for maintaining production continuity [1, 2]. However, Roetin Coffee Shop still performs stock recording manually using paper and packaging-based quantity estimation rather than actual weight, which increases the risk of data discrepancies and record loss, especially since the remaining coffee beans inside the grinder are not reweighed [3, 4]. Inaccurate stock opname may lead to shortages of essential raw materials that disrupt sales activities [5], indicating the need for an automatic inventory system

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

that is more accurate, efficient, and real-time [6, 7].

Advancements in Artificial Intelligence and Computer Vision enable the implementation of image based weight estimation using Convolutional Neural Networks (CNN), where MobileNet is selected for its efficiency and speed in visual feature extraction [8, 9, 10]. Similar studies have been conducted on palm fruit weight prediction [11], cattle weight estimation using image processing [12], digital inventory and stock opname systems [5], and load-cell-based measurement approaches [13, 4]; however, no studies have been found that apply CNN specifically for direct coffee bean weight estimation to support coffee shop inventory management. Therefore, this research proposes an automatic coffee bean weight estimation system using a webcam and MobileNet, with its main contribution being the integration of weight measurement and real time inventory logging to improve stock accuracy [14].

2. Problem Solving Approach

2.1 Computer Vision

In this study, a Computer Vision approach was selected as the main method for estimating the weight of coffee beans. This approach was chosen because it can capture visual features of objects directly from images, without requiring physical contact like manual weighing. The visual characteristics of coffee beans, including size, color, surface texture, and the distribution of beans within the container, can be digitally processed and analyzed as indicators of weight estimation. This method also allows measurements to be conducted quickly, continuously, and automatically using only a cam-era, making it highly suitable for monitoring daily stock at Roetin Coffee Shop, which requires time and labor efficiency. Therefore, Computer Vision becomes an effective and practical alternative to conventional weighing methods that still require manual interaction and are prone to recording errors.

2.2 Convolutional Neural Network (CNN)

After determining the visual approach, the classification and feature extraction model used is the Convolutional Neural Network (CNN). CNN was chosen due to its superior capability in recognizing visual patterns compared to traditional machine learning methods, which still rely on manual feature engineering. CNN can automatically extract important information from images through convolutional layers, so details such as bean shape, pile density, and color intensity can be accurately analyzed. In the context of this study, the ability of CNN to map the visual relationship between coffee bean images and weight values is essential to achieve precise estimation results. With high efficiency and accuracy, CNN provides the best foundation for building an image-based coffee bean weight prediction model.

2.3 MobileNet (CNN Architecture)

To ensure the system can run lightweight and fast on devices with limited computational resources, such as operational laptops and webcams, the MobileNet architecture was chosen as the main CNN model.

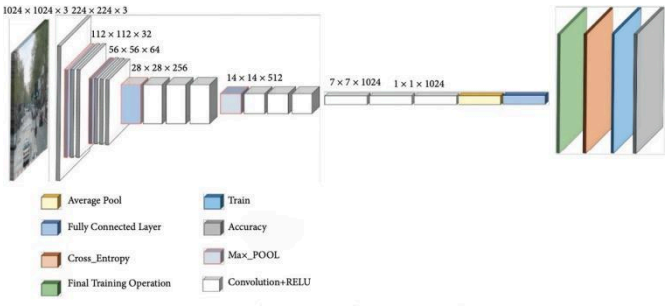


Fig. 1 Architecture Mobile Net

MobileNet is specifically designed to be more efficient in memory and computational usage through the depthwise separable convolution mechanism, allowing training and prediction processes to run much faster compared to larger CNN architectures, as illustrated in Fig. 1. This advantage makes MobileNet not only efficient but also ideal for real-time applications in work environments such as coffee shops, which require quick response in stock recording. With the selection of MobileNet, the coffee bean weight estimation system can operate optimally without high-specification devices, thereby increasing the potential for direct implementation in business operations.

3. Research Method

This study uses the Research and Development (R&D) method, which includes stages from system design to testing to evaluate the performance of a prototype automatic coffee bean weight measurement device, as shown in Fig. 2. The system is developed using image processing technology based on MobileNet integrated with a webcam, so that coffee bean weight measurement is no longer performed manually using a scale, but automatically through image analysis. The research stages include problem identification related to manual stock recording prone to errors, literature review and collection of a coffee bean image dataset in various containers and fill levels, design of the device including the placement of main components such as the Raspberry Pi and OLED display, system implementation with training of the CNN MobileNet model, and prototype testing and evaluation based on the accuracy of weight estimation compared to a digital scale.

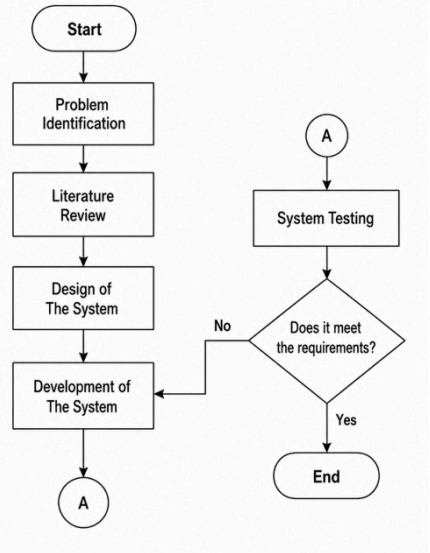


Fig. 2 Research method

The research was conducted at Roetin Coffee Shop, Trimulyo I, Kepek, Kec. Wonosari, Kab. Gunungkidul, Yogyakarta, from April 2025 to November 2025. The success of the system is measured based on its ability to produce consistent weight estimates with a low error margin (less than 5%) and the ease of monitoring daily coffee bean stock automatically through an application interface that displays real time results.

3.1 Requirements Analysis

The automatic coffee bean weight measurement system requires hardware that supports image acquisition, data processing, and real-time result display. A laptop is used for storing and managing the coffee bean image dataset as well as for visual verification before the model is trained, while the Raspberry Pi 4 Model B serves as the main processing unit, handling all image processing, MobileNet model inference, and displaying the estimated weight results. A webcam captures images of the coffee jars automatically, a tripod stabilizes the camera to ensure consistent images, and a 1.3-inch OLED screen displays the estimated results with high contrast and good visibility, even under low-light conditions. The complete setup of the required hardware can be seen in Fig. 3, which illustrates all the main components used in this system.

Fig. 3 Hardware requirements

The software used includes Visual Studio Code as the source code editor for developing Python programs, including image processing, training and implementing the MobileNet model, and creating a simple user interface. Python is the main programming language due to its flexibility in executing the entire system logic, from image acquisition via webcam, image pre-processing

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

using OpenCV, to training and inference of the MobileNet model for automatic coffee bean weight estimation.

3.2 System Architecture Design

The proposed system for automatic coffee bean weight measurement utilizes a visual approach based on a webcam and an artificial intelligence model, which can be seen in the system architecture shown in Fig. 4. In this system, the webcam captures daily images of jars containing coffee beans. The images are then processed on the server using the OpenCV library for image pre-processing, including resizing, contrast adjustment, and segmentation of the coffee bean area within the jars. After pre-processing, the images are input into a pre-trained MobileNet model to estimate the coffee bean weight based on visual characteristics such as height, density, and color of the jar contents. The model runs on a Raspberry Pi and outputs weight estimations in grams without the need for physical sensors like load cells. The estimated data can be displayed on a website for daily stock recording and stored in a database for future reference, as shown in Fig. 4.

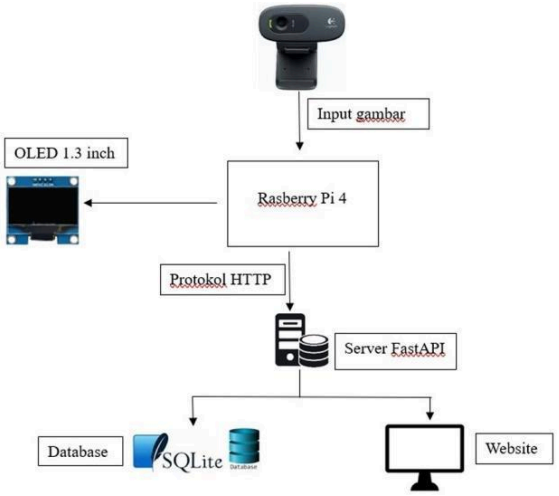


Fig. 4 System architecture design

To support model training, a dataset was collected to provide both training and testing data for the MobileNet-SSD model. The dataset consists of seven object classes: toples1, toples2, toples3, toples4, plastik1kg, plastik2kg, and bijikopi. Images were captured from multiple angles and lighting conditions to increase dataset variability. All collected images were annotated manually using Labeling software, as can be seen in Fig. 5, in Pascal VOC (XML) format to ensure accurate bounding box positioning for each object. The annotated dataset was divided into training (70%), validation (20%), and test (10%) subsets, with 2,694 images for training, 769 for validation, and 386 for testing. The training set enables the model to learn object patterns, the validation set evaluates model performance during training to prevent over-fitting, and the test set assesses the model's generalization capability on previously unseen data.

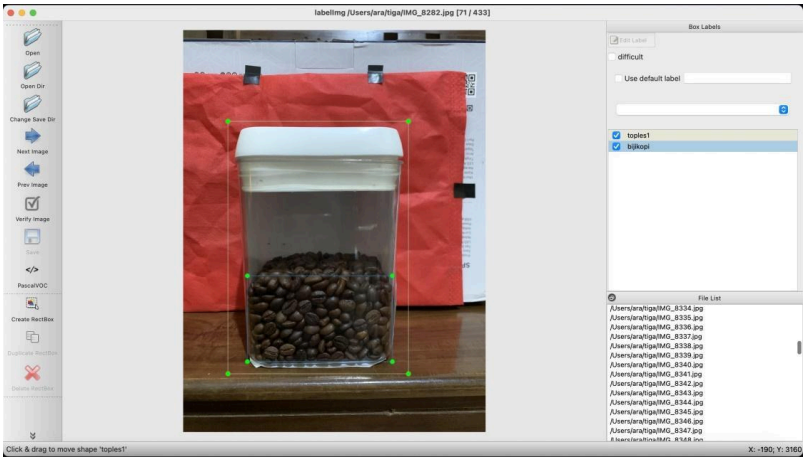


Fig. 5 Labelling system

3.3 Image Processing Algorithm Flow for Coffee Beans

The image processing in the coffee bean weight estimation system based on computer vision using MobileNet consists of four main stages: image acquisition, pre-processing, feature extraction, and weight prediction.

3.3.1 Image Acquisition

Images were captured using a Logitech C270 camera connected to a Raspberry Pi. The camera acquired frontal images of transparent jars containing varying amounts of coffee beans, as shown in Fig. 6 and Fig. 7, with the jars positioned perpendicular to the surface. The lighting was adjusted to minimize reflections on the jar surfaces that could affect image quality. Captured images were automatically stored in the dataset folder according to their actual weight labels, with 5–10 images per category used for training.

3.3.2 Image Pre-Processing

The image pre-processing stage consists of several key steps. First, the images are resized to 320×320 pixels to match the input layer of the model and reduce computational load. Next, normalization is applied, converting pixel values from the range 0–255 to 0–1 to stabilize the distribution of values and improve model performance. During training, data augmentation such as rotation, horizon-tal flipping, and brightness adjustment is performed to increase the model's robustness to variations in lighting and camera angles. Additionally, the image color format is converted from BGR (Blue, Green, Red) to RGB (Red, Green, Blue) so that the colors interpreted by the model accurately reflect the original image.

Fig. 6 Image acquisition

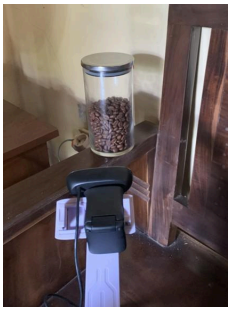


Fig. 7 Front view image capture

3.3.3 Feature Extraction

Feature extraction is carried out by the MobileNet architecture integrated with a Single Shot Detector (SSD). MobileNet extracts visual patterns from RGB images, from low-level features such as edges and textures to high-level features such as object shapes and characteristics. These features are processed by a Feature Pyramid Network (FPN) to ensure that both small and large objects are consistently recognized. The extracted features are then used by the SSD to locate objects (bounding boxes) and classify them into categories such as jars, plastic bags, and coffee beans.

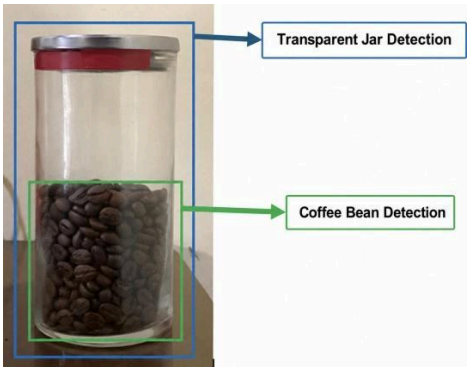


Fig. 8 Detection of transparent jars and coffee beans

The identification of important areas within the image is illustrated in Figure 8, where the trans-parent jar area is highlighted with a blue box, and the coffee bean contents are marked with a green box.

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

3.3.4 Coffee Bean Weight Prediction

Each type of container has a unique shape and volume, meaning the relationship between the height of the coffee bean content (cm) and its weight (grams) varies. Weight estimation is performed separately for each container using a linear approach based on empirical calibration. The height-to-weight relationship is calculated using linear interpolation as follows:

$$w = w_1 + (w_2 - w_1) \times \frac{(h - h_1)}{(h_2 - h_1)} \quad (1)$$

Where:
- w = estimated weight of coffee beans (grams)
- h = detected height of coffee beans (cm)
- h_1, h_2 = reference heights from the calibration table (cm)
- w_1, w_2 = reference weights from the calibration table (grams)

3.4 Hardware Design

The hardware design of the coffee bean weight estimation system using visual images consists of five main components Raspberry Pi 4 Model B, 1.3-inch OLED, laptop, webcam, and tripod. The laptop serves as the central hub for dataset management and program development, including image acquisition from the webcam, image pre-processing using OpenCV, and training the MobileNet model. Once the training is complete, the model is deployed on the Raspberry Pi, which acts as the main computational unit to process input images, estimate weight, and display prediction results. The webcam captures real-time images of coffee jars and is mounted on a tripod to ensure stable position-ing and optimal image acquisition. The estimated weight is displayed on the 1.3-inch OLED, directly connected to the Raspberry Pi, allowing quick access to weight information without additional devices. The hardware system design is illustrated in Fig. 9.

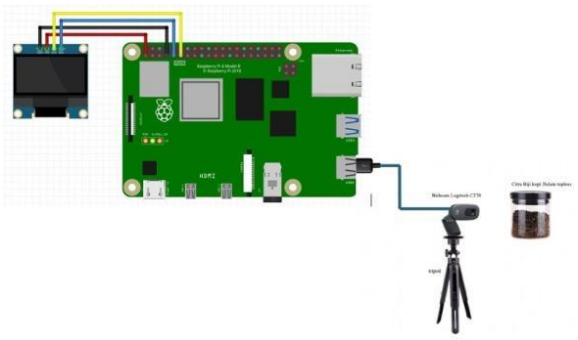


Fig. 9 Hardware design of the system

3.5 Software Design

The software design of the system consists of several main modules. First, image acquisition of coffee jars is performed using a webcam connected to the Raspberry Pi, capturing images periodically at the end of each operational day to serve as visual input data. Second, weight prediction is carried out using the MobileNet model, which has been trained on a dataset of coffee bean images with actual weight labels; the inference output provides an estimated weight in grams. Third, the predicted data is transmitted and stored in a database for automatic daily stock recording, facilitating inventory monitoring. Finally, the estimated results are displayed via a simple web interface, enabling the staff of Roetin Coffee Shop to evaluate raw material usage, plan purchases, and minimize waste.

3.6 System Testing

System testing was conducted to evaluate the overall performance of the image based coffee bean weight estimation system. The first test involved the confidence score, which aimed to ensure that the detection model could recognize objects with a high level of confidence in each prediction, both for container bounding boxes and coffee beans. The higher the confidence score, the greater the model's trust in its detection results. Subsequently, the weight prediction test was carried out by taking three measurements for each container using the webcam and CNN based system, and then comparing the results with the actual weight to calculate the average and the percentage of error. The test results indicate that the system is

capable of providing weight estimates with a satisfactory level of accuracy according to the prior calibration.

Additionally, an integration test was performed for the website status, database storage, and OLED display. This test ensured that all functions operated smoothly, from coffee bean weight prediction to data storage in the database, and real-time visualization on the OLED screen. Based on the testing results, all system components demonstrated good performance, as indicated by the ”success” status on the website, correctly stored data in the database, and weight results displayed directly on the OLED screen, supporting automatic monitoring of coffee bean inventory.

4. Result and Discussion

4.1 Hardware Implementation Results

The hardware implementation of the coffee bean weight estimation system consists of two main parts, namely device design and component integration. The device is designed in the form of a compact enclosure that functions as a protective casing for all electronic components. The box is built with dimensions of approximately 14.5 cm × 9.5 cm × 5 cm, adjusted to the size of the Raspberry Pi 4 Model B which serves as the main processing unit. A dedicated slot is provided on the top side of the box for the installation of a 1.3-inch OLED display, which is used to present real-time coffee weight estimation results. The physical appearance of the hardware casing is shown in Fig. 10.

Fig. 10 Hardware en-closure design

Fig. 11 Integrated hardware assembly with enclosure

The hardware integration process is carried out by placing and connecting all electronic components inside the enclosure based on the predefined layout. The Raspberry Pi 4 Model B functions as the main controller responsible for executing the CNN inference process, handling network communication with the FastAPI server, and sending the estimated weight output to the OLED display. The webcam is connected to the Raspberry Pi via a USB port to capture coffee bean container images, while the OLED communicates using an I2C interface through GPIO 17 (SCL) and GPIO 27 (SDA). The integrated hardware assembly is presented in Fig. 11.

4.2 Software Implementation Results

The software implementation of the coffee bean weight estimation system consists of three main programs: the coffee bean weight estimation program, the FastAPI server with database storage, and the website dashboard. The estimation program was developed using the Python programming language on a Raspberry Pi, supported by OpenCV, TensorFlow, and luma.oled modules. The system captures images through a camera and detects the coffee beans and container using a pre-trained MobileNet-SSD model. Weight estimation is obtained by calculating the filling height of the container based on the vertical coordinate difference of the bounding box, then converting it into centimeters using a calibration factor stored in config.json. The height value is then processed using a linear interpolation method according to the calibration data of each container. The estimated height and weight are displayed through the OLED screen and periodically sent to the server, enabling automatic stock monitoring without the need for physical weight sensors.

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

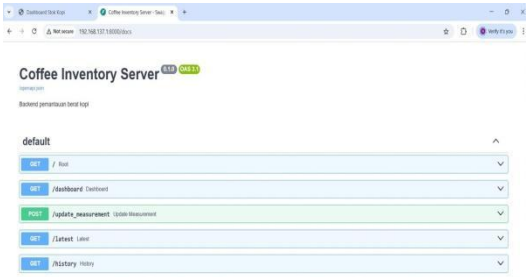


Fig. 12 Fast API server display

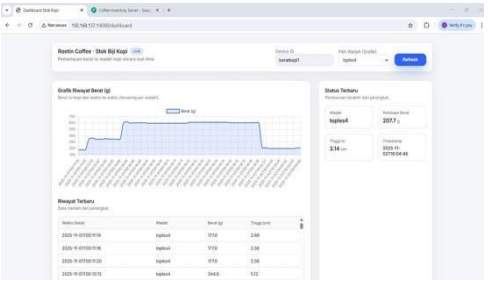


Fig. 13 Website dashboard for estimated coffee bean weight

The second program is the FastAPI server, which functions as a data communication handler between the Raspberry Pi and the website. The server receives estimated weight data via the HTTP protocol, stores it in an SQLite database, and distributes it to the web dashboard. The interface display of the server is shown in Fig. 12. The third program is the website display system which serves as a monitoring panel for coffee bean stock. The website presents information about container type, estimated height, estimated weight, as well as measurement history in table and graphic form to facilitate stock analysis. The website dashboard visualization is shown in Fig. 13.

4.3 Implementation of the CNN (Convolutional Neural Network) Method

The MobileNet SSD (Single Shot MultiBox Detector) method was implemented to detect container and coffee bean objects during the weight estimation process based on the height of the beans inside the container. MobileNet-SSD combines the lightweight MobileNetV2 architecture as the feature extractor with the SSD network as the object detector, enabling multi-object detection in a single inference.

This process began with training the MobileNet-SSD model using an image dataset obtained from six types of containers, consisting of four jar types (toples1, toples2, toples3, toples4) and two plastic pack types (plastik1kg and plastik2kg). Each image also contained a coffee bean object (bijikopi) used as an indicator of the container fill height. All images were annotated using Labeling in Pascal VOC (XML) format to define accurate bounding boxes and object classes.

The dataset was divided into 70% train set, 20% validation set, and 10% test set, with a total of 3,849 images: 2,694 training data, 769 validation data, and 386 test data. The dataset consisted of seven main classes: toples1, toples2, toples3, toples4, plastik1kg, plastik2kg, bijikopi. Model train-ing was conducted using the TensorFlow Object Detection API with the SSD MobileNetV2 FPNLite 320×320 architecture. The training configuration consisted of 25,000 steps and batch size = 4. The training process ran for approximately 21 hours on Google Colab CPU using Python 3.10 and TensorFlow 2.12.0.

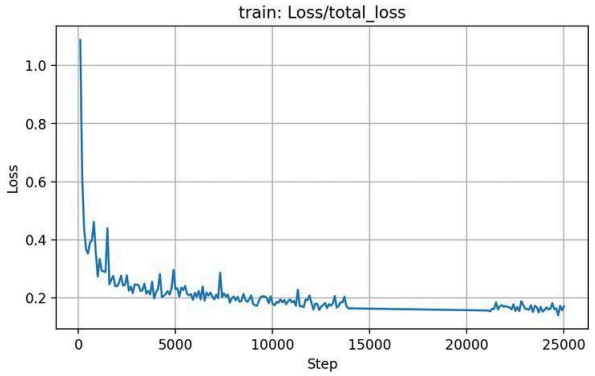


Fig. 14 Model training total loss graph

Based on the training graph, the initial loss value was very high because the weights were still random. As iterations increased, the loss decreased progressively until it stabilized below 0.2 around 20,000–25,000 steps, indicating that the model had successfully learned and reached convergence. The model evaluation results are shown through the mAP and classification performance metrics below:

- mAP@50:0.95 = **0.957**
- mAP@50 = **0.999**
- mAP@75 = **0.999**

• Recall = **0.971**

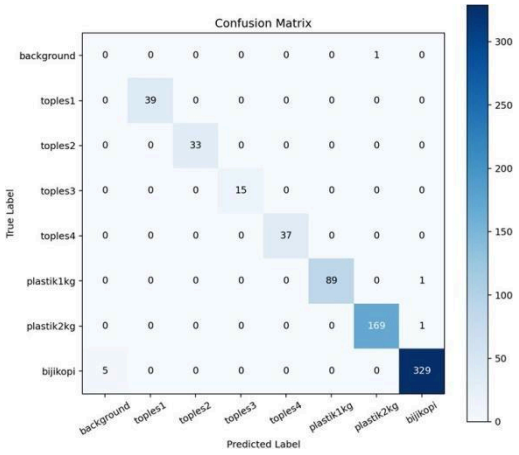


Fig. 15 Model confusion matrix results

The model was further evaluated using a confusion matrix for two major classification groups: coffee beans (bijikopi) and containers (jar/plastic). Precision, Recall, F1-Score, and Accuracy for each class are shown in Table 1. The results indicate that the model achieved very high performance with an overall accuracy of 0.994, showing that object detection is highly accurate and suitable for application in the weight estimation system.

Table 1. Precision, Recall, F1-Score, and Accuracy for Each Class

Class	Precision	Recall	F1-Score	Accuracy
toples1	1.000	1.000	1.000	1.000
toples2	1.000	1.000	1.000	1.000
toples3	1.000	1.000	1.000	1.000
toples4	1.000	1.000	1.000	1.000
plastik1kg	0.988	0.989	0.988	0.994
plastik2kg	0.994	0.994	0.985	0.985
bijikopi	0.994	0.994	0.985	0.985

(a) Original Image of Plastic Bag 1 kg

(b) Original Image of Plastic Bag 2 kg

(c) Webcam Detection Result of Plastic Bag 1 kg

(d) Webcam Detection Result of Plastic Bag 2 kg

Fig. 16 Capture and detection results for 1kg and 2kg plastic containers

During image capture, the camera position was adjusted to clearly display the plastic container outline and coffee surface. However, detection issues occurred with transparent jars, as the container boundary was difficult to read, resulting in inaccurate bounding boxes. The applied solution was adding a sticker behind

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

the jar to increase edge contrast, allowing bounding box detection to become more stable and accurate.

4.4 Project Testing

Project testing was carried out to evaluate the system's ability in estimating coffee bean weight based on image processing, starting from object detection, weight prediction, accuracy calculation, to data integration with server and OLED display. The testing is divided into four main stages as follows.

4.5.1 Confidence Score Testing

This test aims to determine the CNN model's capability in recognizing container and coffee bean objects through the confidence value on the bounding box. The higher the confidence value, the stronger the model certainty in detecting an object. The confidence score test results are shown in Table 2. Confidence values range from 0.68–1.00, indicating that object detection runs well across all test samples.

Fig. 17 Inaccurate bounding box on trans-parent jar

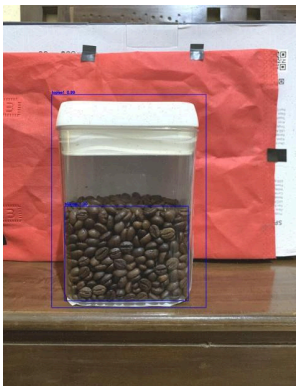
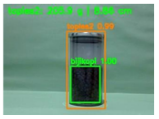
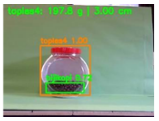


Fig. 18 Accurate bounding box after sticker placement

Table 2. Confidence Score Test Results on Bounding Box

Container	Image	Container Conf.	Coffee Bean Conf.
Jar 1		0.95	1.00
Jar 2		0.99	1.00
Jar 3		0.98	0.96
Jar 4		1.00	0.72
Plastic 1 Kg		0.96	1.00
Plastic 2 Kg		1.00	0.68

4.5.2 Coffee Bean Weight Prediction Testing

The testing was performed by comparing three measurement estimations (P1, P2, P3) for each container with the actual weight. The error percentage was calculated using Eq. 2.

$$\text{Error}(\%) = \frac{\text{Estimated Weight} - \text{Actual Weight}}{\text{Actual Weight}} \times 100\% \quad (2)$$

The testing summary is presented in Table 3. From a total of 18 samples, the system achieved an average error of only 0.91%.

Table 3. Coffee Bean Weight Prediction Testing

Container	Actual	P1	P2	P3	Avg	Error %
Jar 1	200	203.1	197.9	197.9	199.6	0.20
	400	400.9	400.9	400.9	400.9	0.22
	500	499.9	494.7	499.9	498.1	0.38
Jar 2	100	99.1	101.6	99.1	99.9	0.07
	200	205.9	208.9	205.9	206.9	3.45
	300	307.4	310.1	304.8	307.4	2.46
Jar 3	120	119.2	123.5	119.2	120.6	0.50
	300	312.5	304.5	304.5	307.1	2.36
	500	510.2	518.3	500.8	509.7	1.94
Jar 4	200	197.8	207.7	197.8	201.1	0.55
	350	344.6	353.5	353.5	350.5	0.14
	600	602.3	594.6	610.0	602.3	0.38
Plastic 1kg	100	100.2	97.9	100.2	99.4	0.60
	200	201.4	198.6	201.4	200.4	0.20
	300	312.4	303.0	298.4	304.6	1.53
Plastic 2kg	100	97.7	102.7	97.7	99.3	0.70
	500	507.9	498.2	498.2	501.4	0.28
	700	713.1	693.5	703.3	703.3	0.47
Average Error						**0.91**

4.5.3 Overall Accuracy Testing

System accuracy was calculated using Eq. 3.

$$\text{Accuracy}(\%) = \left(1 - \frac{|\text{Estimated Weight} - \text{Actual Weight}|}{\text{Actual Weight}}\right) \times 100\% \quad (3)$$

Table 4 shows that the system achieved an average accuracy of 99.09%.

Table 4. Overall Accuracy Testing

Container	Actual	Estimated	Accuracy (%)
Jar 1	200	199.6	99.60
	400	400.9	99.77
	500	499.9	99.98
Jar 2	100	99.9	99.93
	200	206.9	96.55
	300	307.4	97.53
Jar 3	120	120.6	99.50
	300	307.1	97.63
	500	509.7	98.06
Jar 4	200	201.1	99.45
	350	350.5	99.85
	600	602.3	99.61
Plastic 1kg	100	99.4	99.40
	200	200.4	99.80
	300	304.6	98.46
Plastic 2kg	100	99.3	99.30
	500	501.4	99.72
	700	703.3	99.52
Overall Accuracy			**99.09%**

5. Conclusion

Based on the process of designing, implementing, and testing a non-contact coffee bean weight measurement system using a webcam and a Convolutional Neural Network (CNN) with the MobileNet SSD architecture, the developed system successfully detected coffee beans and the container automatically and performed quantity estimation with an average accuracy of 99.09% and a maximum error of 3.45%. These results

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

indicate that the system is capable of providing weight estimations that are very close to manual measurements, making it suitable as a supporting solution for stock opname activities at Roetin Coffee Shop. For further development, it is recommended to add an automatic calibration algorithm, increase dataset variations (lighting conditions, viewing angles, and container types), and integrate cloud storage services so that the system can operate more adaptively, accurately, and be accessed in real-time from various devices.

Author Contribution:

All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

Funding:

This research received no external funding.

Acknowledgment:

The author would like to express gratitude to Universitas Negeri Yogyakarta for providing research facilities and support.

Conflicts of Interest:

The authors declare no conflict of interest.

References

[1] Badan Pusat Statistik, Statistik Kopi Indonesia 2023. Jakarta, Indonesia: Badan Pusat Statistik, 2024, Publikasi No. 05100.24022, ISSN 2714-8505.

[2] K. Anam et al., Budidaya Tanaman Kopi dan Olahannya untuk Kesehatan. Makassar, Indonesia: CV Tohar Media, 2023, ISBN: 978-623-8148-42-4.

[3] J. G. Djoni and Tony, "Sistem manajemen stok barang berbasis web pada Toko Bangunan Sumber Abadi," Jurnal Ilmu Komputer dan Sistem Informasi, vol. 12, no. 1, 2024, doi: 10.24912/jiksi.v12i1.28273.

[4] Wahyudi, A. Rahman, and M. Nawawi, "Perbandingan nilai ukur sensor load cell pada alat penyortir buah otomatis terhadap timbangan manual," ELKOMIKA: Jurnal Teknik Energi Elektrik, Teknik Telekomunikasi, & Teknik Elektronika, vol. 5, no. 2, pp. 207–220, 2017, doi: 10.26760/elkomika.v5i2.207.

[5] M. M. Purba and C. Rahmat, "Perancangan sistem informasi stok barang berbasis web di PT Mahesa Cipta," JSI (Jurnal Sistem Informasi) Universitas Suryadarma, vol. 8, no. 2, pp. 123–158, 2021, doi: 10.35968/jsi.v8i2.721.

[6] O. Irnawati, "Implementasi metode Waterfall pada sistem informasi stock opname," Indonesian Journal on Software Engineering (IJSE), vol. 4, no. 1, pp. 79–84, 2018, doi: 10.31294/ijse.v4i1.6301.

[7] S. B. Nauli and Riduan, "Perancangan aplikasi persediaan barang menggunakan barcode Quick Response dengan metode First-In First-Out berbasis mobile (Studi kasus: PT Kopi Kenangan)," SENTRI: Jurnal Riset Ilmiah, vol. 3, no. 4, pp. 2126–2137, 2024, doi: 10.55681/sentri.v3i4.2608.

[8] A. Fuadi and A. Suharso, "Perbandingan arsitektur MobileNet dan NASNetMobile untuk klasifikasi penyakit pada citra daun kentang," JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika), vol. 7, no. 3, pp. 701–710, 2022, doi: 10.29100/jipi.v7i3.3026.

[9] Herdianto and D. Nasution, "Implementasi metode CNN untuk klasifikasi objek," METHOMIKA: Jurnal Manajemen Informatika & Komputerisasi Akuntansi, vol. 7, no. 1, pp. 54–60, 2023, doi: 10.46880/jmika.Vol7No1.pp54-60.

[10] S. Ramdani and A. Rahmatulloh, "Implementasi MobileNet untuk klasifikasi gambar dan deteksi emosi menggunakan KERAS," JUSTIN (Jurnal Sistem dan Teknologi Informasi), vol. 12, no. 2, pp. 259–264, 2024, doi: 10.26418/justin.v12i2.73389.

[11] A. Wijaya, B. R. Daryono, R. Toyib, and Y. Apridiansyah, "Analisis pengenalan pola pada citra digital untuk prediksi berat buah sawit," Decode: Jurnal Pendidikan Teknologi Informasi, vol. 4, no. 3, pp. 713–724, 2024, doi: 10.51454/decode.v4i3.481.

[12] D. B. Lasfeto and M. D. Letik, “Rancangan teknologi pengukur berat badan ternak sapi Timor berbasis citra sebagai pengganti timbangan mekanis dalam mendukung inovasi peternakan sapi di Pulau Timor Provinsi Nusa Tenggara Timur,” in Proc. Seminar Nasional dan Konferensi Sistem Informasi, Informatika dan Komunikasi (SEMMAU 2015), Kupang, Indonesia, 2015, pp. 129–134.

[13] Martinus, A. Susilo, M. Telaumbanua, and M. A. Muhammad, “Rancang bangun sistem penghitung jumlah dan massa biji kopi berbasis mikrokontroler pada konveyor sabuk,” Barometer, vol. 5, no. 2, pp. 267–271, 2020, doi: 10.35261/barometer.v5i2.3816.

[14] I. A. Hidayat, “Implementasi algoritma FIFO pada sistem manajemen stok dan deteksi kadaluwarsa otomatis di Toko Agus Mart,” S1 thesis, Universitas Muhammadiyah Magelang, Magelang, Indonesia, 2025.

JRAEE

Journal of Robotics, Automation, and Electronics Engineering
Vol. 4, No. 1, 2026, pp. 250-259
e-ISSN 3025-3780
https://journal.uny.ac.id/v3/jraee

Air Quality Classification Using the K-Nearest Neighbors Algorithm: A Case Study in Juwana District

Tanzila Bagus Ramadhan[a,1,*], Dessy Irmawati[b,2]

[a,b] Dept. of Electrical and Electronics Engineering, Vocational Faculty, Universitas Negeri Yogyakarta
[1]tanzilabagus.2021@student.uny.ac.id; [2]dessy.irmawati@uny.ac.id
[*]Corresponding Author

ARTICLE INFO

Article History
Received 12 Jan 2026
Revised 29 Aug 2026
Accepted 31 Aug 2026

Keywords
Air Quality Detector;
K-Nearest Neighbors;
Internet of Things;
Graphical User Interface;
Engineering Design Process.

ABSTRACT

Air quality is an important factor in maintaining human health and environmental sustainability. Increasing transportation activity in Pati Regency contributes to higher air-pollutant emissions. This study develops a real-time air-quality monitoring and classification system using the K-Nearest Neighbors (KNN) algorithm. The system uses MQ-7, MQ-135, and SDS011 sensors to measure CO, CO_2, PM2.5, and PM10 concentrations. Measurement data are transmitted to Firebase Realtime Database and classified through a graphical user interface using Euclidean distance. Experimental results show that the system achieves 83% accuracy, 85% precision, 83% recall, and an F1-score of 82%. These results indicate that the proposed system can provide reliable real-time air-quality information for environmental monitoring applications.

ABSTRAK

Kualitas udara merupakan faktor penting dalam menjaga kesehatan manusia dan keberlanjutan lingkungan. Peningkatan aktivitas transportasi di Kabupaten Pati turut berkontribusi terhadap meningkatnya emisi pencemar udara. Penelitian ini mengembangkan sistem pemantauan dan klasifikasi kualitas udara secara real-time menggunakan algoritma K-Nearest Neighbors (KNN). Sistem menggunakan sensor MQ-7, MQ-135, dan SDS011 untuk mengukur konsentrasi CO, CO_2, PM2.5, dan PM10. Data hasil pengukuran dikirim ke Firebase Realtime Database dan diklasifikasikan melalui antarmuka grafis menggunakan jarak Euclidean. Hasil pengujian menunjukkan bahwa sistem mencapai akurasi 83%, presisi 85%, recall 83%, dan F1-score 82%. Hasil tersebut menunjukkan bahwa sistem mampu memberikan informasi kualitas udara secara real-time untuk mendukung pemantauan lingkungan.

1. Introduction

Air quality is a crucial indicator for assessing environmental health and public well-being. In Indonesia, rising industrial, transportation, and urbanization activity has driven a marked increase in air-pollutant emissions. These pollutants are known to directly affect human health, particularly the respiratory system, and to reduce quality of life, especially in urban and industrial areas. Air quality in Indonesia has declined significantly, particularly in densely populated urban regions with heavy transportation activity, where the main source of particulate matter is motor-vehicle emissions, including combustion products and component friction.

Beyond carbon monoxide, fine particulates such as PM2.5 and PM10 are also a major concern in air pollution. PM2.5 and PM10 are microscopic particles with diameters below 2.5 micrometers (μm) and below 10 micrometers, respectively. According to the World Health Organization [1], the recommended limit for fine particulates (PM2.5) is 5 μg/m³ as an annual average and 15 μg/m³ as a 24-hour average, whereas the safety threshold applied by the Indonesian government is 50 μg/m³ for both the 24-hour and annual averages. For PM10, WHO recommends an annual limit of 15 μg/m³ and a daily limit of 45 μg/m³. Data from Indonesia's Meteorology, Climatology, and Geophysics Agency (BMKG) [2] indicate that average air quality in Indonesia frequently falls into the "Moderate" to "Unhealthy" category, particularly in densely populated areas such as Juwana District.

According to data published by the Central Bureau of Statistics (BPS) of Pati Regency, the population of Juwana District had reached 97,507 people by mid-2022 across an area of approximately 55.93 km² [3], making Juwana one of the most densely populated districts in Pati Regency, at 1,743 people per km². This rapid population growth has directly increased mobility needs, which in turn has driven significant growth in the number of motor vehicles. According to the Central Bureau of Statistics of Central Java Province, the number of motor vehicles in Pati Regency reached 718,265 units, consisting of 57,694 cars, 1,362 buses, 25,636 trucks, and 633,573 motorcycles [4]. This increase in motor vehicles has contributed substantially to air pollution through exhaust emissions containing hazardous pollutants.

Accordingly, this study classifies air quality in Juwana District to provide a clear picture of the region's air-quality conditions. It applies a data-mining approach, using the K-Nearest Neighbors (KNN) algorithm as the classification method for its demonstrated ability to achieve high accuracy even with a relatively small dataset [5]. KNN is an effective classification method for predicting air quality, particularly when the available data are limited, since it classifies new data based on their proximity to previously categorized data. The results of this classification are expected to serve as a basis for air-pollution mitigation policy recommendations for the Juwana District government.

2. Problem-Solving Approach

2.1 Air Quality Detector

An air quality detector is a technology-based device designed to monitor ambient air conditions by detecting and measuring the concentration of specific pollutants. The system operates by using various sensors to obtain air-quality data, which is then processed and presented informatively. An air quality detector is intended to provide early information about pollution levels, helping protect human health and supporting environmental control and management efforts. Technologies commonly applied in such systems include gas and particulate sensors, microcontrollers, the Internet of Things, and graphical user interfaces [6].

2.2 Internet of Things

The Internet of Things (IoT) is a technology paradigm that integrates physical objects with the internet, enabling those objects to automatically collect, exchange, and process data [7]. Devices within an IoT system are typically equipped with sensors, actuators, and communication capabilities that support real-time monitoring and control, supported by technologies such as cloud computing to turn the generated data into valuable information. As a result, IoT adoption supports improved efficiency, accuracy, and data-driven decision-making across a wide range of fields.

Some examples of IoT applications in ambient air monitoring are as follows.

- Air quality monitoring: IoT sensors can be installed in urban or industrial areas to monitor pollutant concentrations in real time.
- Early pollution warning: IoT technology enables systems to issue notifications or early warnings when pollutant concentrations exceed defined thresholds.
- Environmental management and evaluation: air-quality data collected by IoT sensors can be stored and

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

analyzed to evaluate pollution trends over time.

2.3 Graphical User Interface

A Graphical User Interface (GUI) is a graphics-based interface that enables interaction between users and a system through visual elements such as icons, buttons, menus, and other graphical displays. A GUI is designed to make a system easier to operate, present information intuitively, and improve the effectiveness and efficiency of monitoring and control processes. It also serves as the link between the data generated by a system and its users, so that information can be presented in real time in a form that is easy to understand and that supports decision-making.

The following are applications of a GUI in ambient air monitoring.

- Air-quality data monitoring: the GUI displays air-quality sensor readings in real time, allowing users to track conditions directly.
- Information visualization: the GUI presents air-quality data as charts, indicators, or tables so that the information is easier to understand and analyze.
- Hazard warnings: the GUI can display notifications or alerts when pollutant values exceed set thresholds, allowing users to take prompt action.
- Decision support: the information presented through the GUI helps users evaluate air-quality conditions and supports fast, well-informed decision-making.

3. Research Design

This study applies the Engineering Design Process (EDP), a systematic framework used to design and develop a product or system in a structured manner [8]. The EDP method was selected for its advantage in breaking each stage of the process down in greater detail, as shown in Fig. 1.

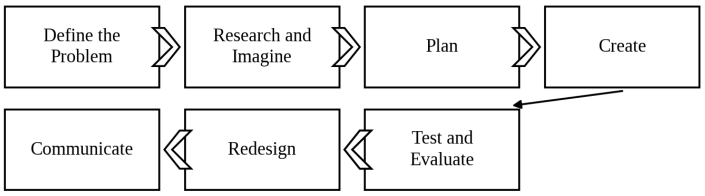


Fig.1 Engineering design process

3.1 Define the Problem

Define the Problem is the process of identifying and analyzing the problem underlying the study, namely the high level of air pollution in Juwana District. This condition creates a need for an accurate, reliable air-quality classification system that is readily accessible to the public and relevant stakeholders.

3.2 Research and Imagine

At this stage, the hardware, software, and supporting equipment used in the study were identified, as shown in Tables 1 and 2.

Table 1. Hardware List

No	Hardware	Function
1	ESP32 DevKIT V1	Used as the microcontroller.
2	MQ-7 Sensor	Selected to detect CO gas.
3	MQ-135 Sensor	Used to measure CO2.
4	SDS011 Sensor	Selected to detect dust-particle levels such as PM2.5 and PM10 [9].
5	Step-Down Module 5V	Needed to step down the voltage to match the operational requirements of the sensors and microcontroller.
6	TP4056 Module	Used as a practical and efficient battery-charging module.
7	Solder	Used to connect electronic components so they are properly joined.
8	Multimeter	Plays an important role in measuring voltage, current, and resistance in electronic components.
9	Mini Drill	Needed during assembly, since the system integrates several sensors that must be combined onto a single circuit board.
10	Laptop	Used as the center for development, programming, and data analysis.

Table 2. Software List

No	Software	Function
1	Tinkercad	Used for 3D modeling of the hardware box.
2	Fritzing	Used because it can visualize electronic circuits in both schematic and PCB-layout form.
3	Arduino IDE	Used to program the hardware.
4	JupyterLab	Used to train the KNN model.
5	Firebase	Used to store the data transmitted by the hardware.

3.3 Plan

At this stage, the system was planned comprehensively as the basis for the subsequent design process, including the preparation of a block diagram. Fig. 2 illustrates the block diagram of the air-quality classification system, which is divided into three parts: Hardware, Database, and Data Classification. In the hardware section, the system is designed to detect air-quality parameters using several sensors: the MQ-7 sensor for carbon monoxide (CO) levels, the MQ-135 sensor for other pollutant gases such as carbon dioxide (CO2), and the SDS011 sensor for measuring particulate concentrations such as PM2.5 and PM10, consistent with calibration and validation practice for low-cost gas and particulate sensors [10].

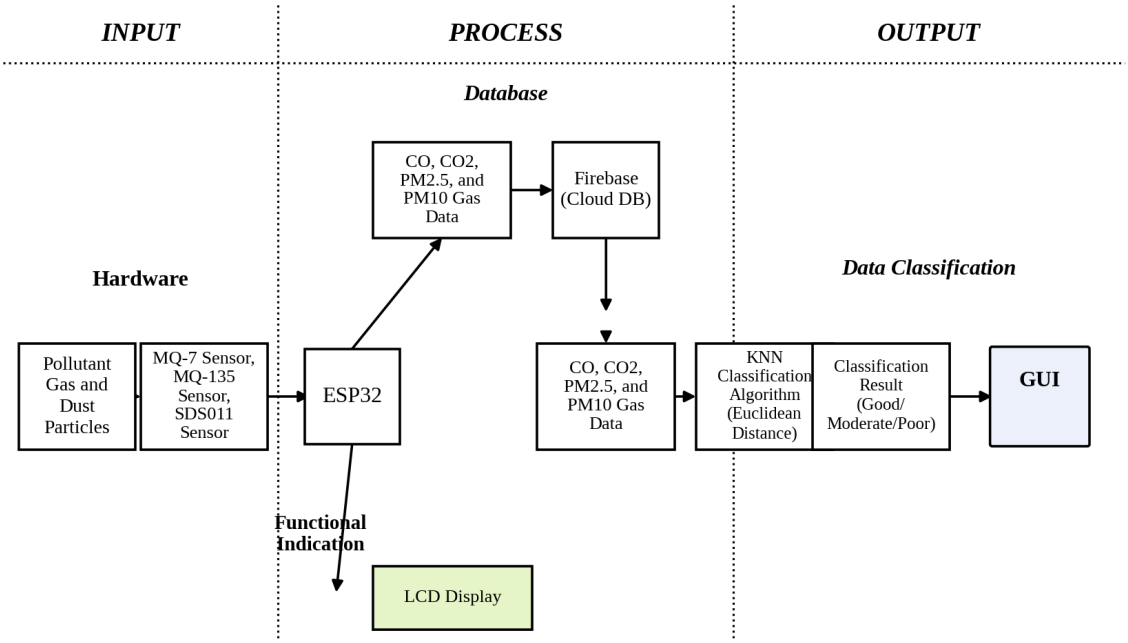


Fig. 2 Block diagram

In the database section, the data collected from these sensors is transmitted in real time to a cloud database platform (Firebase). Firebase is used to store air-quality parameter data, including CO, CO2, PM2.5, and PM10, in a structured format.

In the data-classification stage, the system retrieves the data stored in Firebase and processes it using the K-Nearest Neighbors (KNN) algorithm. The KNN algorithm works by comparing new input data against training data labeled "Good," "Moderate," and "Poor" based on nearest distance. These class labels are assigned according to previously defined air-quality thresholds, used as the criteria for categorizing air-quality conditions consistent with the applicable air-quality standard [11]. The air-quality threshold details for each category are presented in Table 3.

Table 3. Air Quality Threshold Indications

PM2.5 (µg/m³)	PM10 (µg/m³)	CO (PPM)	CO2 (ppm)	Status
0.0–12.0	0–54	0–25	0–700	Good
12.1–35.4	55–154	26–50	701–1000	Moderate
35.5–55.4	155–254	51–100	1001–1500	Poor

3.4 Create

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

At this stage, a prototype air-quality monitoring system was built by integrating the hardware and software. This prototype was constructed based on the design established during the planning stage, so that each selected component could be implemented according to its function. On the hardware side, the work involved assembling the circuit, fabricating and printing the PCB, mounting the components, and conducting initial testing to confirm the performance of each module. On the software side, the work involved writing the program code, implementing the classification algorithm, and integrating the system so that the hardware could communicate optimally with the GUI. The resulting prototype is intended to represent the system design in physical form and to be ready for testing in the next stage.

3.5 Test and Evaluate

This stage covers the testing and evaluation of the developed system prototype. Functional testing was carried out to confirm that each hardware component, software component, and the system as a whole operated according to the design. Hardware testing involved evaluating the sensors by comparing their readings against a reference instrument to confirm data accuracy, as well as testing the step-down module and the TP4056 module to confirm voltage stability and charging function. Software testing covered the software functions and the graphical user interface (GUI) to confirm that data processing, storage, and visualization operated correctly. System testing, meanwhile, evaluated the performance of the KNN algorithm in classifying air quality using the study's dataset, comprising a total of 137 records split into 70% training data and 30% test data. To obtain optimal model performance, hyperparameter tuning was carried out using the GridSearchCV method to determine the best parameter values [12]. In addition, the model was evaluated using k-fold cross-validation to improve generalization and minimize overfitting to the training data [13].

3.6 Communicate

This stage covers the complete system design, development process, and research findings, communicated clearly and systematically to relevant stakeholders through written reports, presentations, or scientific publications.

4. Results and Discussion

4.1 Create Stage Implementation

Hardware integration was carried out by placing the electronic components inside a 3D-printed casing. Assembly began with mounting the PCB as the base of the circuit, followed by connecting the components neatly and systematically. The component layout was arranged to be stable, safe, and easy to maintain, while each connection was positioned so as not to interfere with the others, making efficient use of the available space inside the box. The resulting hardware box is shown in Fig. 3.

Fig. 3 Hardware box display

To support user interaction, a Graphical User Interface was developed using libraries such as Tkinter or PyQt. This GUI displays air-quality sensor values in real time, covering the CO, CO2, PM2.5, and PM10 parameters. The air-quality classification result is displayed visually as text or a color indicator denoting the category (good, moderate, or poor). The GUI also provides a classification-history feature, displayed as a table, which can be exported to CSV format for further analysis or documentation. The GUI display is shown in Fig. 4.

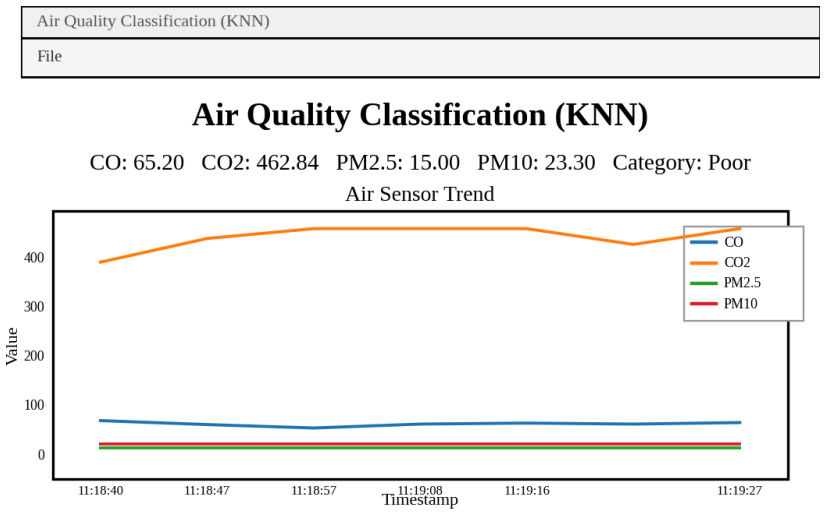


Fig. 4 GUI display

4.2 Test and Evaluate Stage Implementation

Table 4 presents the results of training the KNN model using 137 training records labeled by air-quality category (Good, Moderate, and Poor), containing the CO, CO2, PM2.5, and PM10 parameters. Before use, the data was normalized with a Standard Scaler to equalize the scale across features, improving classification accuracy. During KNN classification testing, the value of the parameter k was obtained by searching across the range k = 1–20 using a grid-search method, with k = 3 producing the highest accuracy. In addition, model validation was performed using 5-fold cross-validation so that the resulting classification was more stable and not dependent on a single data subset; k = 3 means the system determines a class based on the three nearest neighbors. Of the 137 records available, 96 were used as training data and 41 as test data, consistent with the 70:30 training–testing split. Evaluation showed an accuracy of 83%; a precision of 85%; a recall of 83%; and an f1-score of 82%.

Table 4. Classification Report (KNN)

	precision	*recall*	*f1-score*	*support*
Good	0.78	0.88	0.82	8
Moderate	0.81	0.92	0.86	24
Poor	1.00	0.56	0.71	9
accuracy			0.83	41
macro avg	0.86	0.78	0.80	41
weighted avg	0.85	0.83	0.82	41

The Confusion Matrix in Fig. 5 was used to evaluate the types of errors made by the model, showing the number of correct and incorrect predictions for each category – for example, whether the model more frequently misclassified air quality as “Good,” “Moderate,” or “Poor.”

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

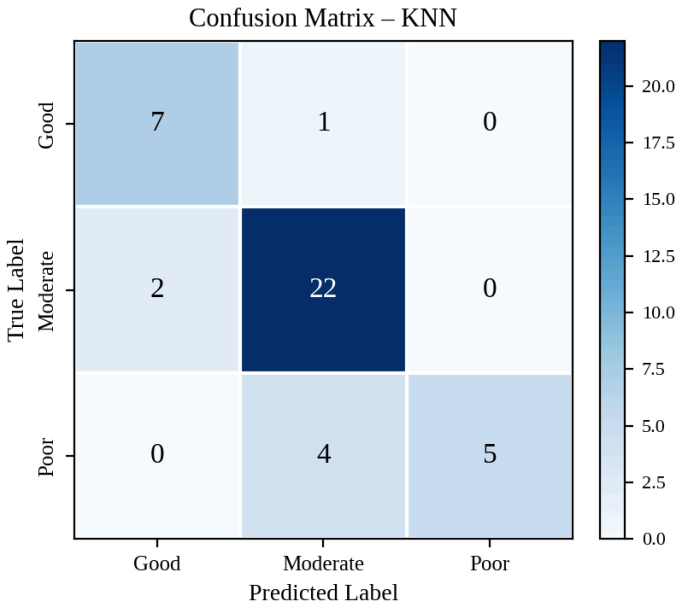


Fig. 5 Confusion matrix

- 7 records that were actually “Good” were correctly predicted as “Good” (True Positive for the “Good” class).
- 2 records that were actually “Moderate” were incorrectly predicted as “Good” (False Negative for the “Moderate” class).
- Only 1 record that was actually “Good” was incorrectly predicted as “Moderate” (False Positive for the “Good” class).
- 4 records that were actually “Poor” were incorrectly predicted as “Moderate” (False Positive for the “Moderate” class).
- 22 records that were actually “Moderate” were correctly predicted as “Moderate” (True Positive for the “Moderate” class).
- 5 records that were actually “Poor” were correctly predicted as “Poor” (True Positive for the “Poor” class).

The results above show that the KNN model performs reasonably well, particularly for the “Moderate” and “Good” classes, as reflected in the high True Positive counts and very low False Negative counts. For the “Poor” class, however, some misclassification remains: 4 records that should have been “Poor” were instead predicted as “Moderate.” This is attributable to similarities in data characteristics, particularly given the imbalanced distribution of the data, indicating that although the model is fairly accurate overall, its performance on the minority class still has room for improvement.

Fig. 6 shows the Receiver Operating Characteristic curve for the KNN model in classifying air quality into three categories: “Good,” “Moderate,” and “Poor.” The test results show that the “Good” class achieved an AUC value of 0.972, indicating very strong performance, as the model was able to distinguish this class with a high degree of accuracy. The “Moderate” class achieved an AUC value of 0.860, while the “Poor” class achieved an AUC of 0.854. The resulting weighted-average AUC was 0.881, placing the classification accuracy of this study within the “Good Classification” category [14].

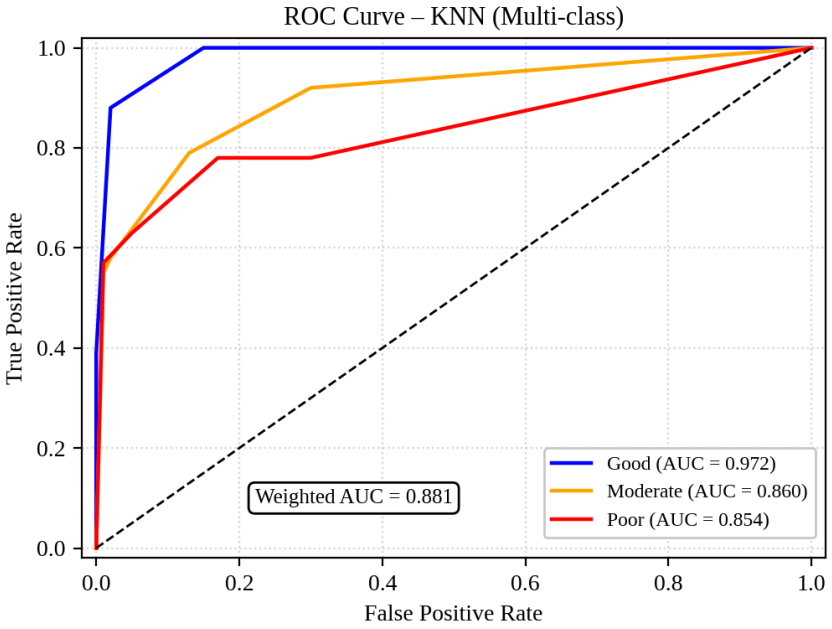


Fig. 6 Receiver operating characteristic curve

4.3 Redesign Stage Implementation

At the redesign stage, an additional 3D-printed component was added to cover the edges of the openings in the hardware enclosure. This addition was intended to improve the neatness and aesthetics of the device and to protect its internal components.

4.4 Communicate Stage Implementation

At this stage, the results of the device implementation were reported comprehensively. Testing was carried out by placing the device at four different locations in Juwana District with contrasting environmental characteristics. The sensor readings were then processed by the classification system to determine the air-quality category at each location – "Good," "Moderate," or "Poor" – based on the previously defined gas and particulate concentration thresholds shown in Table 3. The classification history, presented in tabular form, can be downloaded in CSV format for further analysis or as documentation of the test results.

The first test was conducted in the Alun-Alun Juwana (town square) area. Based on the sample data obtained, shown in Table 5, the classification results indicate that the "Moderate" category was dominant, with PM2.5 the most influential parameter in the classification outcome.

Table 5. Classification Results at Location 1 (Alun-Alun Juwana)

Timestamp	CO	CO2	PM2.5	PM10	Category
08:37:24	58.24837	505.78915	22.1	36.9	Moderate
08:37:33	42.73223	544.86188	22.7	37.3	Moderate
08:55:26	32.14607	558.32928	22.4	37.9	Moderate
08:57:26	25.66893	551.56757	16.2	19.2	Moderate
08:58:46	37.92812	592.98737	17.8	20.3	Moderate
08:59:46	22.89186	565.14728	17.0	21.9	Moderate
09:00:46	22.89186	538.21191	16.8	23.0	Moderate
09:01:46	24.98453	505.78915	21.7	65.0	Moderate
09:02:46	22.83829	551.56757	21.0	52.3	Moderate

The second test was conducted in front of SMK BTB Juwana. Based on the sample data obtained, shown in Table 6, the classification results indicate that the "Moderate" category was dominant, with PM2.5 the most critical parameter influencing the classification outcome.

Table 6. Classification Results at Location 2 (SMK BTB Juwana)

Timestamp	CO	CO2	PM2.5	PM10	Category
09:57:19	13.91927	551.56757	21.6	28.6	Moderate
09:58:29	13.24391	544.86188	21.0	26.9	Moderate
09:59:39	13.71874	531.61743	20.4	25.5	Moderate
10:01:49	13.79876	544.86188	20.8	25.7	Moderate

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

10:02:19	13.51979	505.78915	21.3	27.5	Moderate
10:03:29	13.55945	525.07825	20.9	26.7	Moderate
10:04:39	13.32242	512.16437	20.7	28.4	Moderate
10:05:49	13.40118	499.46823	20.2	27.9	Moderate

The third test was conducted in the Pasar Juwana (market) area, a location with substantial pedestrian and vehicle activity. Based on the sample data obtained, shown in Table 7, the classification results indicate that the "Moderate" category was dominant, with PM2.5 the most critical parameter influencing the classification outcome.

Table 7. Classification Results at Location 3 (Pasar Juwana)

Timestamp	CO	CO2	PM2.5	PM10	Category
09:53:59	20.10632	592.98737	15.4	24.9	Moderate
09:54:09	18.22607	585.94171	15.8	24.3	Moderate
09:55:19	20.35523	544.86188	15.3	23.5	Moderate
09:56:29	21.21512	614.47131	16.2	24.4	Moderate
09:57:39	19.85921	578.95325	17.1	29.1	Moderate
09:58:49	17.21230	565.14728	16.8	28.6	Moderate
09:59:59	17.39415	585.94171	15.5	25.3	Moderate
10:00:09	18.41399	578.95325	14.7	22.8	Moderate
10:01:19	16.58437	551.56757	15.4	20.9	Moderate

The fourth test was conducted along Jalan Toko Horizcom, Juwana, a location with lighter vehicle traffic but still containing a substantial amount of airborne dust. Based on the sample data obtained, shown in Table 8, the classification results indicate that the "Moderate" category was dominant, with PM2.5 the most critical parameter influencing the classification outcome.

Table 8. Classification Results at Location 4 (Jalan Toko Horizcom, Juwana)

Timestamp	CO	CO2	PM2.5	PM10	Category
09:33:03	5.45842	439.18259	13.8	15.9	Moderate
09:34:13	7.44745	444.97568	13.8	18.2	Moderate
09:34:23	6.38359	450.82056	13.7	17.2	Moderate
09:35:33	6.53985	486.98785	13.4	16.3	Moderate
09:36:43	5.49009	480.82803	13.5	17.4	Moderate
09:37:53	5.01033	474.72113	13.1	17.8	Moderate
09:38:03	4.99525	499.46823	12.8	17.5	Moderate
09:39:13	4.92020	505.78915	13.8	20.0	Moderate
09:40:23	5.47424	538.21191	14.9	21.1	Moderate

5. Conclusion

The results of this study show that the air-quality monitoring system was successfully designed and implemented through the integration of sensor hardware, a Firebase Realtime Database for data storage and synchronization, and a Graphical User Interface for visualization and user interaction. This integration enables the acquisition, transmission, storage, and display of air-quality data to occur in real time and in a centralized manner, allowing the system to operate stably and in accordance with its intended design.

In terms of data processing, the KNN algorithm was able to classify air quality with an accuracy of 83% on the test data. Evaluation of model performance using weighted-average metrics showed a precision of 85%, a recall of 83%, and an f1-score of 82%. These values indicate that the model achieves a good balance between its ability to recognize each class and the accuracy of its classification results, and demonstrate consistent performance in classifying air-quality categories in Juwana District based on the parameters used.

Author Contribution:

All authors contributed equally as the main contributors to this paper. All authors read and approved the final manuscript.

Funding:

This research received no external funding.

Acknowledgement:

The authors thank the Department of Electrical and Electronics Engineering, Vocational Faculty, Universitas Negeri Yogyakarta, for providing the facilities used in this research.

Conflicts of Interest:

The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

[1] World Health Organization, WHO Global Air Quality Guidelines: Particulate Matter (PM2.5 and PM10), Ozone, Nitrogen Dioxide, Sulfur Dioxide and Carbon Monoxide. Geneva: World Health Organization, 2021.

[2] Badan Meteorologi, Klimatologi, dan Geofisika (BMKG), "Laporan Kualitas Udara Indonesia," 2025.

[3] Badan Pusat Statistik Kabupaten Pati, "Kabupaten Pati dalam Angka 2022," Badan Pusat Statistik Kabupaten Pati, 2022.

[4] Badan Pusat Statistik Provinsi Jawa Tengah, "Jumlah Kendaraan pada Kabupaten Pati dalam Angka 2021," Badan Pusat Statistik Provinsi Jawa Tengah, 2021.

[5] R. K. Halder, M. N. Uddin, M. A. Uddin, S. Aryal, and A. Khraisat, "Enhancing K-Nearest Neighbor Algorithm: A Comprehensive Review and Performance Analysis of Modifications," Journal of Big Data, vol. 11, no. 1, Art. no. 113, 2024, doi: 10.1186/s40537-024-00973-y.

[6] Z. Ahmad and S. Hussain, "Real-Time Monitoring of Air Pollution Using IoT and Machine Learning," Journal of Environmental Informatics, vol. 10, no. 1, pp. 15–28, 2024.

[7] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," Computer Networks, vol. 54, no. 15, pp. 2787–2805, 2010, doi: 10.1016/j.comnet.2010.05.010.

[8] S. N. Aini, D. N. Munahefi, A. S. Pramasdyahsari, and R. D. Setyowati, "Engineering Design Process STEM: Projek Miniatur Gazebo Joglo," in PRISMA, Prosiding Seminar Nasional Matematika, 2024, pp. 37–43.

[9] H.-Y. Liu, P. Schneider, R. Haugen, and M. Vogt, "Performance Assessment of a Low-Cost PM2.5 Sensor for a near Four-Month Period in Oslo, Norway," Atmosphere, vol. 10, no. 2, Art. no. 41, 2019, doi: 10.3390/atmos10020041.

[10] D. Margaritis, C. Keramydas, I. Papachristos, and D. Lambropoulou, "Calibration of Low-Cost Gas Sensors for Air Quality Monitoring," Aerosol and Air Quality Research, vol. 21, no. 11, Art. no. 210073, 2021, doi: 10.4209/aaqr.210073.

[11] M. Erik, F. Nurdiyanto, and R. Hidayat, "Aerosense Monitor: Integrasi Sensor DHT11 dan MQ135 untuk Pemantauan Kualitas Udara Berbasis Arduino Uno," Jurnal Komputer dan Elektro Sains, vol. 2, no. 2, pp. 8–11, 2024.

[12] M. R. Siregar, D. Hartama, S. Solikhun, et al., "Optimizing the KNN Algorithm for Classifying Chronic Kidney Disease Using GridSearchCV," JITK (Jurnal Ilmu Pengetahuan dan Teknologi Komputer), vol. 10, no. 3, pp. 680–689, 2025.

[13] A. Maulana, A. I. Purnamasari, and I. Ali, "Analisis Klasifikasi Indeks Kualitas Udara Kota di Indonesia Menggunakan Metode K-Nearest Neighbor dan Naïve Bayes," JATI (Jurnal Mahasiswa Teknik Informatika), vol. 8, no. 1, pp. 215–222, 2024.

[14] P. Astuti, N. Nuris, et al., "Penerapan Algoritma KNN pada Analisis Sentimen Review Aplikasi Peduli Lindungi," Computer Science (Co-Science), vol. 2, no. 2, pp. 137–142, 2022.

JRAEE

Journal of Robotics, Automation, and Electronics Engineering
Vol. 4, No. 1, 2026, pp. 260-267
e-ISSN 3025-3780
https://journal.uny.ac.id/v3/jraee

Design And Development Of A Microcontroller-Based Air Pollution Monitoring System At Traffic Light Areas Using The Mamdani Fuzzy Logic Method

Pangeran Antonius[a,1,*], Arya Sony[b,2,*]

[a,b]Dept. of Electrical and Electronics Engineering, Vocational Faculty, Universitas Negeri Yogyakarta
[1]pangeranantonius.2021@student.uny.ac.id; [2]arya.sony@uny.ac.id
[*]Corresponding Author

ARTICLE INFO

Article History
Received 22 Apr 2026
Revised 29 Aug 2026
Accepted 31 Aug 2026

ABSTRACT

Air pollution was a serious issue in Indonesia, particularly in areas with high vehicle density, such as Yogyakarta. In 2024, more than 164 million vehicles were recorded in Indonesia. Motor vehicle emissions contributed to approximately 85% of total air pollution in the country. The high activity of motor vehicles increased the concentration of harmful gases such as carbon monoxide (CO), which had adverse effects on public health. To assess air pollution levels in surrounding areas, a monitoring system of air quality was required to maintain acceptable environmental conditions. In this study, a system was designed by integrating the MQ-135 sensor to detect carbon monoxide (CO) in real-time. The data obtained from the sensor were transmitted to the ESP32, where data processing employed the Mamdani Fuzzy Logic method to classify air quality into three categories: Good, Moderate, and Poor. The process began with CO level detection and was followed by fuzzification, inference, and defuzzification stages, producing a crisp value as a reference for air quality classification. In addition, the system was equipped with a monitoring feature based on Firebase and an application developed with MIT App Inventor. Based on the results of testing, the air pollution monitoring system was successfully implemented. The MQ-135 sensor was able to read surrounding air conditions in real-time with an average error of 0.33%, resulting in a reading accuracy level of 99.67%. The Mamdani Fuzzy Logic method effectively processed ambiguous data; however, testing also identified limitations in the design of membership functions, which led to mismatches in category determination at threshold values. The system successfully transmitted data in real-time to Firebase and the MIT App Inventor application with a success rate of 80%.

Keywords
Air pollution;
ESP32;
Mamdani Fuzzy Logic;
Firebase.

ABSTRAK

Polusi udara di wilayah perkotaan, yang sebagian besar disebabkan oleh emisi kendaraan, merupakan ancaman signifikan bagi kesehatan masyarakat. Penelitian ini menjawab kebutuhan akan sistem pemantauan kualitas udara yang dapat diakses secara *real-time*, khususnya untuk area padat lalu lintas seperti persimpangan. Kontribusi utama dari penelitian ini adalah

pengembangan prototipe berbasis *Internet of Things* (IoT) yang hemat biaya dan memanfaatkan metode Logika Fuzzy Mamdani untuk mengklasifikasikan kualitas udara, sehingga data lebih mudah dipahami oleh masyarakat umum. Sistem ini dirancang menggunakan sensor MQ-135 untuk mendeteksi kadar Karbon Monoksida (CO), mikrokontroler ESP32 untuk pemrosesan data, dan modul SIM800L untuk transmisi data *real-time* ke platform Firebase. Metode Logika Fuzzy Mamdani diimplementasikan untuk mengklasifikasikan konsentrasi CO yang terukur ke dalam tiga kategori: Baik, Sedang, dan Buruk. Hasil implementasi menunjukkan kinerja yang sangat positif. Sensor MQ-135 menunjukkan akurasi yang tinggi, dengan tingkat kesalahan rata-rata hanya 0.33% dan akurasi pembacaan keseluruhan sebesar 99.67% dibandingkan dengan detektor CO standar. Meskipun sistem logika fuzzy secara fungsional berhasil mengolah data ambigu di batas kategori, pengujian juga mengungkapkan adanya keterbatasan desain yang menyebabkan klasifikasi untuk nilai ambang batas tertentu (misalnya, 100 ppm) tidak selaras dengan standar ISPU. Lebih lanjut, sistem berhasil mentransmisikan data ke platform *cloud* Firebase dan aplikasi kustom MIT App Inventor dengan tingkat keberhasilan 80%. Sebagai kesimpulan, penelitian ini berhasil mengembangkan prototipe IoT yang fungsional untuk pemantauan polusi udara. Studi ini mengkonfirmasi kelayakan penggunaan metode Logika Fuzzy Mamdani untuk klasifikasi kualitas udara *real-time*, meskipun menyoroti kebutuhan krusial akan desain fungsi yang presisi untuk memastikan keselarasan dengan standar yang ada.

1. Introduction

Air pollution is a significant environmental challenge in Indonesia, which ranks as the third-largest user of motorized vehicles in Asia. The primary sources of this pollution are the high volume of vehicles exceeding 164 million units, with motorcycles comprising 83% and emissions from numerous industrial factories [1]. Consequently, vehicle emissions are identified as the leading cause of air pollution in Indonesia, contributing to approximately 85% of the total pollutants [2]. In addition to these sources, air pollution is exacerbated by common practices such as open waste burning. The resulting pollutants include particulate matter (e.g., dust and aerosols) and hazardous gases such as Carbon Monoxide (CO), Nitrogen Oxides (NOx), Sulfur Dioxide (SOx), and Hydrogen Sulfide (H2S) [3].

The consequences of this deteriorating air quality are severe for both human health and the environment. Prolonged exposure is linked to a range of conditions, including respiratory disorders, heart disease, and premature death. A 2021 study by the Health Effects Institute (HEI) attributed over 221,600 deaths in Indonesia to air pollution, with Acute Respiratory Infections (ARIs) being a leading cause of morbidity worldwide [4]. Beyond direct health impacts, air pollution alters the Earth's atmospheric structure, allowing harmful UV radiation to increase health risks like skin cancer [5]. Furthermore, pollutants trap heat in a phenomenon known as the greenhouse effect, which contributes to the rise in global temperatures [6].

In response to this crisis, the Indonesian government has established ambient air quality standards through Government Regulation No. 22 of 2021, which sets a threshold for pollutants like Carbon Monoxide at 10,000 µg/m³ over an 8-hour period to safeguard public health. Despite these official regulations and the clear dangers, public awareness of the severe health and environmental risks remains critically low. This gap between established standards and public understanding highlights the need for accessible information to motivate community-level action.

To address this awareness gap, this project proposes the development of an Internet of Things (IoT) based air quality monitoring device. The system is designed to provide real-time, easily understandable data using an MQ-135 sensor for pollution detection, an ESP32 microcontroller for data processing, and a SIM800L

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

module for remote communication [7]. Data is transmitted wirelessly to a Firebase database, ensuring that accurate and timely air quality information can be accessed. This technological solution aims to increase public consciousness and provide a practical tool for monitoring environmental health.

2. The Proposed Method/Algorithm

The core of the data processing in this system is a multi-stage fuzzy inference algorithm designed to convert quantitative CO concentration values (in PPM) into a qualitative assessment of air quality. This method ensures a more human-like and flexible decision-making process by handling imprecise data. The overall process, as illustrated in Figure 1, consists of four main stages: data collection, fuzzy set formation, fuzzy rule evaluation, and defuzzification.

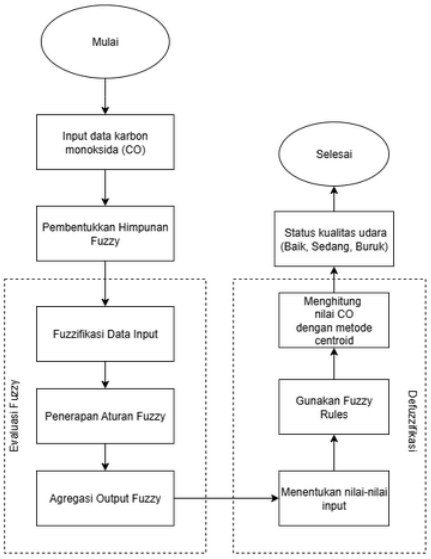


Fig. 1 Fuzzy logic mamdani method flowchart

The membership functions for each set are defined as follows:

- 'BAIK' (Good) Membership Function:

  $\mu Baik(x)\{1 \;;\; x \leq 30 \; \frac{50-x}{20} \;;\; 30 < x < 50 \; 0 \;;\; x \geq 50$

- 'SEDANG' (Moderate) Membership Function:

  $\mu Sedang(x) = \{0 \;;\; x \leq 40 \; or \; x \geq 100 \; \frac{x-40}{20}; 40 < x \leq 60 \; 1; 60 < x \leq 80 \; \frac{100-x}{20}; 80 < x < 100$

- 'BURUK' (Poor) Membership Function

  $\mu Buruk(x)\{0; x \leq 90 \; \frac{x-90}{30}; 90 < x < 120 \; 1; x \geq 90$

The process is divided into three main operational stages:

a. Fuzzification: This is the initial stage where the crisp input value from the MQ-135 sensor (e.g., 42 PPM) is converted into degrees of membership (μ) across predefined linguistic sets ('BAIK', 'SEDANG', 'BURUK'). This is achieved using trapezoidal and triangular membership functions, as defined in the system's design and shown in the equations below. For instance, an input of 42 PPM might result in μ_BAIK = 0.4 and μ_SEDANG = 0.1.
b. Inference (Aggregation): In this stage, the fuzzy rules are evaluated. The system uses a set of "IF-THEN" rules (e.g., "IF CO is SEDANG THEN air quality is SEDANG"). The degree of membership from the fuzzification stage determines the "strength" of each activated rule. The aggregation process then combines the outputs of all activated rules into a single fuzzy set. This is done using the MAX-MIN method, where the final composite area is the maximum envelope of all individual rule outputs that were clipped (MIN) by their activation strength.
c. Defuzzification: This is the final stage where the composite fuzzy set from the aggregation step is converted back into a single crisp numerical value. This research uses the Centroid method, which calculates the center of gravity of the composite area. This crisp value represents the final, decisive assessment of the air quality, which is then used to determine the final output category ('BAIK',

'SEDANG', or 'BURUK').

3. Method

The research methodology followed the Engineering Design Process (EDP), which includes defining the problem, designing a solution, building a prototype, and testing. The system's architecture is designed as an integrated unit for real-time pollution monitoring.

3.1 System Design and Hardware

The system comprises input, processing, and output components integrated into a cohesive unit.

- Input: An MQ-135 gas sensor serves as the primary input, detecting the concentration of CO in the ambient air.
- Processing: An ESP32 microcontroller with integrated Wi-Fi acts as the central processing unit, running the fuzzy logic algorithm and managing all system operations.
- Output: The system provides user feedback through a 16x2 LCD for displaying PPM values and status, and a buzzer for alerts when pollution levels are high.
- Communication: A SIM800L module is used for cellular data transmission to the cloud database, ensuring connectivity even without local Wi-Fi.
- Power: The system is powered by 18650 batteries regulated by an LM2596 step-down module to ensure stable voltage for all components.

3.2 Software and Model Development

- Firmware: The ESP32 was programmed using the Arduino IDE with C++. This includes the code for sensor reading, implementing the Fuzzy Logic Mamdani algorithm, and data transmission protocols.
- Data Management: The system utilizes a cloud-based Firebase Realtime Database to store and synchronize data instantly. This allows for remote access and real-time updates.
- Monitoring Application: A mobile application was developed using MIT App Inventor. This application connects to the Firebase database to display real-time air quality data (PPM and status) on an Android device, providing a user-friendly interface for monitoring.

4. Result and Discussion

4.1 Hardware Implementation and System Design

The system was physically realized in a custom enclosure designed for practical deployment. The design utilized a durable ABS plastic box with dimensions of 18 cm (length) x 11 cm (width) x 6 cm (height), which was manually customized to house all components. The front panel integrates the primary user interface and sensing components: a 16x2 LCD display, the MQ-135 gas sensor, and a buzzer for alerts. A dedicated compartment was fashioned on the side of the enclosure to hold the 18650 batteries, ensuring easy access for replacement. Internally, a custom PCB was designed and manually etched to neatly manage the connections for the ESP32 microcontroller, the SIM800L module, and the power regulation circuit. Subsequent hardware tests confirmed that all components—the ESP32 microcontroller, MQ-135 sensor, LCD display, buzzer, and SIM800L module—functioned correctly and reliably under operational loads.

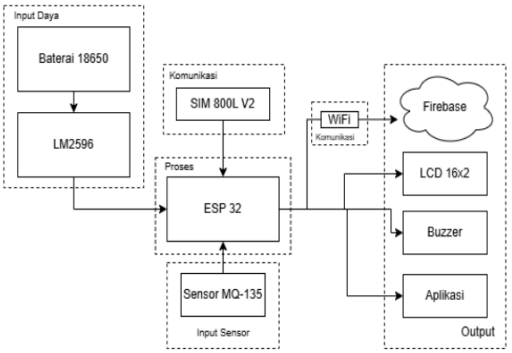


Fig. 2 Electronic architecture system

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

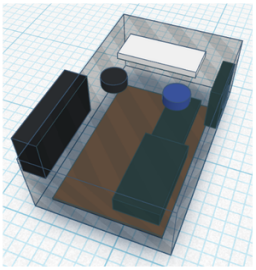


Fig. 3 Mechanics design

4.2 System Performance

The fully integrated system was tested in a real-world scenario at a busy traffic light intersection to evaluate its end-to-end performance. To determine the system's final accuracy, the error percentage was calculated for each set of test data by comparing its readings against a standard CO detector. This analysis confirmed the robustness of the hardware implementation, showing that the MQ-135 sensor demonstrated high accuracy, achieving an overall system accuracy of 99.67% with a mean error of only 0.33%. This result validates that the sensor calibration and data acquisition processes are reliable. The overall system accuracy was derived using the following formulas:

$$Error(\%)\frac{Device\ Reading - Standard\ Detector\ Reading}{Standard\ Detector\ Reading} \times 100\%$$

$$Total\ Error: 0.09\% + 0.04\% + 0.02\% + 0.03\% + 0.03\% + 0.02\% + 0.01\% + 0.01\% + 0.07\% = 0.33\%$$

$$Accuracy = 100\% - Mean\ Error$$

$$100\% - 0.33\% = 99.67\%$$

Table 1. System Performance Result

No.	CO Concentration (ppm)	CO Detector Reading (ppm)	Condition	Air Quality	LCD Display and Buzzer
1.	12	11	Green light, few vehicles	Good	Correct and Off
2.	24	25	Green light, few vehicles	Good	Correct and Off
3.	45	46	Green light, few vehicles	Good	Correct and Off
4.	53	55	Yellow light, vehicles begin to accumulate	Moderate	Correct and Off
5.	66	64	Yellow light, vehicles begin to accumulate	Moderate	Correct and Off
6.	77	79	Yellow light, vehicles begin to accumulate	Moderate	Correct and Off
7.	130	128	Red light, high vehicle density	Poor	Correct and On
8.	155	157	Red light, high vehicle density	Poor	Correct and On
9.	169	171	Red light, high vehicle density	Poor	Correct and On
10.	15	14	Green light again, few vehicles	Good	Correct and Off

4.3 Fuzzy Logic Performance

The performance of the Fuzzy Logic Mamdani method was a key focus of the evaluation. The basis for this logic is a set of membership functions designed for each category, as depicted in Fig.4. This graph illustrates how each PPM value is mapped to a degree of membership for the 'BAIK', 'SEDANG', and 'BURUK' sets.

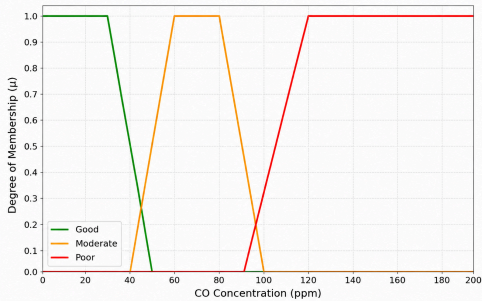


Fig. 4 Fuzzy membership graph

Functionally, the system was successful in processing ambiguous input values and producing a consistent output, as demonstrated in the detailed analysis of 42 PPM and 92 PPM inputs. However, the testing revealed a significant design flaw. A discrepancy was found between the system's output and the official Indonesian Air Pollution Standard Index (ISPU) at critical thresholds. For example, the system incorrectly classified a CO concentration of 100 PPM as 'BURUK', whereas the ISPU standard categorizes it as 'SEDANG'. This classification failure stems from a membership function design that is not perfectly aligned with the official standards.

The calculation follows three main stages of the Mamdani fuzzy inference system:

a. Fuzzification

The crisp input value (42 PPM) is converted into degrees of membership for the relevant fuzzy sets ('BAIK' and 'SEDANG') using the predefined membership functions:

- **μ_BAIK(x)** : For a CO level of 42 PPM, the degree of membership in the 'BAIK' set is calculated as:
  $\mu BAIK(42) \frac{50-42}{50-30} = \frac{8}{20} = 0.4$
- **μ_SEDANG(x)** : The degree of membership in the 'SEDANG' set is:
  $\mu SEDANG(42) = \frac{42-40}{60-40} = \frac{2}{20} = 0.1$
- **μ_BURUK(x)** : The degree of membership is **0** as 42 PPM is outside its range.

b. Inference and Aggregation

Based on the fuzzification results, two fuzzy rules are activated: "IF CO is BAIK THEN..." and "IF CO is SEDANG THEN...". The outputs of these rules are clipped at their respective degrees of membership (0.4 for 'BAIK' and 0.1 for 'SEDANG'). These clipped fuzzy sets are then combined (aggregated) using the MAX operator to form a single composite output fuzzy set. In this stage, a set of "IF-THEN" fuzzy rules are evaluated. The rules for this system are:

- IF CO concentration is 'BAIK', THEN air quality is 'BAIK'.
- IF CO concentration is 'SEDANG', THEN air quality is 'SEDANG'.
- IF CO concentration is 'BURUK', THEN air quality is 'BURUK'.

c. Defuzzification (Centroid Method)

The final step is to convert the composite fuzzy set back into a single crisp value using the Centroid method, which calculates the center of gravity of the composite area. The formula is:

$$x^* \frac{\sum xi \cdot \mu(xi)}{\sum \mu(xi)}$$

The calculation for the 42 PPM input yields:

- Numerator (sum of weighted values): 69.75
- Denominator (sum of membership values): 1.75

$$x^* \frac{69.75}{1.75} = 39.86$$

The resulting crisp value is 39.86 PPM. Since this value falls within the range defined for the 'BAIK' category (1-50), the system's final classification is 'BAIK'. A similar process for an input of 92 PPM (on the threshold of 'SEDANG' and 'BURUK') results in a crisp output of 89.56, correctly classifying it as 'SEDANG'. This demonstrates the method's robustness in handling ambiguous data through its internal fuzzy processing, ultimately producing a decisive and logical classification.

Table 2. Fuzzy Logic Method Result

No.	CO Concentration (ppm)	Fuzzy Output	Standard (Ditjen PPU)
1	12	Good	Good
2	24	Good	Good
3	45	Good	Good
4	50	Moderate	Good
5	66	Moderate	Moderate
6	77	Moderate	Moderate
7	100	Poor	Moderate
8	155	Poor	Poor
9	169	Poor	Poor
10	15	Good	Good

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

4.4 Real-Time Data Transmission Testing

The system's IoT functionality was tested by transmitting data from the device to the Firebase database and the monitoring application. The system achieved an 80% success rate in data transmission over 10 trials. The two failures were attributed to temporary cellular signal disruptions at the test location, indicating that the transmission hardware and software are functional but dependent on network stability.

Table 3. Real-Time Data Transmission Result

No.	CO Concentration (ppm)	Transmission Status	Data Received in App
1	12	Success	12
2	24	Success	24
3	45	Success	45
4	53	Success	53
5	66	Fail	65
6	77	Success	77
7	130	Success	130
8	155	Success	155
9	169	Success	169
10	15	Fail	14

4.5 Discussion

The results demonstrate the successful development of a highly accurate sensor system and a functional IoT framework. The 99.67% accuracy of the sensor readings is a strong indicator of the system's capability for reliable environmental monitoring. The 80% data transmission success rate is acceptable for a prototype and highlights the importance of stable network connectivity for IoT devices.

The most significant finding is the mixed performance of the fuzzy logic implementation. While the algorithm worked correctly from a mechanical standpoint (handling ambiguous data through aggregation and defuzzification), the design of its core logic (the membership functions) was flawed. This highlights a critical lesson in designing expert systems: the underlying logic must be meticulously validated against established standards to ensure the output is not just consistent but also correct. By successfully automating the process of air quality sensing and data logging, the system addresses the core goal of providing real-time information, but the classification inaccuracy remains a key limitation.

5. Conclusion

Based on the research and implementation conducted, it can be concluded that the design of an air pollution monitoring system using the Fuzzy Logic Mamdani method has been successfully realized by effectively integrating hardware and software components. The system demonstrated strong performance, with the MQ-135 sensor achieving a high accuracy rate of 99.67% in real-time air quality readings. Furthermore, the system's performance in data transmission was proven to be functional and reliable; it successfully sent air quality data to the Firebase platform and monitoring application with an 80% success rate, ensuring data accessibility for users. The results achieved are consistent with the objectives stated in the introduction, where this system was proposed as a solution to provide real-time air quality information. The successful implementation provides an automated and integrated framework for environmental monitoring. The Fuzzy Logic Mamdani method proved to be functionally robust in processing ambiguous data at category boundaries, such as for 42 and 92 PPM values. However, a critical flaw was identified in the membership function design, which led to a classification failure at key thresholds, specifically misclassifying 100 PPM as 'BURUK' instead of 'SEDANG' according to the ISPU standard. For future development, several improvements can be made. To enhance classification accuracy, it is highly recommended to redesign and refine the fuzzy membership functions to be fully aligned with the official ISPU standards, particularly at critical boundary points. Additionally, to improve the reliability of data transmission, alternative communication protocols such as LoRa or MQTT could be explored to ensure a more stable connection with lower power consumption, especially for deployment in areas with fluctuating cellular signals.

Author Contribution:

All authors contributed equally as the main contributors to this paper. All authors read and approved the final manuscript.

Funding:

This research received no external funding.

Acknowledgement:

The authors thank the Department of Electrical and Electronics Engineering, Vocational Faculty, Universitas Negeri Yogyakarta, for providing the facilities used in this research.

Conflicts of Interest:

The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

[1] Ramadhan, R. & Chandra, J. C. (2022). Reza rancang bangun sistem pemantauan kualitas udara berbasis iot dengan nodemcu. Prosiding Seminar Nasional Mahasiswa Fakultas Teknologi Informasi (SENAFTI), 1(1), 1183–1190.

[2] Sukma, N. M. K., Senoaji, N. F. A., & Restu, N. K. A. (2024). Analisis upaya penegakan hukum terhadap krisis lingkungan atas implikasi pencemaran udara akibat asap kendaraan bermotor di daerah khusus jakarta (dkj) tahun 2023. Deleted Journal, 1(3), 133–146.

[3] Kurnianto, D., Testy, K. N., & Yuliantoro, P. (2022). Sistem monitoring kualitas udara berbasis komunikasi lora di it telkom purwokerto. Dinamika Rekayasa, 18(1), 35.

[4] Mustafa, M., Sunuh, H. S., Subagyo, I., & Bungawati, A. (2023). Pencemaran udara dan ispa (infeksi saluran pernapasan akut).

[5] Fauziah, N. Z., Sudarti, S., & Yushardi, Y. (2024). Mekanisme terjadinya kanker kulit akibat radiasi sinar ultraviolet. SAINTIFIK, 10(1), 152– 156.

[6] Irma, M. F. (2024). Tingginya kenaikan suhu akibat peningkatan emisi gas rumah kaca di indonesia. Jurnal Sains Dan Sains Terapan, 2(1).

[7] Ramadhani, A. D., Ningsih, N., Nurcahya, A., & Azizah, N. (2023a). Klasifikasi dan Monitoring Kualitas Udara Dalam Ruangan menggunakan Thingspeak. Jurnal Teknik Elektro dan Komputer TRIAC, 10(1), 1–5.

[8] Hidayati, Q., Rachman, F. Z., & Rimbawan, M. A. S. (2020). Sistem monitoring kualitas udara berbasis fuzzy logic. *Prosiding Seminar Nasional Terapan Riset Inovatif (SENTRINOV), 6*(1), 260–267.

[9] Junaidi, A. (2015). Internet of things, sejarah, teknologi dan penerapannya : Review. *Jurnal Ilmiah Teknologi Informasi Terapan, 1*(3).

[10] Hakim, T. N. & Susanto, M. F. (2020). Sistem monitoring kualitas udara berbasis internet of things. *Prosiding Industrial Research Workshop and National Seminar, 11*(1), 496–502.

[11] Majidah, Z., Bianto, M. A., & Saputra, B. D. (2024). Implementasi fuzzy logic mamdani untuk monitoring kualitas udara berbasis iot. *Jurnal Pengembangan Teknologi Informasi dan Komunikasi (JUPTIK), 2*(1), 1–6.

[12] Muhammad, S. I., Rizal, A., & Maulana, I. (2024). Sistem pemantauan udara di kiic menggunakan metode fuzzy mamdani. *JATI (Jurnal Mahasiswa Teknik Informatika), 8*(1), 43–47.

[13] Virdaus, M. S. S. & Ihsanto, E. (2021). Rancang bangun monitoring dan kontrol kualitas udara dengan metode fuzzy logic berbasis WEMOS. *Jurnal Teknologi Elektro, 12*(1), 22.

[14] Budianto, H. & Sumanto, B. (2023). Perancangan sistem monitoring kualitas udara dalam ruangan berbasis internet of things. *Jurnal Listrik, Instrumentasi, dan Elektronika Terapan, 5*(1).

[15] Salasa, M. G., Rosadi, A., & Fahriani, N. (2021). Perancangan alat monitoring polusi udara berbasis mikrokontroler menggunakan sensor gas tgs-2442. *Computing Insight: Journal of Computer Science, 3*(1).

JRAEE

Journal of Robotics, Automation, and Electronics Engineering
Vol. 4, No. 1, 2026, pp. 268-280
e-ISSN 3025-3780
https://journal.uny.ac.id/v3/jraee

Integrating Color Segmentation and Texture Analysis for Citrus Leaf Disease Identification through K-Means and Local Binary Pattern

Armin[a,1], Jayanti Yusmah Sari[b,2], Suharsono Bantun[c,3]

[a,b,c] Universitas Sembilanbelas November Kolaka, Indonesia
[1] armin@usn.ac.id; [2] jayanti@usn.ac.id; [3] suharsonob@usn.ac.id
* Corresponding Author

ARTICLE INFO

Article History
Received 30 Aug 2026
Revised 31 Aug 2026
Accepted 31 Aug 2026

ABSTRACT

Early identification of citrus leaf diseases is important for reducing crop losses and supporting effective disease management. This study proposes a citrus leaf disease identification framework combining K-means color segmentation and Local Binary Pattern (LBP) texture analysis. The method was evaluated using 90 citrus leaf images from three disease classes: Citrus Canker, Leaf Miner, and Sooty Mold. Images were preprocessed through cropping, resizing, and color space transformation. K-means clustering in the Lab color space was used to segment diseased regions, while LBP extracted texture features. Classification was performed using Euclidean Distance and evaluated with 10-fold cross-validation. The K-means-based method achieved an average accuracy of 76.65%, while LBP alone obtained 45.65%. Combining K-means and LBP produced the best performance, with an average accuracy of 80.02%. These results indicate that integrating color and texture features provides a more effective representation of citrus leaf disease symptoms and improves classification performance.

Keywords
Citrus leaf disease identification;
K-means clustering;
Local Binary Pattern;
texture analysis;
computer vision.

ABSTRAK

Identifikasi dini penyakit daun jeruk penting untuk mengurangi kerugian hasil panen dan mendukung pengelolaan penyakit yang efektif. Penelitian ini mengusulkan kerangka identifikasi penyakit daun jeruk dengan menggabungkan segmentasi warna K-means dan analisis tekstur Local Binary Pattern (LBP). Metode ini dievaluasi menggunakan 90 citra daun jeruk dari tiga kelas penyakit, yaitu Citrus Canker, Leaf Miner, dan Sooty Mold. Citra di praproses melalui cropping, resizing, dan transformasi ruang warna. K-means pada ruang warna Lab digunakan untuk melakukan segmentasi area yang terinfeksi, sedangkan LBP digunakan untuk mengekstraksi fitur tekstur. Klasifikasi dilakukan menggunakan Euclidean Distance dan dievaluasi dengan 10-fold cross-validation. Metode berbasis K-means menghasilkan akurasi rata-rata 76,65%, sedangkan LBP sebesar 45,65%. Kombinasi K-means dan LBP memberikan performa terbaik dengan akurasi rata-rata 80,02%. Hasil ini menunjukkan bahwa integrasi fitur warna dan tekstur dapat merepresentasikan gejala penyakit daun jeruk secara lebih efektif dan meningkatkan

performa klasifikasi.

1. Introduction

Citrus is one of the most economically important fruit crops worldwide, contributing significantly to food security, agricultural trade, and rural livelihoods. The global citrus industry supports millions of farmers and is a major source of vitamins and nutritional compounds for human consumption [1]. However, citrus production faces numerous challenges caused by biotic stress, particularly plant diseases that affect vegetative growth, fruit quality, and overall productivity. Several diseases, including citrus canker, huanglongbing, citrus black spot, and citrus variegated chlorosis, are major threats to sustainable citrus cultivation worldwide [1]. Among these diseases, citrus canker is considered one of the most destructive bacterial diseases, causing lesions on leaves, stems, and fruits, which subsequently leads to reduced photosynthetic activity, premature fruit drop, deterioration of fruit quality, and significant yield losses [2].

Leaf diseases play a critical role in determining citrus productivity because leaves are directly involved in photosynthesis and metabolism. Disease symptoms commonly appear first on leaf surfaces, making leaves an important indicator of early disease diagnosis. Severe infections can result in defoliation, impaired nutrient transport, reduced fruit development, and substantial economic losses to farmers [2]. In addition, some diseases exhibit similar visual symptoms during their early stages, making manual diagnosis difficult and potentially leading to delayed treatment. Therefore, the development of reliable methods for early citrus leaf disease identification is essential to support disease management strategies and minimize production loss [1], [2].

Traditionally, disease diagnosis has been conducted through visual inspection by farmers or agricultural experts. Although this approach remains widely practiced, it is often subjective, time-consuming, and highly dependent on the expert knowledge. Variations in human perception and field conditions may also affect the consistency of the diagnosis. Recent advances in computer vision and image processing have created opportunities to develop automated disease identification systems capable of providing objective, rapid, and nondestructive assessments of plant health [3]. By analyzing digital images, computer vision systems can extract meaningful visual information, including color, texture, shape, and pattern characteristics, which can subsequently be used for disease recognition and classification [3], [4].

Image processing techniques have become increasingly important in agricultural applications because they enable automated monitoring and diagnosis without requiring direct physical contact with plants. In plant disease identification, image segmentation is commonly employed to isolate diseased regions from healthy tissues, whereas feature extraction techniques transform visual symptoms into quantitative representations suitable for classification [4]. Previous studies have demonstrated that disease symptoms frequently manifest as color abnormalities and texture alterations. Consequently, combining color- and texture-based information has the potential to improve disease representation and increase identification accuracy.

Several studies have investigated image-processing approaches for disease identification. Premana et al. [5] applied K-Means clustering for image segmentation using color and texture feature extraction on tomato fruit images. Their study demonstrated the feasibility of combining color and texture information; however, the classification performance and quantitative evaluation of the extracted features were not comprehensively reported. Consequently, the effectiveness of the integrated feature representation cannot be fully assessed.

Lestari et al. [6] developed a citrus leaf disease detection system based on RGB-HSV color segmentation and Fuzzy K-Nearest Neighbor (F-KNN) classification. This study focused on identifying citrus leaf diseases using color characteristics and achieved an accuracy of 69%. Although the proposed method demonstrated the usefulness of color information for disease recognition, the texture characteristics of disease symptoms were not explicitly incorporated into the identification process. As different diseases may exhibit similar color distributions, relying solely on color features may limit the classification performance.

Another study conducted by Ariesdianto et al. [7] employed K-Nearest Neighbor (K-NN) classification and gray-level co-occurrence matrix (GLCM) texture features for citrus leaf disease identification. Their approach analyzed two disease classes and one healthy leaf category, achieving an accuracy of 70%. Although texture analysis successfully represented the disease surface characteristics, color segmentation was not integrated into the framework. Consequently, complementary information associated with disease

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

discoloration patterns has not been fully utilized.

The literature suggests that both color and texture features provide valuable information for identifying diseases. However, previous studies have generally focused on either color-based or texture-based approaches independently. Color-based methods may encounter difficulties when different diseases exhibit similar color characteristics, whereas texture-based methods may be affected by variations in image quality and environmental conditions. Furthermore, previous citrus disease identification studies have reported moderate classification performance, indicating opportunities for improving disease representation through the integration of complementary image features [5], [6], [7].

Based on these observations, a research gap has been identified. Existing studies have not sufficiently explored the integration of K-means color segmentation and Local Binary Pattern (LBP) texture analysis for citrus leaf disease identification. K-means clustering is widely recognized for its ability to separate image regions based on color similarity, whereas LBP is a computationally efficient texture descriptor that captures local texture patterns. Integration of these two techniques may provide a more comprehensive representation of disease symptoms by simultaneously considering chromatic and textural characteristics.

The novelty of this study lies in the integration of color segmentation and texture analysis through the combination of K-means clustering and Local Binary Pattern. Unlike previous studies that primarily emphasized either color or texture features separately, the proposed framework exploits both types of information within a unified disease identification process. Disease regions are first segmented using K-means clustering in the L*a*b color space and subsequently analyzed using LBP to extract texture descriptors. This integrated representation is expected to improve the discrimination capability of different citrus leaf diseases.

The proposed framework was designed to identify three common citrus leaf diseases: citrus canker, leaf miner infestation, and sooty mold disease. Disease identification was performed using Euclidean Distance-based similarity measurement, and the overall performance was evaluated using a 10-fold cross-validation procedure to ensure objective assessment and reliability.

This study makes three contributions. First, we developed a computer vision-based framework for citrus leaf disease identification using integrated color and texture information. Second, it combines K-means clustering and local binary patterns to exploit complementary disease characteristics extracted from citrus leaf images. Third, it evaluates the effectiveness of the proposed approach using a standardized cross-validation procedure, providing insights into the applicability of the integrated image features for agricultural disease identification. The findings are expected to contribute to the advancement of computer vision and image processing applications in agricultural monitoring and plant disease management.

2. Methods

2.1 Dataset Acquisition

The dataset used in this study consisted of 90 citrus leaf images collected from citrus plantations in Siompu District, South Buton Regency, Southeast Sulawesi, Indonesia, using an OPPO A12 smartphone camera under natural light conditions. The images were stored in JPEG format and processed using MATLAB R2015a software. The dataset comprised three disease categories, namely citrus canker, leaf miner infestation, and sooty mold disease, with 30 images in each category. These diseases were selected because they are among the most commonly observed citrus leaf diseases and exhibit distinct color and texture. The distribution of images across the disease categories is presented in Table 1. As shown in the table, the dataset was evenly distributed among the three classes, reducing class imbalance and providing a fair basis for evaluating the proposed disease identification framework. To assess the robustness and generalization capability of the proposed method, a 10-fold cross-validation strategy was employed, in which the dataset was divided into ten subsets, with one subset used for testing and the remaining subsets used for training in each iteration. The process was repeated until all subsets served as testing data once, and the average performance across all iterations was reported as the final evaluation result.

Table 1. Distribution of Citrus Leaf Disease Dataset

Disease Category	Number of Images	Percentage (%)
Citrus Canker	30	33.33
Leaf Miner	30	33.33
Sooty Mold	30	33.33
Total	**90**	**100**

2.2 Research Framework

Figure 1 illustrates the overall research framework proposed for citrus leaf disease identification by integrating color segmentation and texture analysis. The framework begins with the acquisition of citrus leaf images, followed by an image preprocessing stage consisting of cropping, resizing, and RGB-to-L*a*b color space conversion to enhance the image consistency and facilitate color-based analysis. Subsequently, K-means clustering was employed to segment disease regions from healthy leaf tissues based on color characteristics.

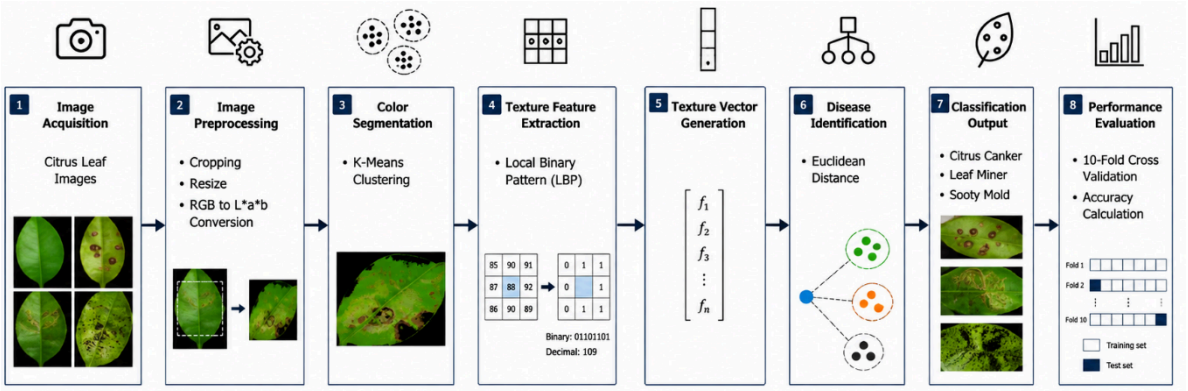


Fig. 1 Overall research framework for citrus leaf disease identification

The segmented regions were then processed using LBP to extract texture descriptors that represent disease-specific surface patterns. The resulting texture features were transformed into feature vectors and compared with stored feature vectors using the Euclidean Distance for disease identification. The classification output consisted of three disease categories: citrus canker, leaf miner infestation, and sooty mold disease. Finally, the performance of the proposed framework was evaluated using a 10-fold cross-validation scheme, and the classification accuracy was calculated to assess the effectiveness of integrating K-means color segmentation and Local Binary Pattern texture analysis for citrus leaf disease identification.

2.3 Image Preprocessing

Image preprocessing was performed to improve the image quality and ensure data consistency prior to feature extraction and disease identification. This stage involved cropping, resizing, and color space conversion, as illustrated in Figure 2. Cropping was applied to isolate the region of interest containing disease symptoms and eliminate irrelevant background information that could interfere with subsequent analyses. The cropped images were then resized to obtain uniform image dimensions, thereby reducing the computational complexity and ensuring consistent processing across all samples. Following resizing, the images were converted from the RGB color space to the L*a*b color space, which separates the luminance information from the chromatic components and provides a more effective representation of the color variations associated with disease symptoms. As shown in Figure 2, the preprocessing workflow transforms the original citrus leaf image into a standardized representation suitable for color-based analysis by sequentially performing region extraction, image normalization, and color-space transformation. These operations enhanced the visual distinction between diseased and healthy leaf tissues and provided a more reliable input for the subsequent K-means clustering stage, which segmented disease regions based on color characteristics.

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

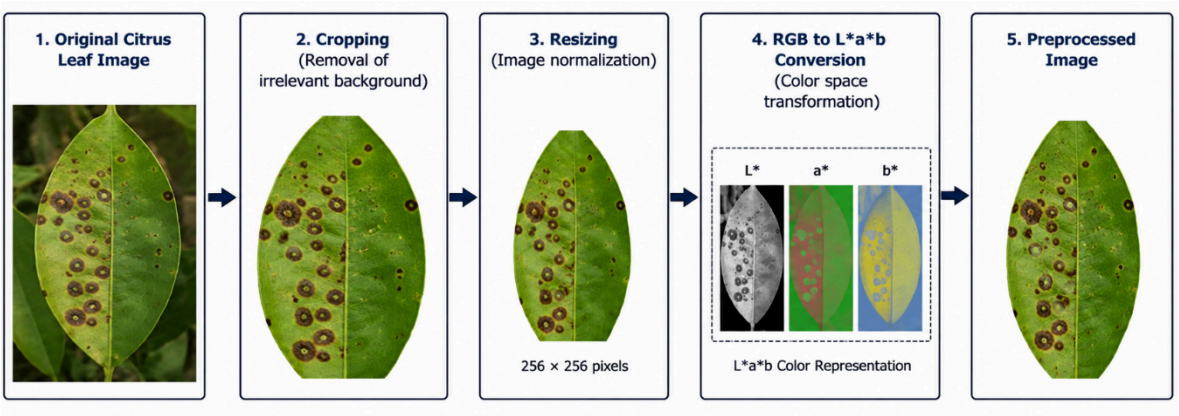


Fig. 2 Image preprocessing stages

2.4 Color Segmentation Using K-Means Clustering

Color segmentation was performed using the K-means clustering algorithm to isolate disease regions from healthy leaf tissues based on color characteristics. This stage plays a crucial role in citrus leaf disease identification because disease symptoms often appear as discoloration patterns that differ from the surrounding healthy tissues. Prior to clustering, the preprocessed images were represented in the L*a*b color space, which separates the luminance information from the chromatic components and facilitates more effective color discrimination. K-means clustering partitions image pixels into groups according to color similarity by iteratively assigning pixels to the nearest cluster centroid and updating the centroid positions until convergence is achieved. The distance calculation used during the cluster assignment follows the Euclidean Distance formulation presented in Equation (1).

$$D\left(x_i, c_j\right) = \sqrt{\sum_{k=1}^{n} \left(x_{ik}, c_{jk}\right)^2} \tag{1}$$

where:

x_i = feature vector of pixel i

c_j = centroid of cluster j

n = number of features

As illustrated in Figure 3, the segmentation process transforms the pre-processed image into clustered regions, enabling the extraction of disease-specific areas while suppressing irrelevant background and healthy tissue information. The resulting segmented regions provide a clearer representation of visual disease symptoms, including lesions, spots, and discoloration patterns associated with citrus canker, leaf miner infestation, and sooty mold. By isolating these disease regions, K-means clustering enhances the quality of subsequent texture analysis and improves the effectiveness of the overall disease identification framework.

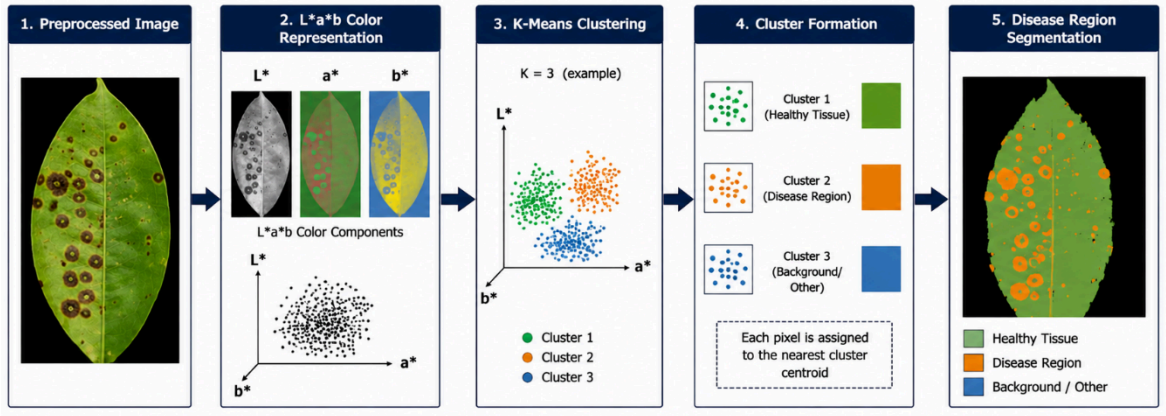


Fig. 3 K-Means color segmentation process

2.5 Texture Feature Extraction Using LBP

Following the segmentation stage, texture features were extracted using the LBP operator to characterize the surface patterns of the citrus leaf diseases. Texture analysis is particularly important in disease identification because many disease symptoms exhibit distinctive visual patterns, such as lesions, spots, and surface irregularities that may not be fully represented by color information alone. LBP is a computationally efficient texture descriptor that captures local texture variations by comparing the intensity values of each neighboring pixel with that of a center pixel within a predefined neighborhood. Neighboring pixels with intensity values greater than or equal to the center pixel are assigned a value of one, whereas lower values are assigned zero, resulting in a binary pattern that is subsequently converted into a decimal representation. The LBP computation follows the formulation presented in Equation (2), and the thresholding operation is defined in Equation (3).

$$LBP\left(x_c, y_c\right) = \sum_{P=0}^{P-1} s\left(i_p, i_c\right) 2^p \tag{2}$$

where:

i_c = center pixel intensity

i_p = neighboring pixel intensity

P = number of neighboring pixels

$$s(x) = \{1,\ x \geq 0\ 0,\ x < 0 \tag{3}$$

As illustrated in Figure 4, the texture extraction process begins with a segmented disease region, followed by a pixel neighborhood comparison, binary pattern generation, and formation of texture descriptors. This procedure enables the extraction of disease-specific texture characteristics that effectively represent the visual appearance of citrus canker, leaf miner infestation, and sooty mold. By transforming local texture information into discriminative feature vectors, LBP enhances the ability of the proposed framework to distinguish between different disease categories and supports subsequent disease classification.

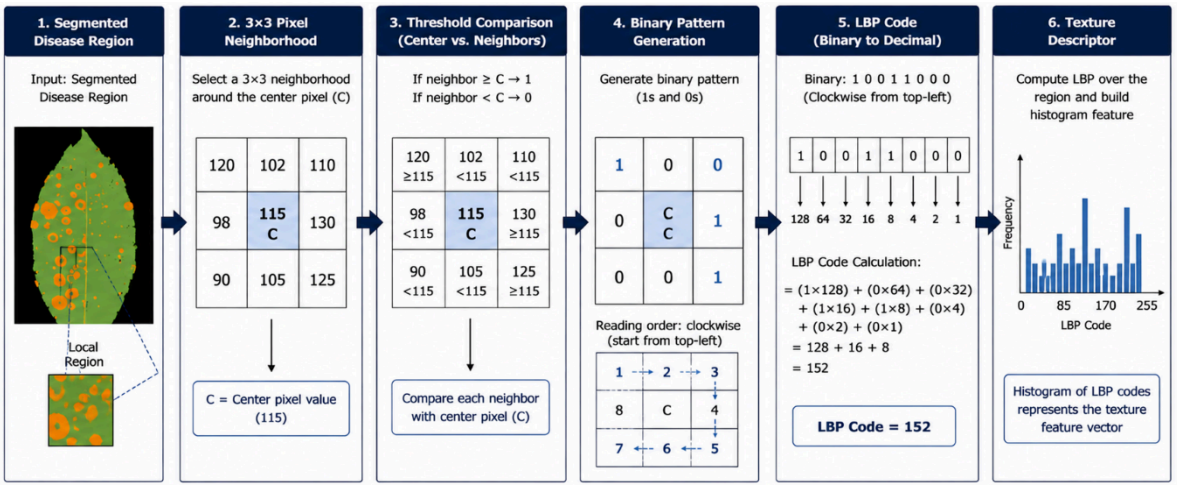


Fig. 4 Local binary pattern texture extraction process

2.6 Disease Classification

Disease classification was performed using the Euclidean Distance method to determine the similarity between feature vectors extracted from test images and those stored in the disease feature database. After the texture descriptors were generated using LBP, each feature vector was compared with the reference feature vectors representing the three disease classes, namely, Citrus Canker, Leaf Miner, and Sooty Mold. The similarity measurement was calculated using the Euclidean Distance formulation presented in Equation (4), where smaller distance values indicate a higher degree of similarity between the feature vectors.

$$d_{ij} = \sqrt{\sum_{k=1}^{n} \left(x_{ik}, x_{jk}\right)^2} \tag{4}$$

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

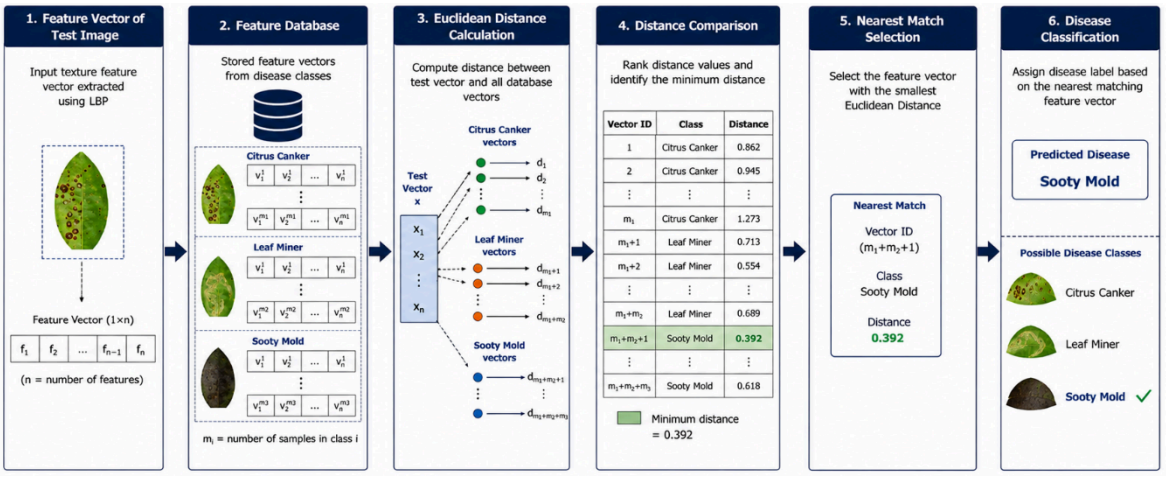


Fig. 5 Disease identification procedure using euclidean distance

The classification process follows the nearest-distance principle, in which a test image is assigned to the disease class associated with the minimum Euclidean Distance. As illustrated in Figure 5, the disease identification procedure begins with the extraction of feature vectors from the segmented disease region, followed by a similarity comparison with the stored feature database and the selection of the closest matching disease class. This approach enables the identification system to distinguish between different citrus leaf diseases based on their texture characteristics and provides a straightforward yet effective mechanism for disease recognition. The resulting classification output served as the basis for evaluating the effectiveness of the proposed framework in identifying citrus leaf diseases.

2.7 Performance Evaluation

The performance of the proposed citrus leaf disease identification framework was evaluated using a 10-fold cross-validation strategy to assess its robustness and generalization capability. In this approach, the dataset was divided into ten approximately equal subsets, where one subset was used as the testing data and the remaining nine subsets were used for training and feature database construction during each iteration. The process was repeated ten times until every subset had served as testing data once, ensuring that all images were included in both the training and testing phases. As illustrated in Figure 6, the cross-validation scheme systematically rotates the testing subset across all folds, thereby reducing the influence of data partitioning bias and providing a more reliable estimate of the model performance. This evaluation strategy is particularly suitable for relatively small datasets because it maximizes the utilization of the available data while improving the stability of the assessment results. The classification performance was measured using accuracy, which represents the proportion of correctly identified disease samples relative to the total number of testing samples and is calculated according to Equation (5).

$$Accuracy = \frac{N_{correct}}{N_{total}} x100\% \quad (5)$$

where:

$N_{correct}$ = number of correctly classified samples

N_{total} = total number of testing samples

The use of repeated training and testing across multiple folds enabled a more comprehensive evaluation of the proposed framework and minimized the likelihood of overestimating performance owing to favorable data splits. The average accuracy obtained from all ten folds was reported as the final performance indicator of the proposed citrus leaf disease identification system.

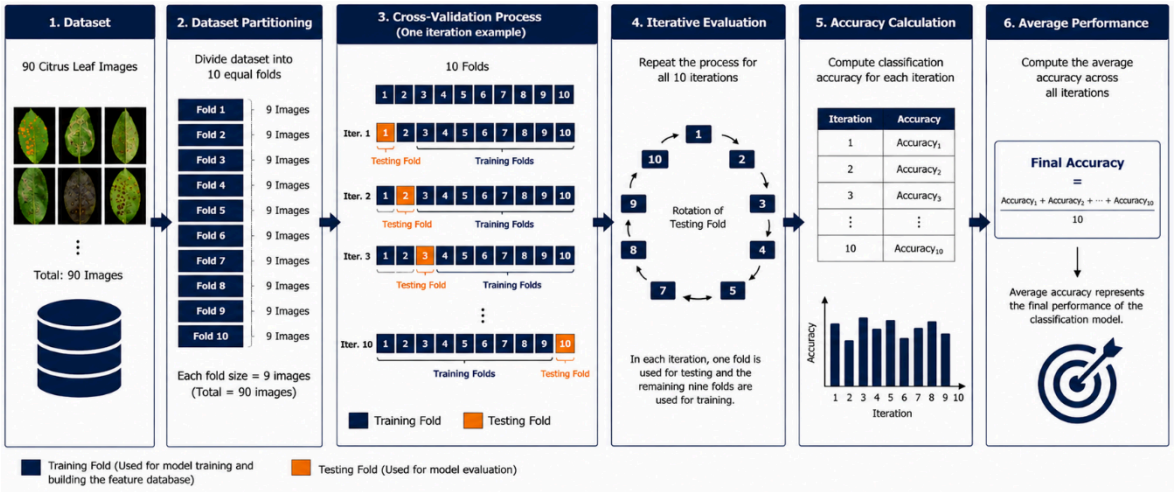


Fig. 6 Ten-fold cross validation scheme

3. Result and Discussion

3.1 Image Preprocessing Results

The image preprocessing stage transformed the acquired citrus leaf images into a standardized representation that was suitable for subsequent segmentation and feature extraction. As illustrated in Figure 7, cropping successfully isolated the region of interest by removing most irrelevant background information, allowing the analysis to focus on the leaf surface and visible disease symptoms. The resizing process further normalized the image dimensions across all samples, ensuring a consistent image representation throughout the dataset and reducing the variability associated with differences in image resolution. The resulting L*a*b images preserved the visual appearance of the disease lesions while providing a color representation in which the luminance information was separated from the chromatic components. This transformation enhanced the visibility of lesion-related color differences and reduced the influence of illumination variations in the original RGB images. As shown in Figure 7, the disease symptoms remained clearly distinguishable after preprocessing, indicating that important visual characteristics were retained throughout the transformation process. Consequently, the preprocessing stage produced a consistent and standardized dataset that facilitated subsequent K-means segmentation and supported the extraction of disease-related color and texture information in the proposed citrus leaf disease identification framework.

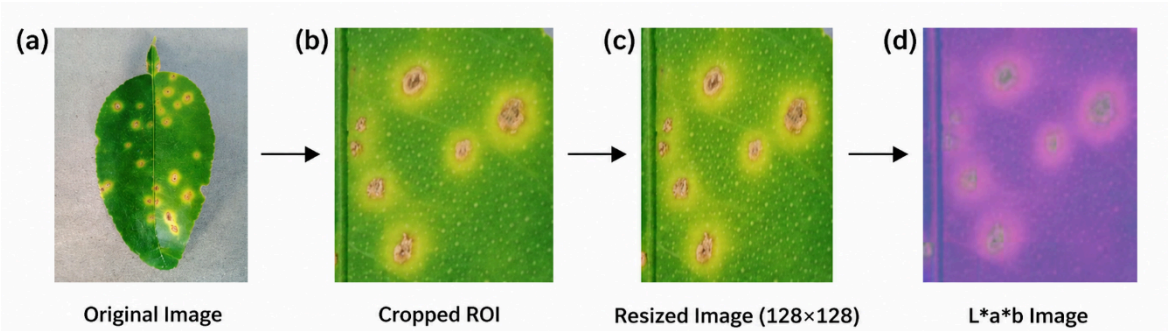


Fig. 7 Image preprocessing results used in the proposed citrus-leaf disease identification framework.

3.2 K-Means Segmentation Results

The segmentation results presented in Figure 8 demonstrate that color-based clustering can separate disease-affected regions from healthy leaf tissues by grouping pixels with similar color characteristics. The segmented outputs revealed distinct visual patterns for each disease category. Citrus Canker images exhibit localized lesion regions with a clear color contrast relative to the surrounding healthy tissue, enabling the symptomatic areas to be isolated more distinctly. In the Leaf Miner samples, the segmented regions corresponded to the characteristic mining trails that appeared as elongated discoloration patterns across the leaf surface. Meanwhile, the Sooty Mold images show concentrated dark regions associated with fungal deposits, which form visually distinguishable clusters within the segmented output.

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

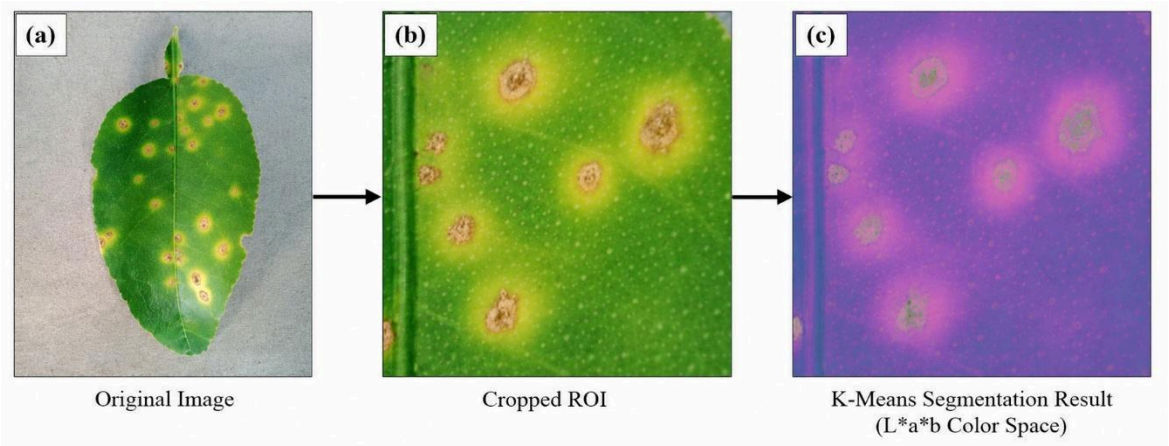


Fig. 8 Results of K-means-based color segmentation applied to citrus leaf images.

As illustrated in Figure 8, the segmentation process reduced the influence of irrelevant visual information and emphasized disease-related areas, resulting in a simplified representation of the leaves' surface. This outcome suggests that the color differences associated with disease symptoms provide sufficient information for clustering-based separation, allowing infected regions to be distinguished from healthy tissues. Similar observations have been reported in previous plant disease identification studies employing K-means segmentation, where color clustering effectively isolated symptomatic regions prior to feature extraction. The segmented images generated in this study also provided a more focused input for subsequent texture analysis, ensuring that the LBP features were extracted primarily from disease-related regions rather than from unaffected regions. Consequently, the segmentation results indicate that K-means clustering successfully highlighted the visual manifestations of citrus leaf diseases and established an appropriate foundation for the extraction of discriminative texture features in the next stage of the proposed framework.

3.3 Texture Analysis Results Using LBP

The texture analysis results shown in Figure 9 illustrate the transformation of the segmented citrus leaf image through several preprocessing stages before LBP feature extraction. Grayscale conversion simplified the image representation by retaining the intensity information while removing the color components, enabling the subsequent texture analysis to focus on the structural variations within the disease region. Median filtering further reduced the image noise while preserving important lesion boundaries, resulting in a smoother visual appearance. The histogram equalization stage enhanced the image contrast and increased the visibility of the local surface structures associated with the disease symptoms. As shown in Figure 9, these preprocessing operations produced a more distinct representation of the lesion area before applying the LBP. The final LBP image reveals detailed local texture patterns characterized by variations in pixel intensity relationships across the disease regions. Compared with the preceding images, the LBP representation emphasizes micro-texture structures that are less apparent in grayscale and filtered images. These results indicate that the LBP transformation successfully captured local texture information associated with citrus leaf disease symptoms and converted it into a representation suitable for subsequent feature extraction and Euclidean Distance-based classification. Therefore, the texture analysis process shown in Figure 9 provides an effective mechanism for preserving disease-related surface characteristics while enhancing the discriminative information required for identifying diseases.

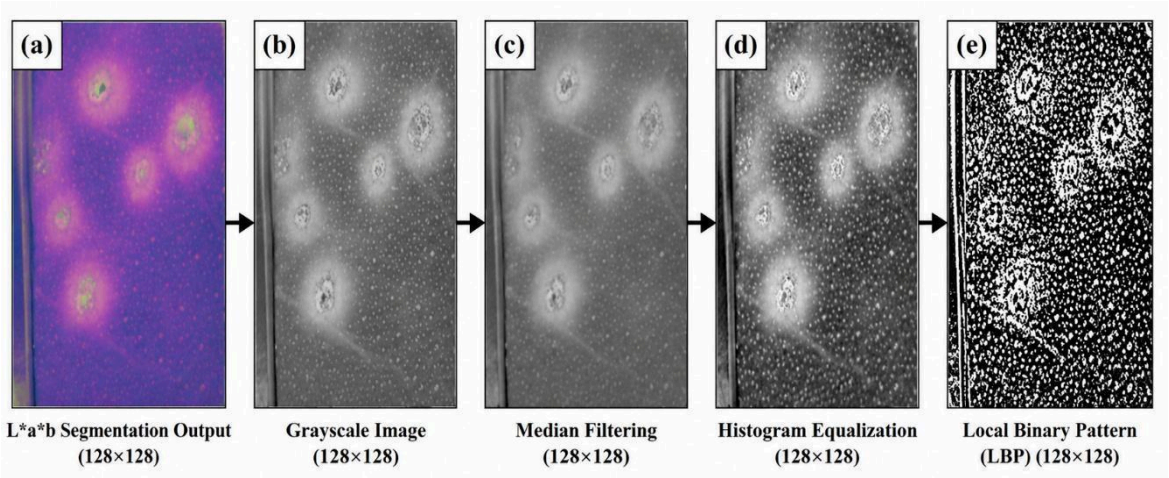


Fig. 9 LBP texture analysis results.

3.4 Feature Extraction Results

The statistical characteristics of the features extracted from the K-means and LBP processing stages are summarized in Table 2. The results indicate that the three disease classes exhibit different distributions of mean feature values, whereas the maximum (Max) and minimum (Min) values remain constant at 255 and 0, respectively, across all samples. This observation suggests that the mean feature serves as the primary discriminative descriptor in the proposed framework. As shown in Table 2, leaf miners generally exhibited the highest mean feature values, ranging from 109.64 to 124.50, reflecting the presence of elongated mining trails that generated relatively brighter texture responses after segmentation and LBP transformation. Citrus Canker presented lower mean values, ranging from 102.50 to 117.21, which correspond to localized lesion regions and irregular surface damage. Meanwhile, Sooty Mold demonstrated the widest variation, with mean values ranging from 97.60 to 122.81, indicating substantial diversity in the distribution of fungal deposits across the affected leaf surface. Despite these differences, a noticeable overlap exists among the mean feature ranges of the three disease classes. For example, several Citrus Canker samples exhibited mean values comparable to those observed in Leaf Miner and Sooty Mold samples, resulting in similar numerical representations. Such overlap reduces the separability of the feature space and may contribute to classification ambiguity when the Euclidean Distance is used to identify the nearest matching sample. Similar observations have been reported in previous plant disease identification studies employing statistical image descriptors, where simple intensity-based features effectively capture disease characteristics but may experience reduced discriminative capability when different disease symptoms share comparable visual patterns. Nevertheless, the results presented in Table 2 demonstrate that the extracted features provide a compact representation of disease-specific texture information and constitute an appropriate basis for the subsequent classification.

Table 2. Statistical Summary of Features Extracted from K-Means and LBP Processing

Disease Class	n	Mean Feature Range	Range Mean ± SD	Max	Min
Citrus Canker	30	102.50–117.21	To be calculated from all samples	255	0
Leaf Miner	30	109.64–124.50	To be calculated from all samples	255	0
Sooty Mold	30	97.60–122.81	To be calculated from all samples	255	0

3.5 Classification Performance

The classification performance of the proposed framework is shown in Figure 10. The experimental results show that the integration of K-means color segmentation and LBP texture analysis achieved an average classification accuracy of 80.02%, indicating the effectiveness of combining color- and texture-based information for citrus leaf disease identification.

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

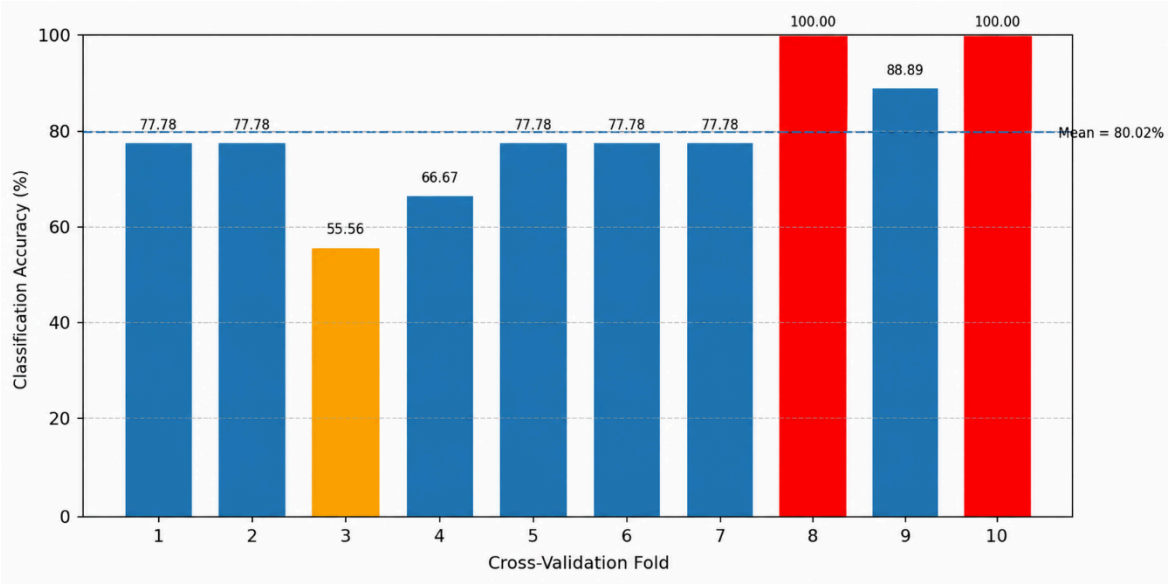


Fig. 10 Classification accuracy across 10-fold cross-validation

As shown in Figure 10, the classification accuracy varied across the ten cross-validation folds, ranging from 55.56% in Fold 3 to 100.00% in Folds 8 and 10. Despite this variation, most folds achieved accuracies above 75%, suggesting that the proposed framework maintained a relatively stable performance across different training and testing partitions. The occurrence of perfect classification performance in two folds demonstrates that the extracted color and texture features were highly discriminative for certain data partitions, whereas lower accuracies in other folds indicate the presence of samples with more challenging visual characteristics. The observed performance variation is consistent with the feature analysis presented in Section 3.4, where an overlap in mean feature values was observed among some disease categories. Such overlap may reduce class separability and increase the likelihood of misclassification when the Euclidean Distance is used as the similarity measure. Nevertheless, the overall trend shown in Figure 10 indicates that the proposed framework consistently achieved a satisfactory classification performance across different folds. Compared with previous citrus leaf disease identification studies, the obtained average accuracy of 80.02% is higher than the 69% reported by Lestari et al. using RGB-HSV color segmentation and Fuzzy K-Nearest Neighbor classification and the 70% reported by Ariesdianto et al. using GLCM texture features and K-Nearest Neighbor classification. These findings suggest that the integration of K-means segmentation and LBP texture descriptors provides a more informative representation of disease symptoms and improves the classification performance. Overall, the results demonstrate that the proposed framework effectively exploits complementary color and texture information to support reliable citrus-leaf disease identification.

3.6 Discussion

The experimental findings demonstrate that the integration of color segmentation and texture analysis provides a more effective representation of citrus leaf disease symptoms than the use of individual visual descriptors. The preprocessing stage successfully standardized the input images through cropping, resizing, and color space transformation, ensuring that disease-related information was preserved while reducing irrelevant variations caused by the background elements and image acquisition conditions. This standardized representation subsequently enhanced the effectiveness of K-means segmentation, which isolated disease-affected regions by exploiting the color differences associated with lesion development, chlorosis, and fungal contamination. The importance of color information is evident from the classification results, where the K-means-based approach achieved an average accuracy of 76.65%, which is substantially higher than the 45.65% obtained using LBP-based texture analysis alone. These findings indicate that color characteristics are an important visual cue for differentiating citrus leaf diseases. However, the additional improvement to 80.02% achieved by integrating K-means and LBP demonstrates that color information alone is insufficient to fully characterize disease symptoms and that texture descriptors contribute complementary discriminatory information. From a computer vision perspective, K-means segmentation primarily improves disease localization by emphasizing disease-related color distributions, whereas LBP captures local structural variations associated with lesion boundaries, tissue degradation, mining trails, and

fungal deposits. Consequently, segmentation quality directly influences texture feature quality because accurate disease region extraction reduces the inclusion of irrelevant healthy tissue and background pixels, enabling LBP to encode more representative disease patterns.

The feature extraction results further support this conclusion. While the Max and Min feature values remained constant at 255 and 0 across all disease categories, the Mean feature exhibited noticeable variation among Citrus Canker, Leaf Miner, and Sooty Mold samples, indicating that the Mean feature carried the primary discriminative information used during classification. However, partial overlap among the mean feature ranges was observed, suggesting that certain disease categories shared similar visual characteristics. This overlap is consistent with the pattern recognition theory, which states that the classification performance is strongly influenced by the class separability within the feature space. Because the Euclidean Distance classification relies on measuring the proximity among feature vectors, overlapping feature distributions reduce inter-class discrimination and increase the likelihood of misclassification. This phenomenon is reflected in the cross-validation results, where the classification accuracy varied from 55.56% to 100.00%, with an overall average of 80.02%. The occurrence of perfect classification performance in Folds 8 and 10 indicates that the extracted features were highly discriminative for specific testing partitions, whereas the lower accuracy observed in Fold 3 suggests the presence of samples with highly similar feature representation. Despite this variation, most folds achieved accuracies above 75%, indicating relatively stable performance and acceptable generalization capability across different training and testing partitions.

Compared with previous studies, the proposed framework demonstrated competitive performance. Lestari et al. reported an accuracy of 69% using RGB-HSV color segmentation combined with Fuzzy K-Nearest Neighbor (KNN) classification, while Ariesdianto et al. achieved 70% using GLCM texture features and K-Nearest Neighbor classification. The higher accuracy achieved in this study suggests that combining K-means segmentation and LBP texture descriptors provides a richer representation of disease symptoms by integrating complementary color and texture information. Nevertheless, this study has several limitations. The framework relies on handcrafted features and a simple Euclidean Distance classifier, which may not fully capture complex intra-class variability and inter-class similarity. Furthermore, overlapping feature distributions and visually similar disease manifestations, particularly among categories exhibiting comparable texture characteristics, continue to limit the classification performance. Future research may benefit from incorporating more discriminative feature representations and advanced classification models that can learn complex decision boundaries. Overall, the results confirm that the integration of color segmentation and texture analysis constitutes an effective and computationally efficient strategy for citrus leaf disease identification and provides a practical foundation for agricultural computer vision applications.

4. Conclusion

In this study, we propose a citrus leaf disease identification framework that integrates K-means color segmentation and LBP texture analysis to improve the disease recognition performance. The experimental results demonstrated that the preprocessing stage effectively standardized image representation through cropping, resizing, and color space transformation, thereby preserving the disease-related visual characteristics while reducing irrelevant image variations. K-means segmentation successfully isolated disease-affected regions and emphasized the color information associated with lesion development, whereas LBP captured local texture patterns related to tissue damage and disease symptoms. Feature analysis revealed that the mean feature provided greater discriminative information than the maximum and minimum features, which were constant across disease categories. The classification experiments further showed that K-means segmentation alone achieved an average accuracy of 76.65%, whereas LBP-based texture analysis alone achieved 45.65%. The integration of K-means and LBP produced the highest performance, achieving an average classification accuracy of 80.02% under 10-fold cross-validation, indicating that color and texture information provide complementary representations for citrus leaf disease identification.

The findings confirm that combining color-based segmentation and texture-based feature extraction improves disease representation and classification effectiveness compared with using either approach independently. Nevertheless, this study has several limitations. Partial overlap among feature distributions reduces class separability, particularly for disease categories exhibiting similar visual characteristics. In addition, the use of handcrafted features and Euclidean Distance classification may limit the ability of the framework to capture more complex disease variations. Therefore, future research should investigate more discriminative feature representations, larger and more diverse datasets, and advanced machine or deep learning classifiers to further improve the classification accuracy and generalization performance. Overall, the proposed framework provides an effective, computationally efficient, and practical approach for

ISSN 3025-3780 (online)
ISSN 3025-4590 (printed)

automated citrus leaf disease identification, with potential applications in agricultural monitoring and decision-support systems.

Author Contribution:

All authors contributed equally as the main contributors to this paper. All authors read and approved the final manuscript.

Funding:

This research received no external funding.

Acknowledgement:

The authors thank the Department of Electrical and Electronics Engineering, Vocational Faculty, Universitas Negeri Yogyakarta, for providing the facilities used in this research.

Conflicts of Interest:

The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

[1] H. B. Barbieri, L. S. Fernandes, J. G. de M. Pontes, A. K. Pereira, and T. P. Fill, "An overview of the most threatening diseases that affect worldwide citriculture: Main features, diagnose, and current control strategies," *Front. Nat. Prod.*, vol. 2, p. 1045364, 2023.

[2] S. Ali *et al.*, "Citrus canker: a persistent threat to the worldwide citrus industry—an analysis," *Agronomy*, vol. 13, no. 4, p. 1112, 2023.

[3] J. R. Sykes, K. J. Denby, and D. W. Franks, "Computer vision for plant pathology: A review with examples from cocoa agriculture," *Appl. Plant Sci.*, vol. 12, no. 2, p. e11559, 2024.

[4] A. Upadhyay *et al.*, "Deep learning and computer vision in plant disease detection: a comprehensive review of techniques, models, and trends in precision agriculture," *Artif. Intell. Rev.*, vol. 58, no. 3, p. 92, 2025.

[5] A. Premana, R. M. H. Bhakti, and D. Prayogi, "Segmentasi K-Means Clustering Pada Citra Menggunakan Ekstrasi Fitur Warna dan Tekstur," *J. Ilm. Intech Inf. Technol. J. UMUS*, vol. 2, no. 01, pp. 89–97, 2020.

[6] F. Lestari, J. Sari, I. P. Sutardi, and N. Purnama, "Deteksi Penyakit Tanaman Jeruk Siam Berdasarkan Citra Daun Menggunakan Segmentasi Warna RGB-HSV," in *Seminar Nasional Teknologi Terapan Berbasis Kearifan Lokal (SNT2BKL)*, 2018, pp. 276–283.

[7] R. H. Ariesdianto, Z. E. Fitri, A. Madjid, and A. M. N. Imron, "Identifikasi Penyakit Daun Jeruk Siam Menggunakan K-Nearest Neighbor," *J. Ilmu Komput. dan Inform.*, vol. 1, no. 2, pp. 133–140, 2021.