EVOLUTION IN ELECTRICAL AND ELECTRONIC ENGINEERING
e-ISSN: 2756-8458

EEEE

Vol. 6 No. 2 (2025) 34-43
https://publisher.uthm.edu.my/periodicals/index.php/eeee

Yolov5–Based Freshness Detection of Selected Vegetables Using Raspberry Pi

Raja Ummu Sulaim Raja Mohd Azman[1], Mariyam Jamilah Homam[1]*

1 Faculty of Electrical and Electronic Engineering
Universiti Tun Hussein Onn Malaysia, Batu Pahat. 86400, Johor, Malaysia

*Corresponding Author: mariyam@uthm.edu.my
DOI: https://doi.org/10.30880/eeee.2025.06.02.005

Article Info

Received: 22 June 2025
Accepted: 14 August 2025
Available online: 30 October 2025

Keywords

Yolov5, Object Detection, Vegetable Freshness, Raspberry Pi, Real-Time Monitoring, Image Classification, Food Waste Reduction.

Abstract

Maintaining the freshness of vegetables is essential to reduce food waste and ensure customer satisfaction. This work presents an automated monitoring system that uses the YOLOv5 object detection model to classify the freshness of selected vegetables—specifically tomatoes, eggplants, and red onions. A dataset of 800 annotated images (80% for training and 20% for validation) was used to train the model to distinguish between fresh and spoiled produce. The system is implemented using a Raspberry Pi 4 Model B connected to a camera module, which captures images in real time for freshness detection. Based on the classification result, the system triggers physical alerts: if fresh vegetables are detected, the green LED turns on; if spoiled vegetables are detected, the red LED lights up and the buzzer is activated to warn staff. A web interface provides real-time visual feedback, displaying annotated images, confidence scores, and freshness logs. The model achieved an accuracy of 80% under ideal lighting conditions, with confidence scores ranging from 52% to 84% for fresh produce. Challenges include difficulty detecting early-stage spoilage and reduced performance in poor lighting. This system has potential for improving freshness monitoring and quality control in retail stores through automation and real-time alerts.

1. Introduction

Ensuring vegetable freshness is crucial for customer satisfaction and reducing food waste [1]. Manual inspection is inefficient, especially at scale, prompting the need for automated solutions using image processing. YOLO (You Only Look Once) is an effective real-time object detection algorithm that can identify signs of wilting [2]. When paired with Raspberry Pi and a camera module, it enables low-cost, automated freshness monitoring. Integrating this system with a web-based notification platform allows real-time alerts to staff and customers, improving response time and minimizing waste [3]. This aligns with modern trends that prioritize smart and scalable technology solutions [4]. Previous research supports this approach. Studies on YOLOv8, CNNs, and machine learning have shown high accuracy in crop disease detection [5–7]. For example, YOLOv8 achieved a 6.46% improvement in greenhouse monitoring, while multispectral imaging with VGG16 reached 86.73% accuracy in early wilt detection [8]. A YOLOv5-based mobile app for tomato disease achieved 0.76 mAP [9]. YOLOv5 was selected for this project due to its accuracy, speed, and lightweight design, making it suitable for real-time applications on Raspberry Pi. With further enhancements, this prototype could be applied in retail settings to improve freshness monitoring and reduce waste.

Evolution in Electrical and Electronic Engineering Vol. 6 No. 2 (2025) p. 34-43

2. Methodology

Fig. 1 shows the flowchart vegetables are monitored by the automated system through image processing and artificial intelligence. The process starts every day by automatically taking an image. The YOLO algorithm is then used to analyze the image, since it is famous for being able to detect objects fast in real time. By looking at the image, YOLO can recognize different vegetables and decide if they are fresh or past their prime. When classification has finished, the results are shown on a monitor so that the user or system operator can see them. Next, the system checks to see if the product shows any wilting, because this might suggest the product has started spoiling. If vegetables are fresh, the green LED lights up to show the vegetables are still fresh and the system waits for the next day. If the vegetables show wilting, the buzzer sounds and a red LED light up to show that the vegetables are damaged. The system lets the user know through a web app. It may let the operator know the name of the vegetable, its category and how long ago it was found. The person can stay updated remotely and take the required action if required. At this stage, the process steps back in after the next set image capture to keep checking the quality of the vegetables.

Fig. 2 shows the key hardware components of the system which revolve around a Raspberry Pi. On the Raspberry Pi, sensor data and signals from peripherals are collected and outputs are generated following an analysis of the captured images. The Raspberry Pi is connected to a web camera to take pictures of vegetables regularly. The images get processed by the system to find out the state of the vegetables. The Raspberry Pi is connected to an adapter which allows uninterrupted use of the system. To show the condition, the Raspberry Pi controls two LED lights: the green LED comes on when the vegetables are fresh and the red LED is on when the vegetables look like they are wilting or rotting. Also, the buzzer is turned on at the same time as the red LED when spoilage occurs, to make an audible noise. The monitor is linked to the Raspberry Pi so that the analysis results such as the category and state of the vegetables, can be seen. Users can monitor how fresh their stored vegetables are in real time by looking and listening to the display.

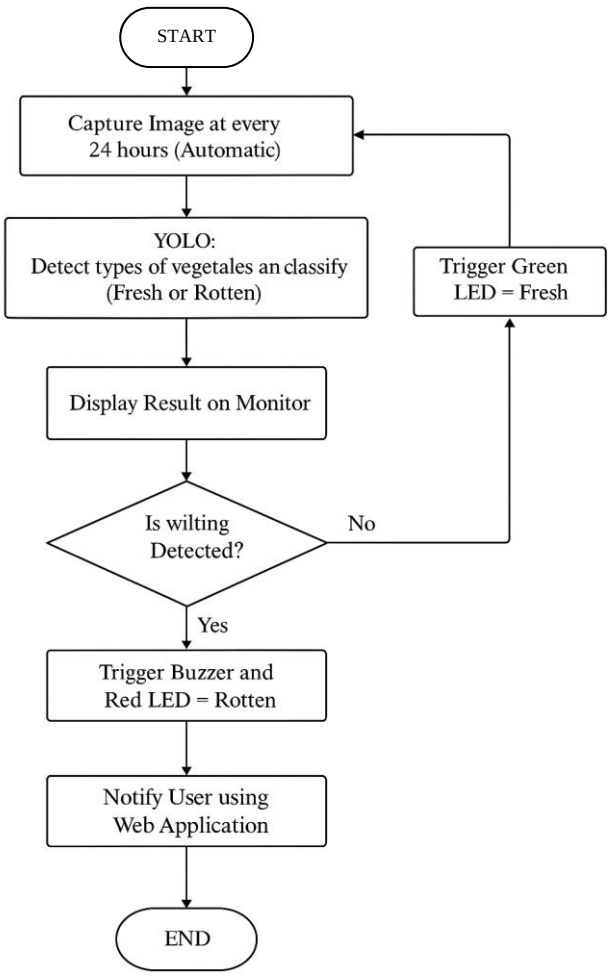


Fig. 1 *Flowchart of project*

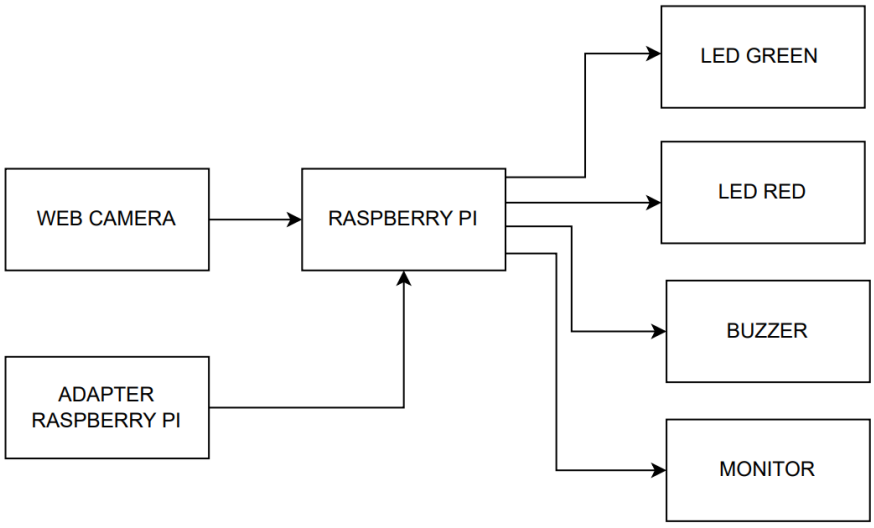


Fig. 2 *Block Diagram of project*

The development of the embedded vision system progresses from initial prototyping to a fully functional final product. Fig. 3 shows the initial prototyping phase, where the components are assembled on a breadboard for testing purposes. The setup includes a Raspberry Pi board (component 1), a web camera (component 2), a buzzer (component 3), a LED (component 4), Raspberry Pi Adapter (component 5) and a monitor (component 6). This configuration allows for easy modification and debugging, facilitating circuit validation and software testing in an open and accessible manner.

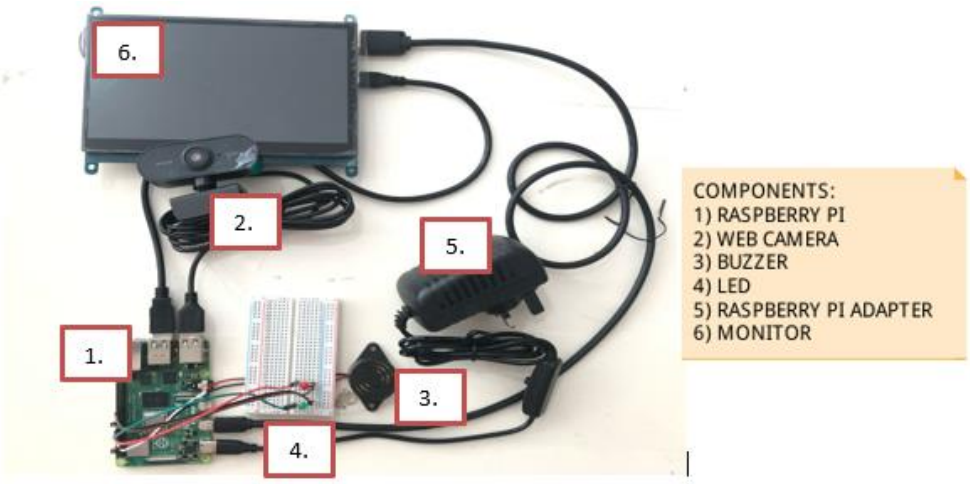


Fig. 3 *Circuit of the project*

2.1 Detection Process

In this project, the data collection and model training process involved teaching the system to recognize whether vegetables are fresh or rotten using image samples. A total of 800 images of tomatoes, eggplants, and red onions were collected from both real environments and online sources. These images were selected to show different lighting conditions, angles, and freshness levels to help the model work well in real-life situations. Before training, each image was manually labelled using a tool called Make Sense, where boxes were drawn around the vegetables and assigned with specific labels like "fresh tomato" or "rotten eggplant." These labels helped the YOLOv5 model understand what to look for during training. The dataset was then divided into two parts: 80% for training and 20% for validation. The training was done on a Raspberry Pi using a lightweight version of YOLOv5 (yolov5s.pt), which is suitable for devices with limited computing power. During training, the system learned to detect vegetables and classify their freshness based on the labelled data, improving its accuracy by learning from its errors.

Evolution in Electrical and Electronic Engineering Vol. 6 No. 2 (2025) p. 34-43

2.2 Database and Web Application

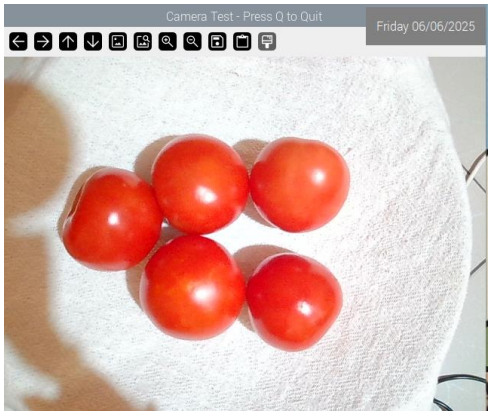
In this project, a database and web application were created to store and show the results of vegetable freshness detection. The system uses MySQL to save important details like the type of vegetable, its freshness status (fresh or rotten), the time and date it was detected, and the image taken. This data is saved automatically every time the system runs. To manage and view this information, phpMyAdmin is used as a simple interface. A web application was also built using Visual Studio Code, which lets users view real-time updates of the detection results on a website. The web app connects to the database on the Raspberry Pi, so users can check the condition of vegetables anytime.

3. Result and Discussion

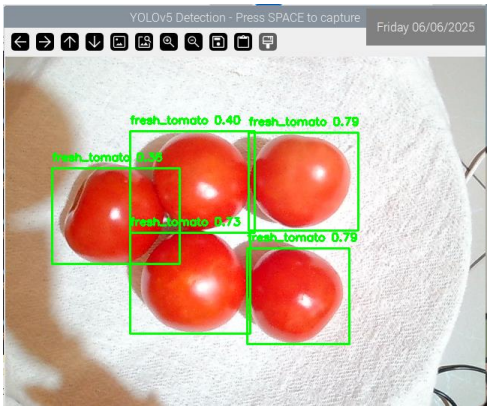
Fig. 4(a) shows the vegetable freshness detection system being tested in an indoor environment designed to mimic typical store display conditions. Stable lighting was provided by ceiling lights and the webcam's built-in LED, enhancing image clarity. The webcam, mounted on a stand 30 cm above the vegetables, captured top-down images on a plain white background to reduce visual noise. The test was conducted at room temperature (25°C), ensuring consistent conditions for accurate YOLO model performance.

In this project, three types of camera preview scripts are used to support the vegetable freshness detection system. Fig. 4(b) shows check_cam.py, which is used to test if the camera is working correctly by displaying a live preview with a simple message, allowing users to ensure the camera is ready before detection. Fig. 4(c) is open_captured_cam.py, which provides a real-time preview of the vegetables along with detection results such as the type of vegetable, freshness status, and confidence score; this is useful for manual monitoring and testing. The third script, auto_capture.py, runs automatically at a scheduled time (e.g., 1:27 AM) to capture images, perform freshness detection, and store the results without needing any user input. These three modes help make the system reliable, flexible, and suitable for both manual and automated operation.

(a)



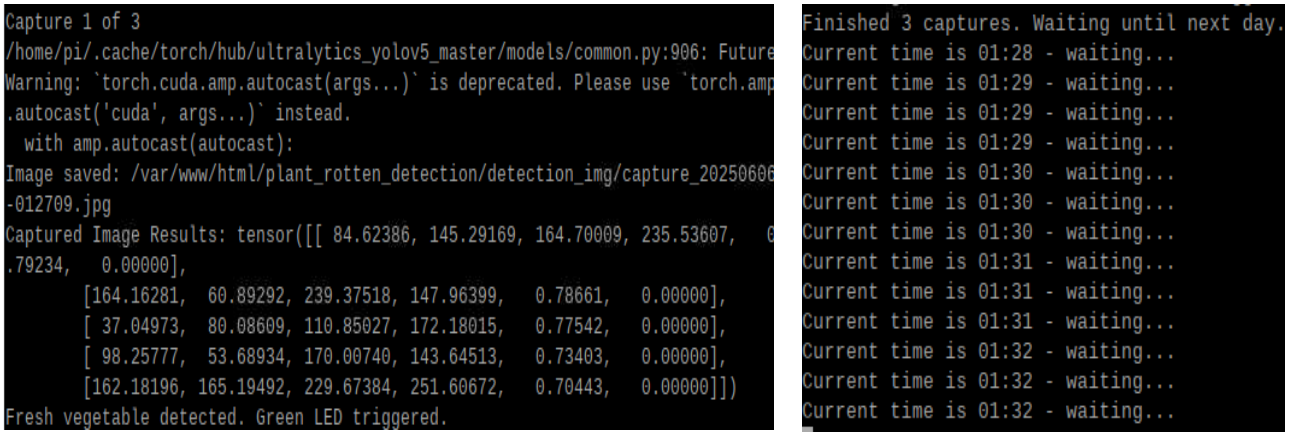
(b)



(c)

Fig. 4 *Camera Detection (a) Setup for Vegetable Image Capture with Lighting and Camera; (b) Camera Preview for check_cam.py Command (c) Camera preview for open_captured_cam.py and auto_capture.py Command*

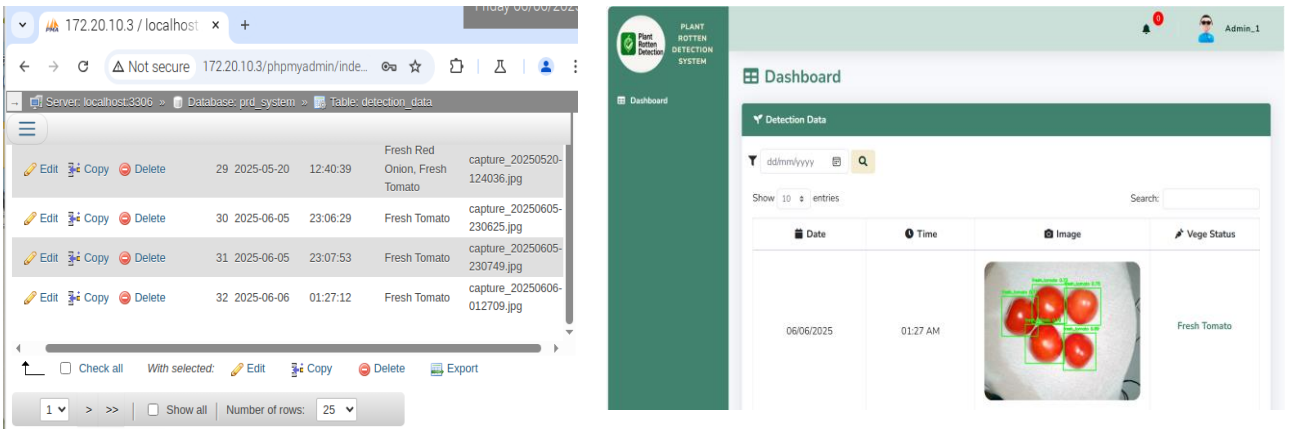
The code in Fig. 5(a) automates plant detection by capturing three images at 1:27 AM daily. It checks the current time, closes any camera preview, takes photos at 5-second intervals, and prevents duplicate runs. The YOLOv5 model detects fresh tomatoes (confidence scores ~0.70-0.79) and triggers a green LED. Images are saved with timestamps (e.g., capture_20250606-012709.jpg). After three successful captures, the system waits until the next day as shown in Fig. 5(b).



(a) **(b)**

Fig. 5 *Detection Result in Raspberry Pi (a) Result for 1 of 3 Capture; (b) Waiting Until Next Day*

The database table in Fig. 6(a) records vegetable freshness detection results in a MySQL database named "prd system". Each entry includes an ID, timestamp, detected item (e.g. "Fresh Tomato"), and corresponding image filename. The table shows multiple detections of fresh tomatoes and onions, demonstrating the system's consistent operation over time. Users can manage records through Edit/Copy/Delete functions, with 25 total entries currently stored. This organized logging system effectively tracks and stores all detection data for analysis.



(a) **(b)**

Fig. 6 *Detection Results (a) Database; (b) Web Application*

The F1-score is a single metric that balances precision (P) and recall (R), making it useful when both false positives and false negatives need to be minimized. Precision measures the proportion of correct detections out of all detections made, while recall measures the proportion of actual objects correctly detected by the model. These P and R values are obtained during the model evaluation stage after training, when the trained model is tested on a validation or test dataset. A higher F1-score indicates strong performance in both correctly identifying objects and avoiding missed detections. Using these values, the F1-score can be calculated with the formula:

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{1}$$

Evolution in Electrical and Electronic Engineering Vol. 6 No. 2 (2025) p. 34-43

Based on the Table 1, the model's performance shows noticeable variation between classes. The *rotten_tomato* class achieves the highest F1-score of 0.749, supported by a very high recall of 0.932, indicating that the model detects almost all rotten tomatoes with good precision. Classes like *fresh_eggplant* (F1=0.728) and *fresh_red_onion* (F1=0.723) also demonstrate a balanced performance between precision and recall. On the other hand, *rotten_red_onion* records the lowest F1-score of 0.525 due to its low precision (0.429), meaning many detections for this class are false positives. Overall, the average recall of 0.971 is much higher than the average precision of 0.483, suggesting that the model is effective at detecting the presence of objects but less accurate at correctly classifying them. Improving precision, particularly for the lower-performing classes, would help boost the model's overall accuracy.

Table 1 *Model Evaluation Metrics (Precision, Recall, and F1-Score) by Class*

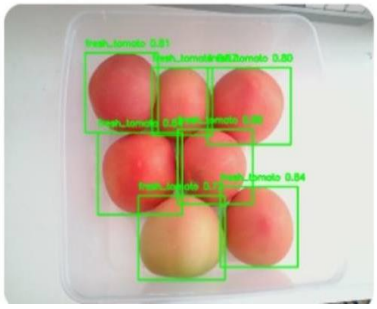
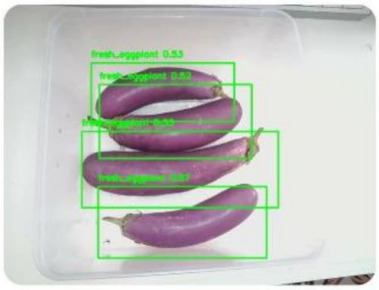
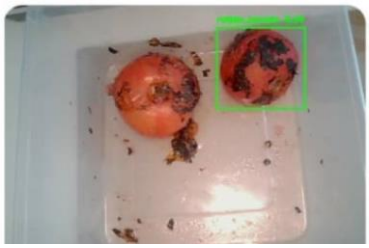
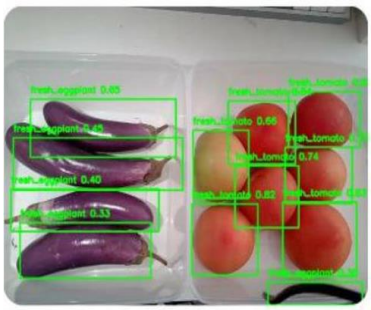
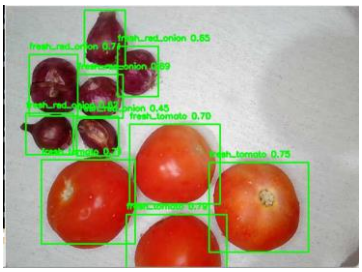
Class	P	R	F1-Score
fresh_tomato	0.607	0.807	0.693
rotten_tomato	0.626	0.932	0.749
rotten_eggplant	0.761	0.682	0.719
fresh_eggplant	0.698	0.761	0.728
rotten_red_onion	0.429	0.675	0.525
fresh_red_onion	0.645	0.822	0.723
Average	0.483	0.971	0.645

The results show high recall across most classes, indicating effective object detection, but lower precision suggests more false positives. Performance varies, with some classes showing balanced metrics while others, such as *rotten_red_onion*, have weaker precision.

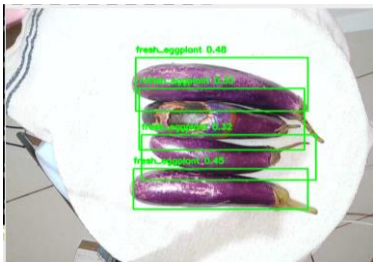
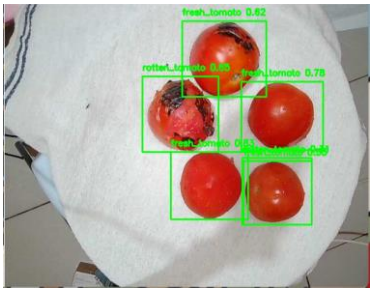
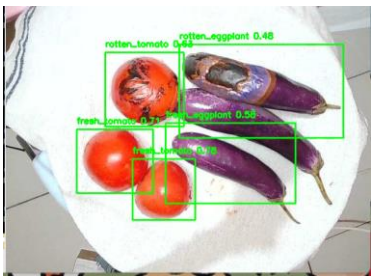
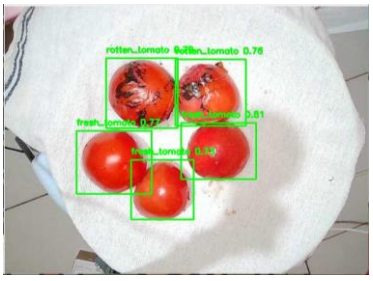
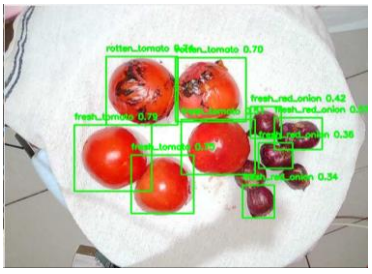
Table 2 shows the accuracy of the vegetable freshness detection system was tested using 15 different vegetable samples consisting of tomatoes, red onions and eggplants. Each sample's actual freshness status was known prior to testing, and the system's predicted status was compared against the ground truth. The system's ability to correctly classify the freshness status was then evaluated.

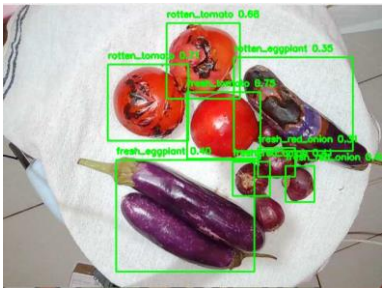
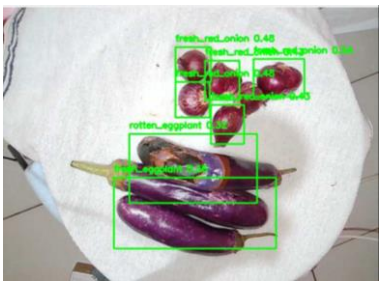
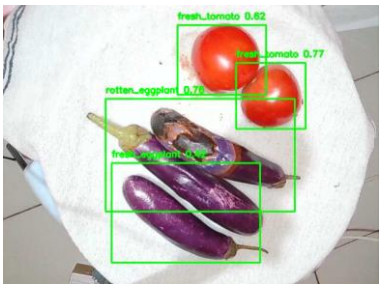
Table 2 *Vegetable Freshness Classification Performance*

No	Sample	Vegetable Type	Actual Freshness Status	Predicted Freshness Status	Result (Correct/ Incorrect)
1.		Red onion	Fresh	Fresh	Correct

2.		Tomato	Fresh	Fresh	Correct
3.		Eggplant	Fresh	Fresh	Correct
4.		Tomato	Rotten	Not detect	Incorrect
5.		Tomato and eggplant	Fresh	Fresh	Correct
6.		Red Onion and Tomato	Fresh	Fresh	Correct

Evolution in Electrical and Electronic Engineering Vol. 6 No. 2 (2025) p. 34-43

7.		Red onion	Fresh and Rotten	Not detect as rotten red onion	Incorrect
8.		Eggplant	Fresh and Rotten	Fresh and Rotten	Correct
9.		Tomato	Fresh and Rotten	Rotten tomato detects as fresh	Incorrect
10.		Tomato and eggplant	Fresh and Rotten	Fresh and Rotten	Correct
11.		Tomato	Fresh and Rotten	Fresh and Rotten	Correct
12.		Tomato and Red Onion	Fresh and Rotten	Fresh and Rotten	Correct

13.		Tomato, Eggplant and Red Onion	Fresh and Rotten	Fresh and Rotten	Correct
14.		Eggplant and Red Onion	Fresh and Rotten	Fresh and Rotten	Correct
15.		Tomato and Eggplant	Fresh and Rotten	Fresh and Rotten	Correct

The results of the YOLO-based detection system were compared to the actual freshness status of each sample. Out of the 15 tested samples, 12 were correctly identified by the system, while 3 was incorrectly classified. This results in an overall detection accuracy of:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Samples}} \times 100 = \frac{12}{15} \times 100 = 80\% \quad (2)$$

The vegetable freshness detection system demonstrated an overall accuracy of 80%, correctly identifying 12 out of 15 tested samples. This indicates that the YOLOv5 model is capable of reliably detecting the freshness status of tomatoes, red onions, and eggplants under ideal conditions. While the system performed well in most cases, a small number of misclassifications suggest that improvements in lighting conditions, dataset variety, or model tuning could further enhance its accuracy and reliability for real-world applications.

4. Conclusion

The vegetable freshness detection system using Raspberry Pi and YOLOv5 provides a reliable, real-time solution for classifying vegetables as fresh or rotten. It combines computer vision, embedded hardware, and database storage to automate quality checks. A USB webcam captures images processed by Python scripts on a Raspberry Pi 4, which controls LEDs and buzzers for immediate feedback. YOLOv5, implemented with Torch and supported by OpenCV and RPi.GPIO, delivers fast and accurate results. Detection data is stored in a MySQL database and displayed through a web application for easy monitoring. Designed to run independently, the system suits environments with limited internet access. While effective under controlled lighting, future improvements could include better performance in varied conditions, broader vegetable support, and enhanced mobile access. The current study is limited to three vegetables which are tomatoes, eggplants, and onions and works best under ideal lighting conditions. Early-stage spoilage detection is less accurate, and the relatively small dataset may limit model generalizability. This project builds on existing research and demonstrates practical use of image

processing for automated food quality control, serving as a prototype that can be adapted for future use in retail settings.

Acknowledgement

The authors would like to express the deepest gratitude to the Faculty of Electrical and Electronic Engineering, Universiti Tun Hussein Onn Malaysia for the logistic support.

Conflict of Interest

Authors declare that there is no conflict of interest regarding the publication of the paper.

Author Contribution

The authors confirm contribution to the paper as follows: ***study conception and design:*** *Raja Ummu Sulaim Raja Mohd Azman, Mariyam Jamilah Homam;* ***data collection:*** *Raja Ummu Sulaim Raja Mohd Azman;* ***analysis and interpretation of results:*** *Raja Ummu Sulaim Raja Mohd Azman;* ***draft manuscript preparation:*** *Raja Ummu Sulaim Raja Mohd Azman, Mariyam Jamilah Homam. All authors reviewed the results and approved the final version of the manuscript.*

References

[1] T. Ewen and D. Faour-Klingbeil, "Impact of food waste on society, specifically at retail and foodservice levels in developed and developing countries," *Foods*, vol. 13, p. 2098, 2024, doi: 10.3390/foods13132098.

[2] M. Pulipalupula, S. Patlola, M. Nayaki, M. Yadlapati, J. Das, and B. R. S. Reddy, "Object detection using You Only Look Once (YOLO) algorithm in convolution neural network (CNN)," in *Proc. IEEE 8th Int. Conf. Convergence in Technology (I2CT)*, Lonavla, India, 2023, pp. 1–4, doi: 10.1109/I2CT57861.2023.10126213.

[3] S. Mathe, H. Kondaveeti, S. Vappangi, S. Vanambathina, and N. Kumaravelu, "A comprehensive review on applications of Raspberry Pi," *Computer Science Review*, vol. 52, p. 100636, 2024, doi: 10.1016/j.cosrev.2024.100636.

[4] J. A. Jose, C. Bertumen, M. Roque, A. Umali, J. C. Villanueva, R. TanAi, E. Sybingco, J. L. San Juan, and E. Gonzales, "Smart shelf system for customer behavior tracking in supermarkets," *Sensors*, vol. 24, p. 367, 2024, doi: 10.3390/s24020367.

[5] X. Wang and J. Liu, "Vegetable disease detection using an improved YOLOv8 algorithm in the greenhouse plant environment," Sci. Rep., vol. 14, no. 1, p. 4261, Feb. 2024, doi: 10.1038/s41598-024-54540-9.

[6] M. U. Rehman et al., "Leveraging Convolutional Neural Networks for Disease Detection in Vegetables: A Comprehensive Review," Agronomy, vol. 14, no. 10, p. 2231, 2024, doi: 10.3390/agronomy14102231.

[7] A. Dolatabadian et al., "Image-based crop disease detection using machine learning," Plant Pathology, 2024, doi: 10.1111/ppa.14006.

[8] Y. Zhang et al., "Early detection of verticillium wilt in eggplant leaves by fusing five image channels: a deep learning approach," Plant Methods, vol. 20, no. 1, 2024, doi: 10.1186/s13007-024-01291-3.

[9] Y. R. Pandeya, S. Karki, I. Dangol, and N. Rajbanshi, "Deep Learning based Tomato Disease Detection and Remedy Suggestions using Mobile Application," arXiv preprint, 2023, doi: 10.2310.05929.