Smart Agricultural Technology 12 (2025) 101142

Contents lists available at ScienceDirect

Smart Agricultural Technology

journal homepage: www.journals.elsevier.com/smart-agricultural-technology

An improved small object detection CTB-YOLO model for early detection of tip-burn and powdery mildew symptoms in coriander (*Coriandrum sativum*) for indoor environment using an edge device

Parwit Chutichaimaytar [a], Zhang Zongqi [a], Kriengkri Kaewtrakulpong [b], Tofael Ahamed [c,*]

[a] Graduate School of Science and Technology, University of Tsukuba, 1-1-1 Tennodai, Tsukuba 305-8577, Japan
[b] Department of Farm Mechanics, Faculty of Agriculture, Kasetsart University, Bangkok 10900, Thailand
[c] Institute of Life and Environmental Sciences, University of Tsukuba, 1-1-1 Tennodai, Tsukuba 305-8577, Japan

ARTICLE INFO

Keywords:
Coriander
Tip-burn
Powdery mildew
YOLO
Indoor farming
Disease detection

ABSTRACT

Indoor cultivation of coriander (*Coriandrum sativum*) ensures a steady supply to meet year-round consumer demand; however, tip-burn and powdery mildew significantly challenge indoor coriander production. Early and accurate detection of these symptoms is critical for maintaining yield and quality, yet traditional visual inspection methods are subjective and prone to error. To address this, we developed an enhanced deep learning model, Coriander Tip-Burn YOLO (CTB-YOLO), specifically tailored for detecting small-object symptoms in coriander leaves, emphasizing the reduction of false-positive detections (FP), which could lead to erroneous alerts and unnecessary interventions. We created a novel, annotated dataset comprising 3240 images of tip-burn and 3340 images of powdery mildew symptoms collected under controlled indoor conditions. The CTB-YOLO model incorporates advanced multiscale feature fusion, significantly improving detection accuracy with mean average precision (mAP) scores of 76.1% for tip-burn and 69.3% for powdery mildew, while notably reducing false-positive detections. The model was deployed on an edge computing device integrated with the LINE application, providing real-time notifications to growers for immediate and reliable intervention. This research demonstrates CTB-YOLO's potential as an effective tool for automating disease monitoring in indoor agriculture, enhancing crop health management, and promoting sustainable farming practices.

1. Introduction

The increasing demand for fresh coriander (*Coriandrum sativum*), driven by consumer interest in healthy foods rich in antioxidants and essential oils [1], has spurred the development of controlled indoor farming systems. However, indoor cultivation presents unique challenges, particularly the early detection of diseases like tip-burn and powdery mildew, which significantly affect crop yield and quality [2,3]. Tip-burn is a physiological disorder primarily caused by nutrient imbalances, temperature fluctuations, inadequate water, and poor air circulation, manifesting as leaf browning, scorching, and eventually necrosis if untreated [4–6]. Additionally, poor air circulation combined with elevated humidity increases the risk of powdery mildew caused by *Erysiphe polygoni* DC., characterized by white mycelial colonies on leaf surfaces [7]. Traditional visual inspections for disease detection are subjective, time-consuming, and prone to errors [8,9].

Recent advancements in computer vision and machine learning, particularly the YOLO (You Only Look Once) model, have demonstrated promising results for real-time plant disease detection [10–12]. However, existing YOLO models often have limitations with accurately identifying subtle and small symptoms such as those found on coriander leaves [2,13]. Recent studies have addressed challenges in indoor farming for various crops by leveraging advanced object detection techniques. Abdullah et al. [14] effectively utilized YOLOv8 and Faster R-CNN models for real-time tomato leaf disease detection in greenhouse environments, significantly reducing manual inspection time. Similarly, Hamidon and Ahamed [5] developed a YOLOv5-based system to detect tip-burn in indoor-cultivated lettuce, highlighting the potential of deep learning for early physiological disorder detection but noting the need to reduce false-positive detections. Zhang et al. [15] proposed a YOLOX-based system for cotton disease detection, demonstrating improved accuracy in detecting subtle symptoms at various growth

* Corresponding authors.
E-mail address: tofael.ahamed.gp@u.tsukuba.ac.jp (T. Ahamed).

https://doi.org/10.1016/j.atech.2025.101142
Received 4 April 2025; Received in revised form 26 June 2025; Accepted 27 June 2025
Available online 28 June 2025
2772-3755/© 2025 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

stages. Additionally, Önler et al. [16] demonstrated YOLOv8m's capability to accurately detect powdery mildew symptoms in wheat, emphasizing the benefit of real-time monitoring to prevent disease spread. Temniranrat et al. [17] successfully integrated YOLO-based detection with mobile IoT solutions for rice diseases, providing timely notifications for precise intervention. Although these studies show significant promise, challenges remain, particularly the accurate detection of subtle and tiny disease symptoms on small-leaf crops such as coriander, emphasizing the need for refined methods.

To bridge these gaps, this research introduces:

- To develop a novel dataset of coriander grown indoors for use in an early detection model for stress.
- To develop a new small object detection model for coriander leaves to address tip-burn and powdery mildew symptoms detection with higher accuracy, that overcomes the limitations of the contemporary models.
- To develop a tip-burn and powdery mildew symptoms detection user notification system based on edge devices and smartphone APIs (LINE) for immediate attention.

To provide a solution for the abovementioned research aims, this paper proposes the development and implementation of an early warning detection alert system for tip-burn and powdery mildew in coriander. This system leverages an improved Coriander Tip-Burn YOLO (CTB-YOLO) model specifically trained to identify the unique characteristics of tip-burn and powdery mildew on coriander leaves grown indoors. The system integrates with the LINE application programming interface (API) to deliver real-time notifications to farmers [18,17] whenever tip-burn and powdery mildew symptoms are detected. Empowering farmers with immediate alerts allows them to take swift action, such as adjusting environmental parameters or applying targeted treatments, to minimize the impact of the disease and ensure optimal crop health and yield [19]. This research presents dataset creation, environmental control, model development and validation, and practical application to promote broader adoption in indoor agriculture.

2. Materials and methods

This section provides a detailed description of the research methodology employed to evaluate the performance of the proposed early warning detection alert system for tip-burn and powdery mildew in

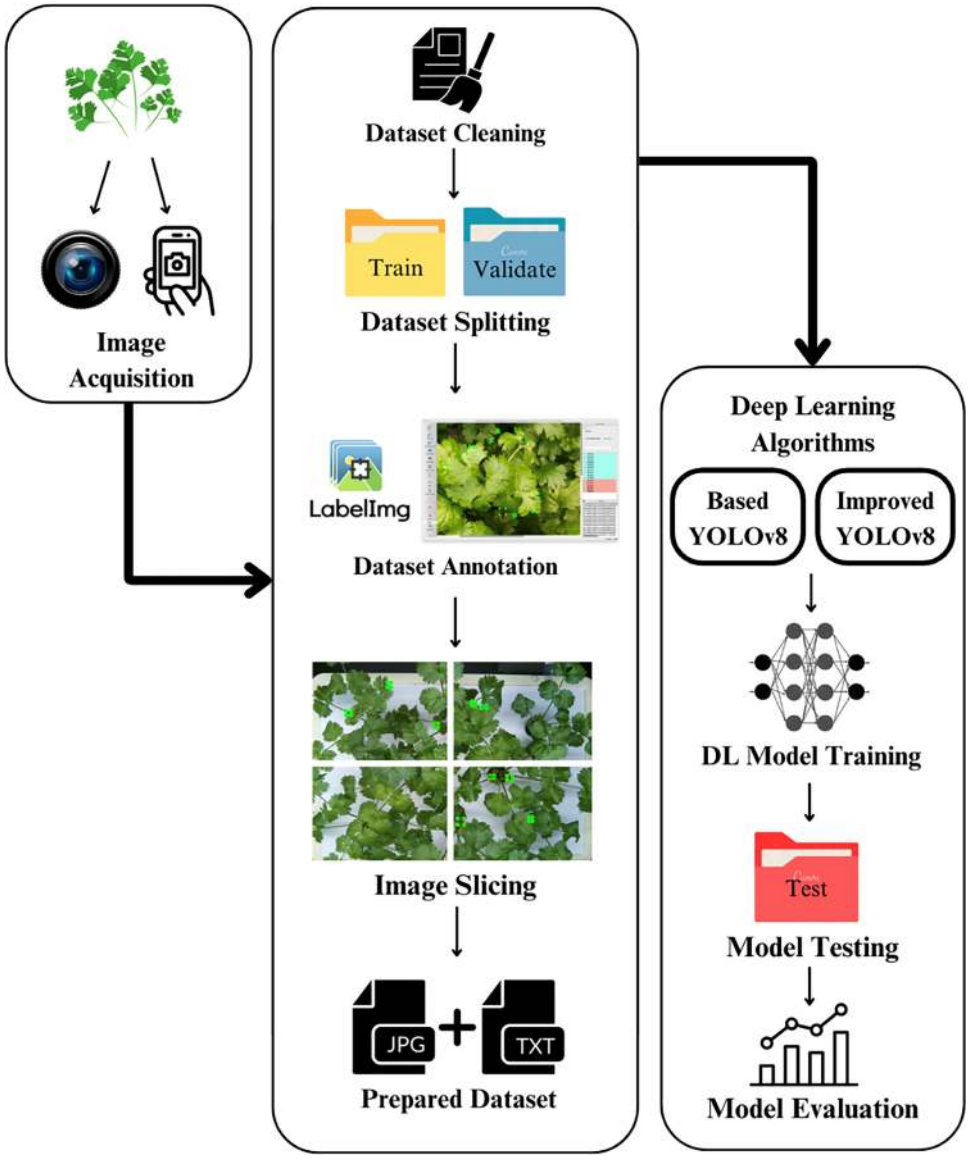


Fig 1. The conceptual framework for detecting Tip-burn and Powdery Mildew in Coriander using DL algorithms.

coriander. The experiment includes steps such as data collection, data preprocessing, model training via YOLO, validation, and evaluation of the prediction results (Fig. 1).

2.1. Cultivation condition

To address the lack of a publicly available dataset on tip-burn and powdery mildew in coriander grown indoors, a new dataset was specifically created by cultivating coriander plants indoors under controlled conditions. As previously noted, various factors, such as inadequate air circulation, contribute to tip-burn in plants grown in indoor farming environments [6]. To investigate this phenomenon, experiments were conducted in a small-scale cultivation room especially designed for research purposes; the room was located at the Bioproduction and Machinery Laboratory, University of Tsukuba, Japan. The experiments took place during the winter season to maintain consistent cooler temperatures, spanning a period from November 2023 to March 2024 (Fig. 2).

The coriander seeds were initially sown in hydroponic sponges for a period of 14 days. Hydroponic sponges provide a sterile and controlled environment for seed germination, allowing researchers to closely monitor the initial growth stage of the coriander plants. After 14 days, the germinated seedlings were subsequently transplanted into a deep-water culture (DWC)-based hydroponic setup. DWC systems were chosen for this experiment because they offer several advantages for controlled environment research, including efficient nutrient delivery, and ease of monitoring nutrient solution concentrations [20]. Six sets of DWC hydroponic systems were constructed specifically for this experiment, each of which was fabricated from food-grade polyvinyl chloride (PVC) trays and pipes. A total of 30 coriander plants, 5 plants per tray, were cultivated across the six DWC systems, providing a statistically robust sample size for the dataset.

The growth cycle of the coriander plants was carried out over two generations to ensure the reproducibility of the results, and to account for any potential variations that might have occurred during a single cultivation cycle. The coriander plants were grown under entirely artificial lighting conditions to eliminate the influence of external sunlight variations. Full-spectrum white LEDs were used as the light source, as they emit wavelengths specifically suited to optimize the photosynthesis process in plants. The intensity of the artificial light was meticulously maintained at a constant level of 9755 lux, PAR 8320.6 $\mu W/cm^3$, and PPFD 365.6 $\mu mol/m^2/s$. The duration of light exposure was also precisely controlled, with a consistent 16-hour period of light followed by 8 hours of darkness. This specific light schedule is commonly used for coriander cultivation because it balances the plants' need for light for photosynthesis with essential rest periods for proper growth and development [21] (Fig. 3).

The coriander plants were cultivated in a solution culture, where their roots were immersed in a specifically formulated hydroponic nutrient solution. The chosen nutrient solution was Hyponica Liquid Fertilizer produced by Kyowa Co., Ltd., Japan. Initially, all the coriander plants were cultivated under normal conditions with a standardized nutrient solution concentration. This standardized concentration range falls within the range of 1.2 to 2.0 mS/cm (milli Siemens per centimeter) [22].

Throughout the entire cultivation period, the temperature within the growth chamber was maintained within a range of 20 to 25 degrees Celsius. This temperature range was optimal for coriander growth, and



Fig 3. Experiment of Indoor Coriander Growing Shelf with Deep Water Culture Method.

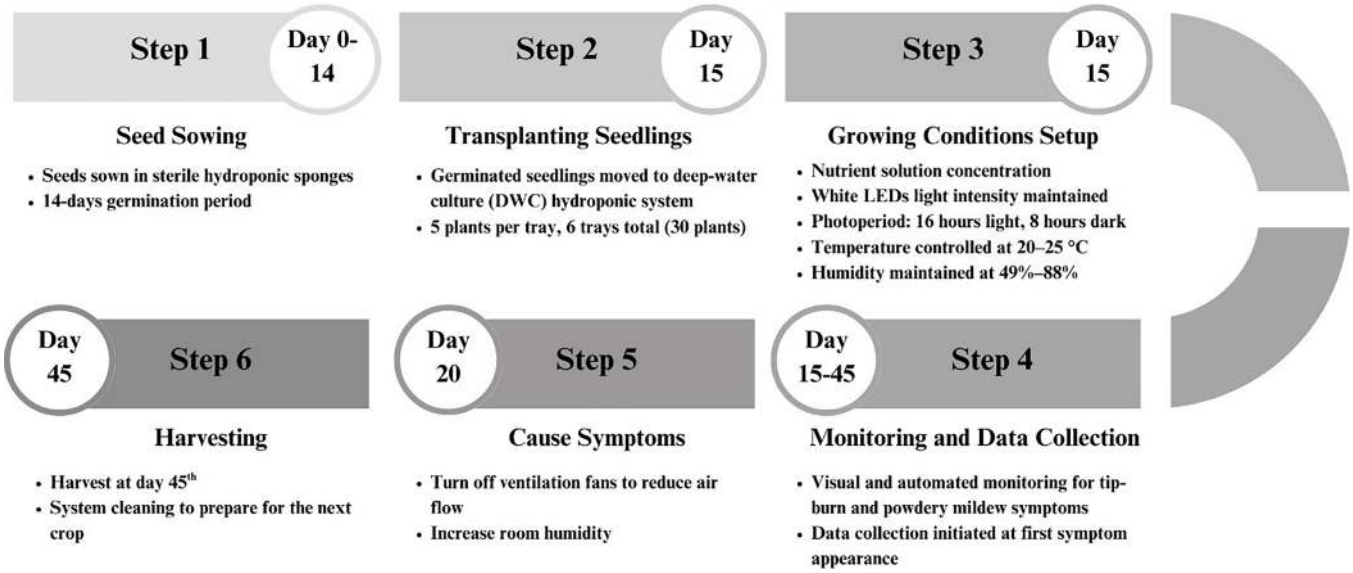


Fig 2. Step-by-step flow chart on the coriander preparation process.

helped ensure consistent plant development [23]. Additionally, the humidity levels were carefully monitored and regulated such that they remained between 49% and 88%. Proper humidity control is essential for plant health, as it can impact processes such as transpiration and nutrient uptake [24]. The trial continued for a designated period until the coriander plants presented visible signs of tip-burn. The development of tip-burn symptoms signaled the successful induction of the target condition and allowed researchers to collect valuable data on the physiological response of the plants to controlled stressors.

2.2. Data collection

Data collection in this study involved monitoring and recording plant images to document occurrences of tip-burn and powdery mildew clearly. A multimodal data collection approach was utilized to comprehensively capture the visual characteristics of these diseases in coriander plants. Two different camera types were employed to enhance the diversity of image resolutions and perspectives, thus improving the robustness and accuracy of the trained detection model. Initially, manual image acquisition was performed using a smartphone camera (OnePlus Nord) with a 3:4 aspect ratio (1080 × 2340 pixels). These manual photographs began upon the first visual observation of symptoms, capturing images directly from above and at various oblique angles throughout the plant growth cycles. Concurrently, a dedicated automated high-resolution image acquisition system using a Raspberry Pi single-board computer coupled with a Pi camera module was implemented. This system provided continuous high-resolution images (16:9 aspect ratio, 3280 × 2464 pixels) every 15 minutes (Fig. 4).

For the tip-burn dataset, a total of 673 original images clearly displaying brown or black necrotic leaf tips were collected from all 30 coriander plants cultivated during the first generation. Subsequently, a second generation of 30 coriander plants was cultivated to collect data specifically on powdery mildew, resulting in 328 original images clearly identifying early-stage symptoms as white mycelial colonies on leaf surfaces. Both datasets included images captured across multiple growth stages, ensuring comprehensive documentation of symptom variability and facilitating the model's capability for early detection (Fig. 5).

2.3. Data preparation

2.3.1. Data labeling

The tip-burn and powdery mildew annotation process utilized LabelImg (https://github.com/HumanSignal/labelImg), an open-source image annotation tool developed with Python and Qt interfaces. Each clearly identifiable disease symptom was meticulously labeled using rectangular bounding boxes. To ensure data quality and annotation accuracy, blurred or indistinct instances were excluded, as they could negatively affect the neural network's performance [25].

This research involved clearly separated datasets for tip-burn and powdery mildew, each with distinct labeling strategies. 673 images with tip-burn symptoms were clearly labeled into two distinct classes based on severity stages:

- "burning": represented early-stage symptoms, characterized by initial yellowing and browning at leaf tips (Fig. 6(a)).
- "burned": represented advanced-stage symptoms, identified by darker brown discoloration, significant leaf-shape deformation, or tissue loss (Fig. 6(b)).

For the powdery mildew dataset, the original collection of 328 images clearly depicting symptoms of Erysiphe spp. (the fungal agent causing powdery mildew). All powdery mildew symptoms were clearly labeled under a single class:

- "fungus": indicated visible white mycelial colonies clearly present on coriander leaf surfaces (Fig. 6(c)).

Each annotated image had corresponding bounding box coordinates clearly indicating the upper-left and lower-right corners, stored in separate text files (.txt format). These annotation files were subsequently used to train and validate detection models. To assess annotation consistency, we conducted an inter-annotator agreement test using a randomly selected subset of 50 images from the tip-burn dataset (Table 1). Two annotators independently labeled the "burning" and "burned" symptoms following the defined annotation protocol. Bounding boxes were matched using an IoU threshold of 0.5. The analysis yielded 139 matched bounding boxes with a Cohen's Kappa coefficient of 0.874, indicating substantial agreement. Unmatched box counts were 52 (Annotator 1) and 84 (Annotator 2).

2.3.2. Data augmentation

Deep learning models require extensive datasets for robust training. To address this limitation in our study, we employed a data augmentation approach to artificially expand our initial and limited dataset [26]. Our data collection utilized a high-resolution camera yielding source images of 3280 × 2464 pixels. To effectively expand our dataset while preserving the fine details of small disease symptoms, which could be lost with standard resizing, we employed image tiling (slicing) as our

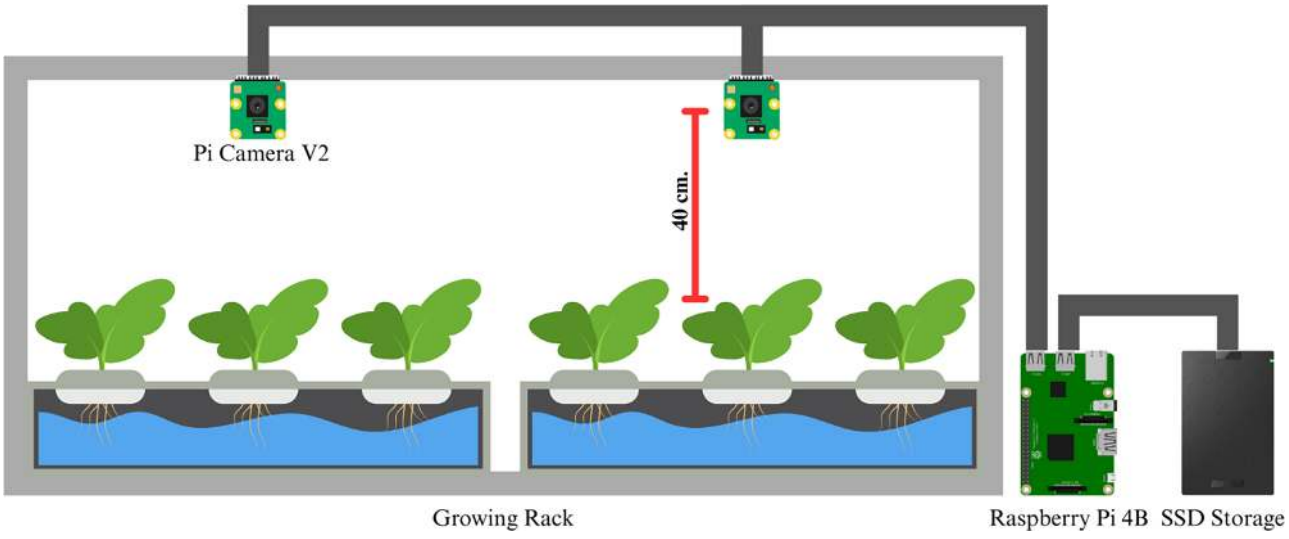


Fig 4. Illustration of Raspberry Pi camera system.



Fig 5. Example of data set images of Coriander in various growth stages.

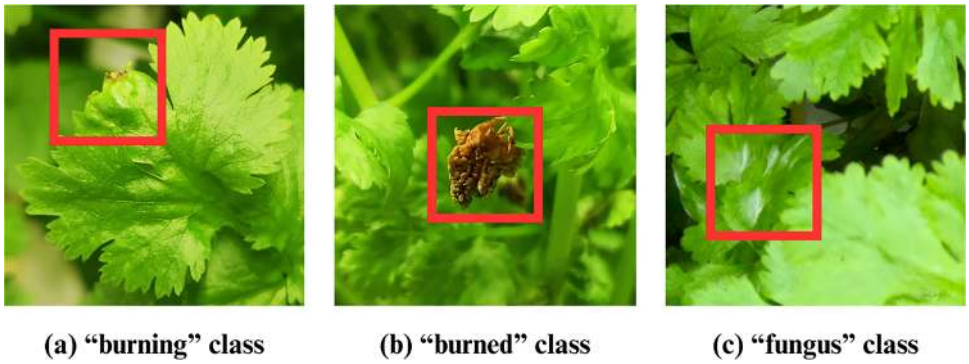


Fig 6. Example of Labeled image. Coriander tip-burn: (a) class "burning" (b) class "burned" and (c) Powdery mildew symptoms: class "fungus".

Table 1
Inter-Annotator Agreement Evaluation Using Cohen’s Kappa.

Metric	Value
Total images evaluated	50
Matched bounding boxes (IoU > 0.5)	139
Unmatched boxes (Annotator 1)	52
Unmatched boxes (Annotator 2)	84
Cohen’s Kappa	**0.874**
Agreement level	Substantial

sole data augmentation technique. This tiling method served the dual purpose of increasing the dataset size while preserving the high-resolution details within each smaller section, which is crucial for small object detection.

The procedure involved partitioning each high-resolution source image into four non-overlapping tiles. This resulted in new, smaller images, each with a resolution of 1640 × 1232 pixels. This tiling process was the method used to expand the initial 673 tip-burn images and 328 powdery mildew images to their final dataset sizes of 3,240 and 3,340, respectively. The final datasets used for training were composed of both the original source images and the newly generated tiles to provide the model with varied scales of contextual information. Prior to model training, all images in the final datasets were resized to the required input resolution of 1080 pixels, as specified in the training hyperparameters.

Across this dataset includes 3,240 images of tip-burn, we annotated 16,975 bounding boxes, with the "burning" class representing early-stage tip-burn and "burned" representing advanced tissue damage. The dataset was manually divided using a fixed hold-out split an approximately 70/20/10 ratio (training/validation/testing), with random shuffling applied before splitting to ensure balanced representation. The powdery mildew dataset remains a separate task and includes 3,340 images with 68,696 bounding boxes for the "fungus" class, captured under similar conditions (Table 2).

Table 2
Summarizes the distribution of annotated bounding boxes for each object class in the tip-burn and powdery mildew datasets.

Dataset	Set	Burning	Burned	Fungus	Total Boxes	Images
Tip-burn	Training	6,640	5,334	–	11,974	2,382
	Validation	1,630	1,262	–	2,892	558
	Testing	672	1,437	–	2,109	300
	Total	8,942	8,033	–	16,975	3,240
Powdery Mildew	Training	–	–	48,084	48,084	2,338
	Validation	–	–	14,002	14,002	668
	Testing	–	–	6,610	6,610	334
	Total	–	–	–	68,696	3,340

2.4. CTB-YOLO

Generally, the performance of plant disease detection algorithms is related to their multiscale feature fusion capabilities. Existing studies usually use feature pyramid structures to fuse and extract image features. However, for the coriander tip-burn and powdery mildew images detection problem in this article, because the disease evidence is so small in the images, and the detection environment is complex, various connections based on pyramid structures cannot guarantee the effective utilization of the correlation between all the pyramid feature maps [27]. Moreover, when the feature images of tiny objects are transmitted within the neural network, two additional key points need to be considered: information transmission efficiency and information loss during the transmission process. For the YOLO series of algorithms, the feature images face the loss of detailed information during the convolution and calculation process between each layer, which is unavoidable [28]. Therefore, for the detection of tiny objects, a more complex neck is not associated with an improvement in algorithm performance. With respect to transmission efficiency, we believe that for the detection of tiny objects in complex environments, there are high demands on both the deep and shallow features of the image, so this requires the algorithm to have the ability to fuse deep and shallow features [29]. As a

classic neural network algorithm structure, the pyramid attention network (PAN) structure has been used in many engineering cases, however, it has shortcomings for the tip-burn detection task in this article. This project needs to integrate the deep and shallow information of the algorithm, which inevitably requires a more complex neck structure to meet the requirements. Moreover, more information transmission at different levels also means more detailed information loss. The pyramid information transmission structure of the PAN structure cannot meet this contradictory demand. Therefore, to improve the detection effect, the CTB-YOLO algorithm was developed, which is based on YOLO v8. The neck and head parts were redesigned so that the algorithm can meet the detection requirements of this task.

2.4.1. Structure of CTB-YOLO

Compared with the YOLO v8 algorithm, the CTB-YOLO algorithm has three improvements. First, the introduction of low-GD and high-GD structures in the neck part improves the information transmission efficiency and feature fusion ability of the algorithm. Second, the introduction of the MSFF module further improves the algorithm's ability to obtain features from the backbone and feature fusion, providing a basis for the redesign of the detection head. The multiscale sequence feature fusion (MSFF) module is divided into two usage methods: the channel attention multiscale feature fusion (CA-MSFF) module, which focuses on feature information acquisition, and the back-flow multiscale feature fusion (BF-MSFF) module, which focuses on feature fusion. Third, this research redesigned the detection head part to effectively improve the algorithm's recall rate. The overall structure of the CTB-YOLO algorithm is shown in Fig. 7.

2.4.2. Channel attention multiscale sequence feature fusion module

To improve the detection capability of the YOLOv8 algorithm, we propose a new channel attention multiscale feature fusion (CA-MSFF) module (Fig. 8). We were inspired by the ASF-YOLO algorithm [30]. In the ASF-YOLO algorithm, they proposed a scale sequence feature fusion module (SSFF) that can better integrate surface features and deep feature information. This module enables the ASF-YOLO algorithm to perform well in cell detection tasks.

The CA-MSFF module in this research constructs a method to obtain enhanced information from multiscale feature maps. These feature maps contain image information obtained at different scales, which can provide richer information for the neck of the YOLO algorithm.

As shown in Fig. 8, inputs 1-3 represent feature maps of different scales. The feature map is first operated by 2D convolution, which is used to change the number of channels of different scales. In this process formula was represented by Eqs.(1).

$$\begin{aligned} Y_2 &= Conv_{1\times1}\left(X_{input2}\right) \in R^{\left(N,C_,H_2,W_2\right)} \\ Y_3 &= Conv_{1\times1}\left(X_{input3}\right) \in R^{\left(N,C_,H_3,W_3\right)} \end{aligned} \tag{1}$$

Next, images of different scales were adjusted to the same clarity (taking the Input3, Input4, and Input3 as examples), and the Upsampling method was used to align all the feature maps to the Input3 level. This is because the Input3 level has higher resolution, and high-resolution images often contain more feature information from smaller detection targets. Then, the Unsqueeze block increases the depth channel. These 4D feature maps are concatenated along the depth dimension. The formula in this process will be expressed as Eqs.(2). For the lp is the low-pass filter. Which could be represented by Eqs.(3).

$$\begin{aligned} Y_2 &= Upsampling(Y_2, lp, 2) \in R^{\left(N,C_,H_1,W_1\right)} \\ Y_3 &= Upsampling(Y_3, lp, 3) \in R^{\left(N,C_,H_1,W_1\right)} \\ \begin{pmatrix} X_{Input1} \\ Y_2 \\ Y_3 \end{pmatrix} &= UpSqueeze \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \end{pmatrix} \in R^{\left(N,C_,1,H,W\right)} \end{aligned} \tag{2}$$

$$I_{LP}(x,y) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} G(u,v) I(x-v, y-v) \tag{3}$$

And Eqs. (4,5) are used in this progress, where $f(i,j)$ represents the acquired 2D feature. and where $G(x,y)$ is smoothed by a Gaussian filter with a standard deviation of ∂.

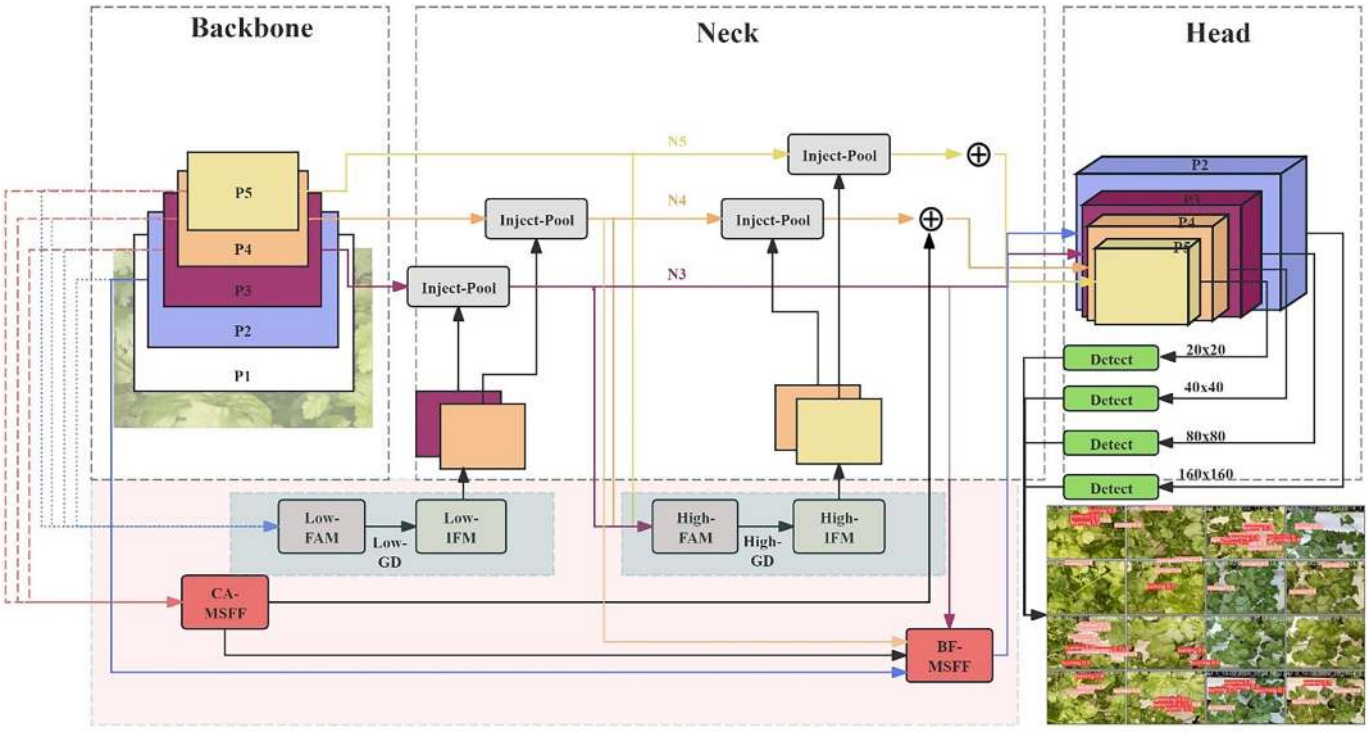


Fig 7. The network structure of the newly proposed CTB-YOLO for small object detection.

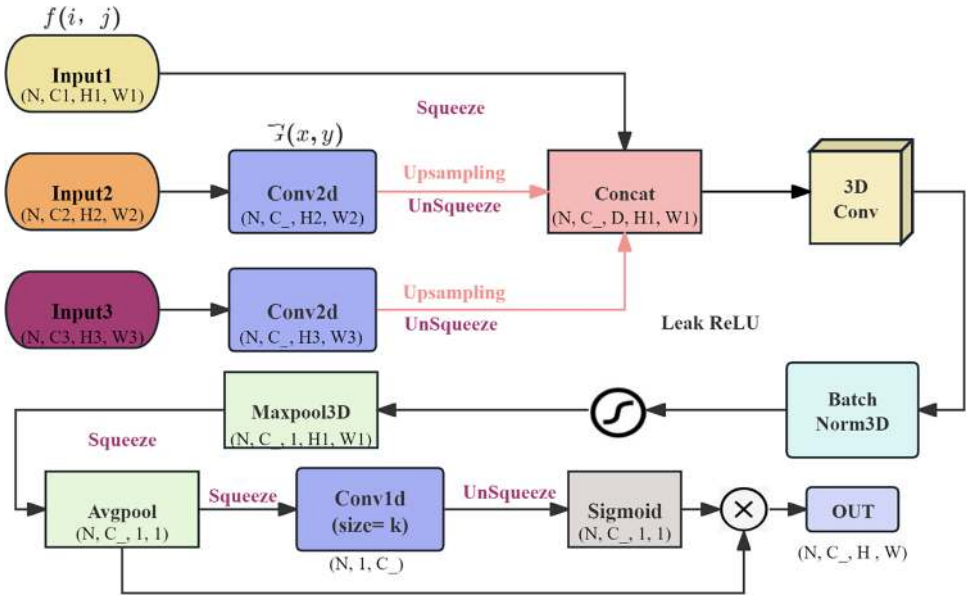


Fig 8. Channel Attention Multi-scale Sequence Feature Fusion (CA-MSFF) module structure to filter unimportant information for the neck of the YOLO algorithm.

$$F(i,j) = \sum_u \sum_v f(i-u, i-v) \times G(u,v) \tag{4}$$

$$G_\partial(x,y) = \frac{1}{2\pi\partial^2} e^{-\frac{(x^2+y^2)}{2\partial^2}} \tag{5}$$

After the block of 3D convolution, normalization, and LeakyReLU. This progress could be represented by Eqs.(6). And the output result enters the channel attention mechanism.

$$\begin{aligned} Y_{concat} &= Concat(Y_1, Y_2, Y_3) \in R^{(N,C_,3,H,W)} \\ Y_ &= Conv3D_{1\times1\times1}(Y_{concat}) \in R^{(N,3C_,3,H,W)} \\ Y_ &= \mathrm{ReLu}(Normal3D(Y_)) \\ Y_ &= Maxpool3D(Y_) \in R^{(N,C_,1,H,W)} \\ Y_ &= Squeeze(Y_)R^{(N,C_,H,W)} \end{aligned} \tag{6}$$

The feature map is first processed by the global AvgPool model for each channel independently. Conv1d aims to capture the interaction of nonlinear cross channels. In this progress, the formula is Eqs.(7).

$$\begin{aligned} Y_{at} &= Avgpool(Y_) \in R^{(N,C_,1,1)} \\ Y_{at} &= transpose(Squeeze(Y_{at})) \in R^{(N,1,C_)} \\ Y_{at} &= transpose(UnSqueeze(Conv1D(Y_{at}))) \in R^{(N,C_,1,1)} \\ Y_{out} &= Y_ * Sigmoid(Y_{at}) \end{aligned} \tag{7}$$

The kernel size of Conv1d can be represented by k, follow the Eqs.(8), which represents the channel range of cross-channel communication.

$$k = \left| \frac{\log_2(C) + b}{\gamma} \right|_{odd} \tag{8}$$

where the C represents the dimension number and the value of b is usually set to 1, and γ is set to 2.

2.4.3. Backflow multiscale feature fusion module

To improve the algorithm's ability to detect tiny objects in complex environments, we need an algorithm to improve the efficiency of transferring feature maps between different layers. The purpose of the back-flow multiscale feature fusion (BF-MSFF) module is to provide a feature fusion platform for feature maps at different levels. Moreover, this module can obtain the feature image of the P2 part from the backbone part to supplement more shallow information. To make better use of shallow features, we need a structure to fuse feature images at different levels. We fuse the features of N3 and N4 from the neck through the MSFF module. This back-propagation structure allows the P2 layer to communicate with multiple layers of information at the same time. In this process, N3 and N4 are processed by the low-GD and high-GD structures to obtain the features of each level of the P3, P4, and P5. Currently, the shallow features from P2 and CA-SSFF are fully integrated to improve the efficiency of information exchange. For the initial stage, the equation is as shown below Eqs.(9). Different from the CA-MSFF module, the BF-MSFF module uses the nearest as the Upsampling function.

$$\begin{aligned} Y_2 &= UnSqueeze(Conv_{1\times1}(X_{P2})) \\ Y_3 &= UnSqueeze(\mathrm{nearest}(Conv_{1\times1}(X_{N3}))) \\ Y_4 &= UnSqueeze(\mathrm{nearest}(Conv_{1\times1}(add(X_{N4}, X_{CA-MSFF})))) \end{aligned} \tag{9}$$

The BF-MSFF structure was designed on the basis of MSFF. In the input part, a fusion mechanism of the N4 and CA-MSFF structures was designed to obtain feature information from more levels. In the output part, an Upsampling structure was added to fuse with the P2 feature map again, then input into the detection head Eqs.(10). The Back Flow Multi-scale Feature Fusion (BF-MSFF) module structure was show in Fig. 9.

$$\begin{aligned} Y_{concat} &= Concat(Y_2, Y_3, Y_4) \in R^{(N,C_,3,H,W)} \\ Y_ &= Conv3D_{1\times1\times1}(Y_{concat}) \in R^{(N,3C_,3,H,W)} \\ Y_ &= \mathrm{ReLu}(Normal3D(Y_)) \\ Y_ &= Maxpool3D\left(Y_\right) \in R^{(N,C_,1,H,W)} Y_{out1} = Squeeze\left(Y_\right)R^{(N,C_,H,W)} Y_{out2} = C2f(Concat(P_2, Y_4)) Y_{out} = add(Y_{out1}, Y_{out2}) \end{aligned} \tag{10}$$

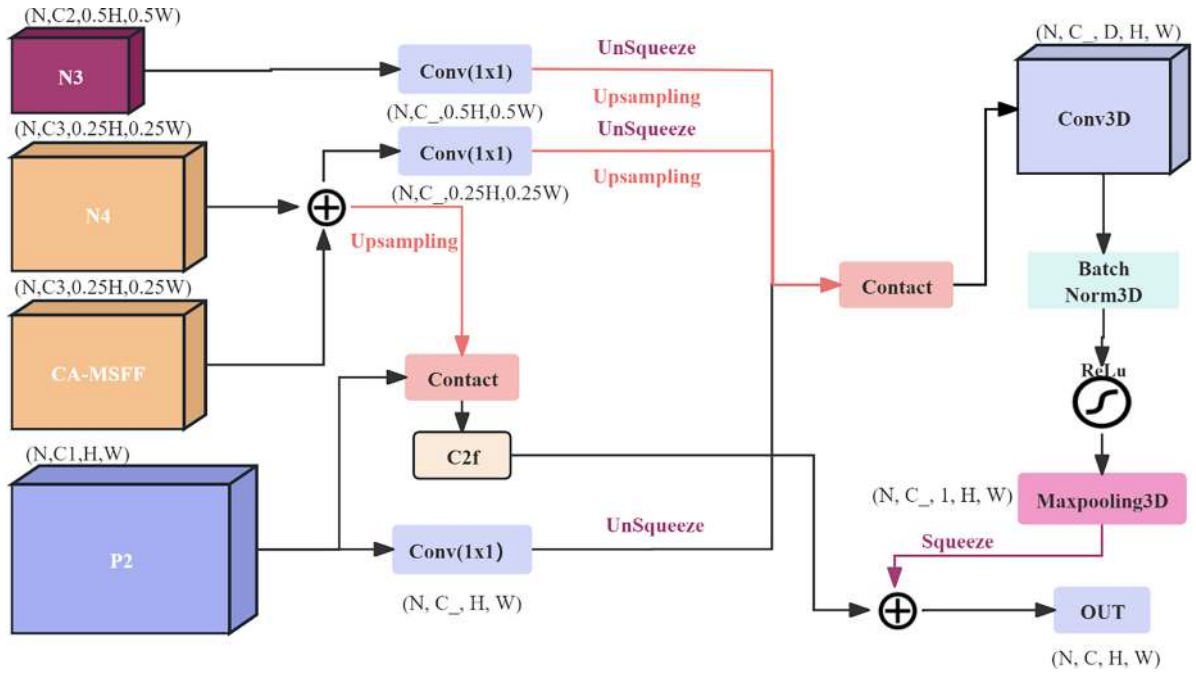


Fig 9. Back Flow Multi-scale Feature Fusion (BF-MSFF) module structure to improve the information transmission efficiency in fusion among multiple layers.

2.4.4. Gather-and-distribute (GD) mechanism

The low-GD and high-GD models are from the GOLD-YOLO algorithm published by Wang C. et al. [29]. This module is a new feature fusion structure that reduces the information loss in cross-layer information fusion. In the traditional pyramid structure, if the P3 layer wants to obtain the information from the P5 layer, it must go through the P4 layer. However, the detailed information is lost during the Upsampling process. This has a negative effect on the detection of tiny objects in complex environments. To improve the algorithm's ability to detect objects at all screen sizes, the GD structure consists of two parts: a low stage and a high stage. Each structure consists of two parts: a feature alignment module (FAM) and an information fusion module (IFM). The structures of low-GD and high-GD are shown in Fig. 10.

2.4.4.1. Structure of the Low stage. For the low-FAM structure, this structure was used to unify input information at different levels. All the features were unified into the smallest feature layer, which ensures the effective aggregation of information. Moreover, more shallow features are retained. Eqs.(11) is as follows:

$$F_{align} = Low_FAM([P2, P3, P4, P5]) \tag{11}$$

For the Low-IMF structure, this structure was used to fuse features at different levels. Eqs. (12) and (13) are as follows:

$$F_{fuse} = RepVGG(F_{align}) \tag{12}$$

$$F_{inj_N3}, F_{inj_N4} = Split(F_{fuse}) \tag{13}$$

where F_{fuse} is processed by a RepVGG (represented visual geometry group) block. Then, it is divided into two features, F_{inj_N3} and F_{inj_N4}, which contain different information through the split module.

2.4.4.2. Structure of the high stage. For the high-FAM structure, this module further processes the $[N3, N4, P2]$ layer information obtained via low-GD. The N3 and N4 layers are processed by the AvgPooling module, and the size is adjusted to the smallest value. The features are fully integrated and reduce the computational redundancy. Eqs.(14) expresses the high-FAM structure as follows:

$$F_{align} = High_FAM([N3, N4, P5]) \tag{14}$$

For the high-IMF structure, the feature is first combined by the Transformer module, Then, processed by Conv1D. The feature is finally separated along the channel dimension through a splitting operation. Eqs.(15) and (16) are expressed as:

$$F_{fuse} = Transformer(F_{align}) \tag{15}$$

$$F_{inj_P4}, F_{inj_P5} = Split(Conv1D(F_{fuse})) \tag{16}$$

2.4.5. Redesigned detection head

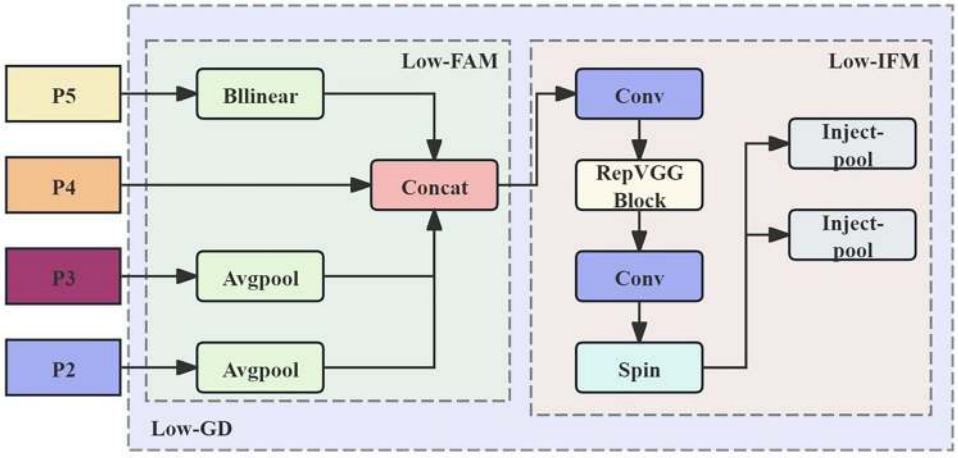
Generally, the smaller the object is, the fewer pixels it occupies, so relatively few features can be collected. Owing to the complex background and the repetition of similar feature information, too much and too little information is collected at the same time. This makes it difficult for the model to fully utilize all the feature information at a size of 80 × 80. Therefore, the CTB-YOLO algorithm adds a 160 × 160 detection head to help generate higher-resolution features, and with the CA-MSFF and BF-MSFF structures, it effectively improves the model's ability to acquire small objects in complex environments.

2.5. Edge device implementation for user notifications

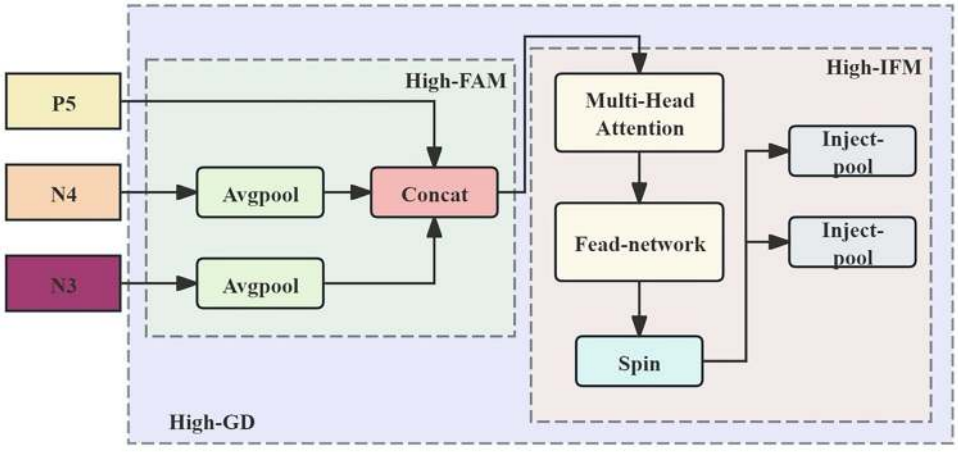
This study presents an automated system for coriander tip-burn detection utilizing a Raspberry Pi, a CTB-YOLO object detection model, and LINE notification integration. The system leverages Python libraries for image acquisition ('PiCamera'), deep learning ('PyTorch' and 'TorchVision'), image processing, and communication ('requests') (Fig. 11).

2.5.1. Integration of hardware and software

The Raspberry Pi served as the core platform because of its low power consumption, GPIO pins, and native camera support. The PiCamera library facilitates straightforward image capture. PyTorch and Torch Hub enable the deployment of trained deep learning models on the Raspberry Pi, making it a powerful and cost-effective tool for edge computing applications.



(a) Structure of Low-GD modules



(b) Structure of High-GD modules

Fig 10. Structure of GD modules to reduce information loss in cross-layer information.

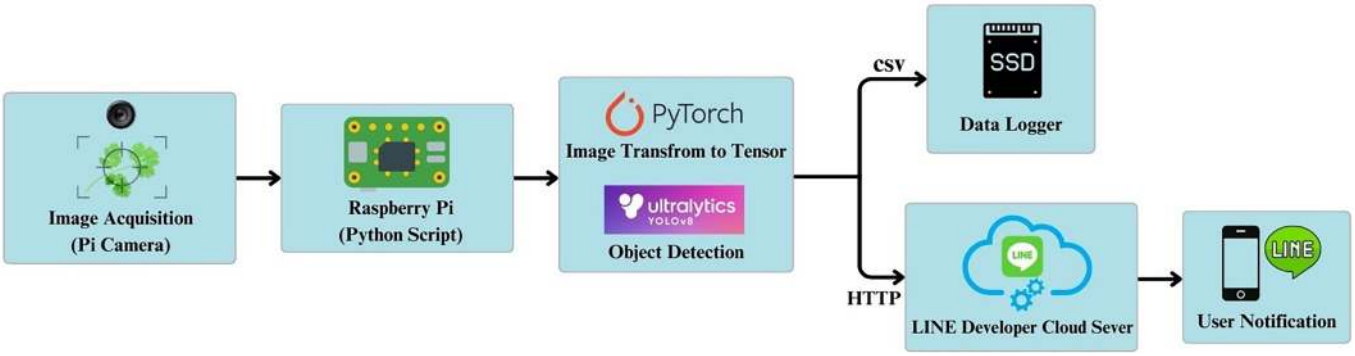


Fig 11. Coriander Tip-Burn and Powdery Mildew Early Warning System for User Notifications.

2.5.2. Initialization and system setup

The process commences by importing the necessary libraries: timing, operating system interaction, camera interfacing, PyTorch for deep learning, and OpenCV for image handling. The LINE notification token for authentication was defined to interact with the LINE API to send notifications via HTTP requests.

2.5.3. Real-time notification via LINE integration

LINE Notify, a service for sending messages and images to the popular messaging application LINE, was employed for real-time notifications [31]. The 'send_line_notification' function uses a LINE identifier token for authentication. Upon object detection, these functions send notifications with the captured image and warning messages to a designated LINE chat group to notify the user [32]. The LINE

Algorithm 1
LINE notification phase.

```
FUNCTION send_line_notification(message, image_path):
    INPUT: message (string), image_path (string)
    SET headers with authorization token
    OPEN image file
    SET payload with message
    SEND POST request to LINE API with headers, files, and payload
    IF response status is 200:
      PRINT success message
    ELSE:
      PRINT failure message with response text
```

Algorithm 2
Detection Process.

```
CLASS ObjectDetection:
  FUNCTION __init__():
    INITIALIZE Picamera2
    CONFIGURE camera
    START camera
    LOAD YOLO model
    SET device to CUDA if available, else CPU
    INITIALIZE last detection time to None
    SET image save path
FUNCTION capture_image():
    CAPTURE image from camera
    OUTPUT: RETURN image (array)
FUNCTION detect_objects(image):
    INPUT: image (array)
    RUN model on image
    OUTPUT: RETURN results (object detection results)
```

notification phase is described in Algorithm 1.

2.5.4. Image capture, object detection and processing loop

The system operates in a continuous loop, capturing images at hourly intervals. The PiCamera is initialized for image capture on the Raspberry Pi. Images are saved with unique timestamps to a solid-state drive (SSD). The images are subsequently loaded and transformed into a tensor via the defined pipeline. Torchvision, a PyTorch library, is used to convert captured coriander images into tensors, a crucial step, as PyTorch models require input data in this format [33,34]. This tensor is then fed into the CTB-YOLO trained model (best.pt) for object detection (Algorithm 2).

The model's output is processed to identify symptoms. The model checks for detections in the first (and only) prediction result. If objects are detected, the program triggers notifications via the 'send_line_notification' function, sending both a text message and the captured image via the LINE application. The processing loop is described in Algorithm 3.

Algorithm 3
Display Loop.

```
FUNCTION run ():
TRY:
  WHILE True:
    GET current time
    IF last detection time is last hour ago:
      CAPTURE image
      DETECT objects in image
      SET symptom detected to False
      FOR each result in results:
        IF symptom detected:
          SET symptom detected to True
          BREAK loop
  IF burning_detected:
    CREATE timestamp
    SAVE image with timestamp
    SEND LINE notification with message and image path
    UPDATE last detection time
  SLEEP 1 second
EXCEPT KeyboardInterrupt:
    PRINT ("Program interrupted. Exiting...")
    STOP Picamera2
```

2.5.5. Process overview

The Python script was implemented on a Raspberry Pi, leveraging the Raspberry Pi's capabilities for code execution, camera interfacing, and internet connectivity, creating a versatile solution for various surveillance and monitoring applications. The overall system captured coriander images via the Pi camera, converted them to tensors, performed object detection with the CTB-YOLO model, and sent notifications through LINE on the basis of the results. This integration of hardware interfacing, deep learning, and cloud communication establishes a robust and automated monitoring system for early detection and notification of symptoms, capitalizing on the power of deep learning, IoT, and messaging technologies. The Raspberry Pi, with its Pi camera and integration capabilities, serves as a compact and efficient platform for this automated detection and notification system (Fig. 12).

3. Results

3.1. Training process

All the models in this research were trained on a high-performance computing platform equipped with an NVIDIA® RTX 3090™ GPU (24 GB) operating on a CUDA version 11.7 computing platform and an Intel® Xeron™ Platinum 8255C CPU 2.50 GHz (48 GB RAM). The PyTorch framework, version 1.11.0, was utilized in conjunction with Python 3.8®. The training hyperparameters are listed in Table 3.

3.2. Evaluation metrics

To quantitatively assess the model's efficacy in detecting tip-burns within a controlled indoor growing environment, we employed established object detection metrics, including complete intersection over union (CIoU) and mean average precision (mAP). These metrics are widely utilized for evaluating the accuracy and overall performance of object detection models [28].

The CIoU (complete intersection over union), a well-established metric for localization accuracy, is a more comprehensive metric than the IoU (intersection over union), as it considers not only the overlap between the predicted and ground-truth bounding boxes but also the distance between their centers and the aspect ratio discrepancy [35]. A higher CIoU value signifies greater precision in pinpointing tip-burn locations. By utilizing the CIoU, we establish true-positives (TPs), false-positives (FPs), and false-negatives (FNs). A TP signifies a correctly identified tip-burn instance, an FP denotes a non-tip-burn region erroneously classified as a tip-burn, and an FN represents a missed tip-burn instance. The CIoU loss is calculated as shown in Eqs.(17)-(19).

$$\Lambda_{CIoU} = 1 - IoU + \frac{p^2(b, b^{gt})}{c^2} + \alpha\nu \tag{17}$$

$$\alpha = \frac{\nu}{(1 - IoU) + \nu} \tag{18}$$

$$\nu = \frac{4}{\pi^2}\left(\left(arctan\frac{w^{gt}}{h^{gt}}\right) - \left(arctan\frac{w}{h}\right)\right) \tag{19}$$

The precision (P) gauges the model's accuracy in detecting tip-burns. It mathematically expresses the proportion of TPs relative to the total number of predictions made by the model. Conversely, recall assesses the model's ability to identify all the existing tip-burns accurately, encompassing FNs or initially missed detections. Eqs.(20) and (21) mathematically represent the calculations for precision and recall, respectively [36].

$$Precision\ (P) = \frac{TP}{TP + FP} \tag{20}$$

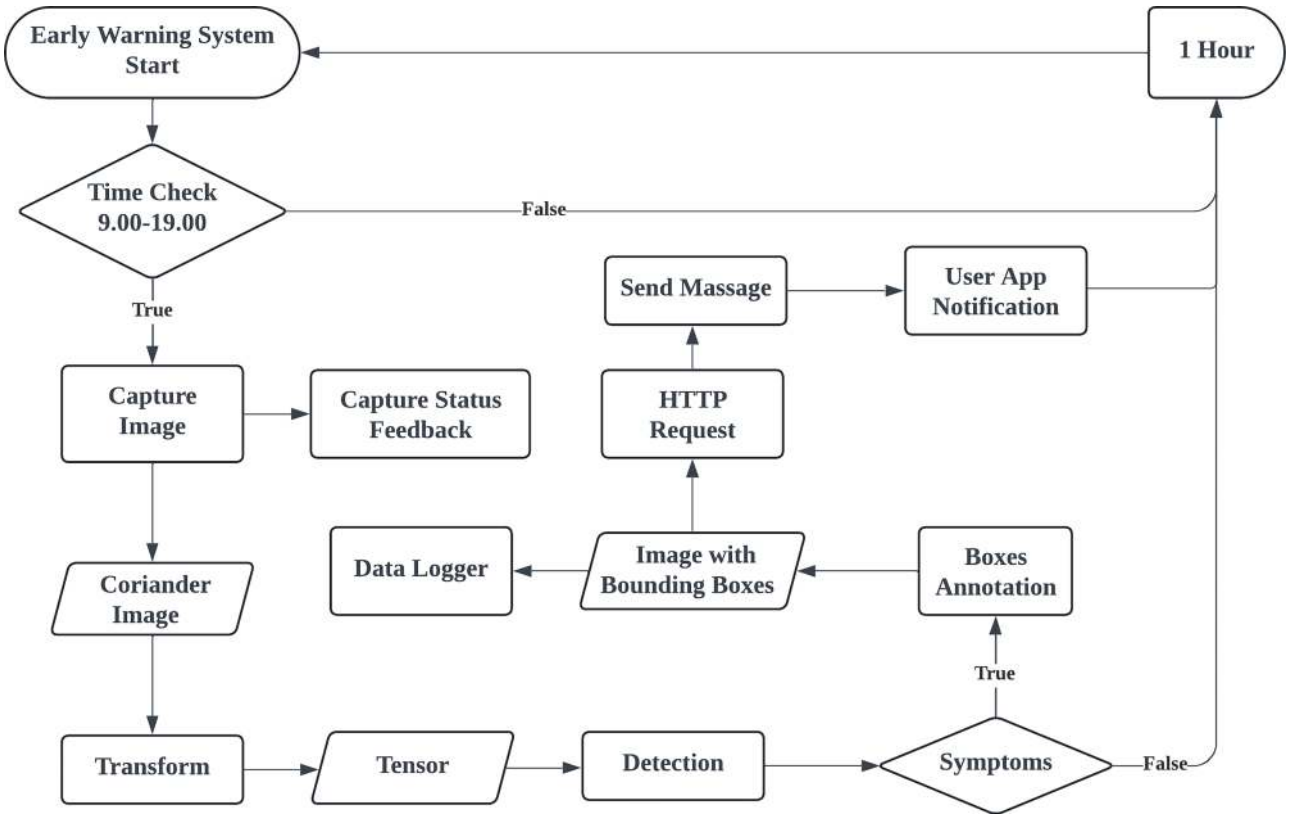


Fig 12. Flow chart of Early Warning System.

Table 3
Hyperparameters for network training for YOLO models.

Parameters	Values
Initial learning rate	0.01
Final learning rate	0.01
Momentum	0.937
Weight decay	0.0005
Batch size	16
Epochs	150
Input image pixel resolution	1080
CIoU	0.65
Optimizer	SGD (Stochastic Gradient Descent)

$$Recall = \frac{TP}{TP + FN} \tag{21}$$

Precision and recall exhibit an inherent trade-off, visualized as a curve through adjustments to the classification threshold for tip-burn symptoms on coriander leaves. The area encompassed by this precision–recall curve represents the average precision (AP) for tip-burn detection within the model. To account for potential variations in tip-burn appearance, we calculate the mean average precision (mAP) across multiple CIoU thresholds via Eq. (22) [37].

$$mAP = \frac{\sum_1^N \int_0^1 P(R)dR}{N} \tag{22}$$

This comprehensive evaluation provides a more robust assessment of the model's overall performance in detecting tip-burns.

3.3. Training and testing for tip-burn symptoms

The dataset images were divided into training, validation, and testing datasets. Five object detection models were evaluated: YOLOv5s, YOLOv8s, ASF-YOLO, MSFF-YOLO, and Gold-YOLO. Additionally, this study proposes improvements to the CTB-YOLO model.

This research trains six deep learning models using identical parameters: an image size of 1080 pixels, a batch size of 16, and 150 epochs. Summarizes the validation performance of the deep learning algorithms at 65% complete intersection over union (CIoU) [38]. Considering precision, recall, and mean average precision (mAP), CTB-YOLO demonstrated a slight edge over the other five models. Specifically, CTB-YOLO achieves 71.0% recall and 76.1% mAP50, outperforming the other models in these metrics. A comparison is illustrated in Table 4.

It is important to detect tip-burn spots on coriander precisely to develop an effective automated tip-burn detection method for coriander growing indoors. To evaluate the detection model, we tested 300 images by using the best-trained weight obtained from each model and evaluated both burn stages ("burning" and "burned") detected separately. The results for each class are listed in Table 5.

While CTB-YOLO does not outperform all compared models across every evaluation metric, it achieves the highest mAP50 (0.701) and highest mAP50-95 (0.362) for the "burning" class, which is the primary target for early detection in this study. The "burning" label represents the earliest visible symptoms of tip-burn, and timely recognition of this stage is crucial for enabling preventative intervention. Although some alternative models, such as MSFF-YOLO or YOLOv8n, show slightly higher performance for the "burned" class, CTB-YOLO strikes an optimal balance by emphasizing early-stage symptom detection while maintaining robust overall accuracy. This validates the proposed architecture's suitability for real-world early warning systems, where minimizing false negatives at the initial stage is more important than post-symptomatic detection.

As illustrated in Fig. 13(a) and Table 6, the models' performance in identifying tip-burn on the coriander via Pi camera images was evaluated. The CTB-YOLO model demonstrated superior accuracy in detecting both early-stage (burning) and late-stage (burned) tip-burns, correctly identifying 9 of the 13 burning spots and 6 of the 9 burned

Table 4
Comparison of precision, recall and mAP between YOLOv5n, YOLOv8n, ASF-YOLO, MSFF-YOLO, Gold-YOLO and CTB-YOLO deep learning algorithms in the training process.

Model	Precision	Recall	mAP50	mAP50-95
YOLOv5n	0.714	0.649	0.721	0.367
YOLOv8n	**0.736**	0.675	0.741	0.383
ASF-YOLO	0.677	0.691	0.728	0.376
MSFF-YOLO	0.689	0.692	0.730	0.368
Gold-YOLO	0.705	0.681	0.741	0.381
CTB-YOLO	0.724	**0.710**	**0.761**	**0.393**

Table 5
Comparison of precision, recall and mAP between YOLOv5n, YOLOv8n, ASF-YOLO, MSFF-YOLO, Gold-YOLO and CTB-YOLO deep learning algorithms in the testing process for each class ("burning" and "burn").

Model	Labels	Precision	Recall	mAP50	mAP50-95
YOLOv5n	Burning	0.724	0.582	0.671	0.332
	Burned	0.692	0.726	0.741	0.474
YOLOv8n	Burning	0.730	0.594	0.684	0.333
	Burned	0.710	**0.763**	0.754	**0.488**
ASF-YOLO	Burning	0.749	0.573	0.679	0.338
	Burned	0.686	**0.763**	0.749	0.482
MSFF-YOLO	Burning	0.748	0.598	0.688	0.331
	Burned	**0.732**	0.744	**0.755**	0.478
Gold-YOLO	Burning	0.770	0.588	0.687	0.343
	Burned	0.709	0.707	0.734	0.473
CTB-YOLO	Burning	**0.775**	**0.601**	**0.701**	**0.362**
	Burned	0.692	0.719	0.730	0.475

spots. While the Gold-YOLO models achieved a greater number of correct detections for 8 burned spots, the performance of CTB-YOLO was notably enhanced by its complete absence of false-positive detections. Given the research's emphasis on early detection, the overall accuracy of CTB-YOLO, with a reduced likelihood of false alarms, makes it a promising candidate for real-world applications in tip-burn monitoring.

3.4. Training and testing for powdery mildew symptoms

Building upon our previous experiment, this research explores the broader applicability of our improved CTB-YOLO algorithm for small object detection. Specifically, we investigated its performance in detecting powdery mildew on coriander leaves (Fig. 13(b)). Detecting powdery mildew presents a significant challenge due to its subtle visual characteristics. However, our proposed CTB-YOLO model demonstrates a notable improvement over the baseline YOLOv8n model. Specifically, CTB-YOLO achieved mAP50 of 69.3% compared to 64.4% for YOLOv8n. Detailed performance metrics are presented in Table 7, while illustrative detection results are shown in Table 8.

3.5. Ablation studies

To isolate the contribution of individual components within CTB-YOLO, we conducted a comprehensive ablation study (Table 9). Starting from the YOLOv8 baseline, integrating the additional 160 × 160 detection head alone improved mAP50-95 to 0.385. The CA and BF modules individually enhanced precision and recall, respectively, while the MSFF module modestly improved overall feature learning. The full CTB-YOLO model, which integrates CA, BF, MSFF, and Extra Head, demonstrates a synergistic effect, achieving the highest mAP50 (0.730) and competitive mAP50-95 (0.382). While this configuration increases inference latency and computational complexity (16.96 GFLOPs), the performance improvements validate the architectural trade-offs.

To evaluate potential redundancy and contribution of MSFF within the CA and BF architecture variants, we conducted a comparative analysis across five model configurations. Results are reported in Table 10. While MSFF alone provided marginal improvement over the baseline YOLOv8, BF and CA showed complementary benefits in boosting recall and precision respectively. The full CTB-YOLO model, which integrates MSFF, CA, and BF, achieved the best mAP50 (0.730) and mAP50-95 (0.382), albeit with increased complexity (GFLOPs: 16.96). This analysis validates the synergistic effect of combining CA and BF with MSFF, while also highlighting the trade-off in model size and inference latency.

To justify the selection of ASF-YOLO's Conv1D-based attention mechanism, we conducted a comparative study against baseline YOLOv8n, CBAM-integrated YOLO, and CTB-YOLO. Table 11 presents the evaluation of key metrics including model parameters, GFLOPs, inference latency, and detection performance. ASF-YOLO achieved a strong trade-off between accuracy (mAP50 = 0.710) and efficiency, with low latency (4.00 ms) and reduced GFLOPs (8.63), outperforming CBAM in speed and model size while offering similar precision. These findings support our design choice to adopt the Conv1D-based attention mechanism, particularly in scenarios requiring a lightweight yet accurate deployment strategy.

3.6. Deployment performance evaluation results

To assess the real-world deploy ability of the CTB-YOLO model, we benchmarked inference performance on a Raspberry Pi 4 Model B, comparing it with other YOLO-based variants, including ASF-YOLO, GOLD-YOLO, MSFF-YOLO, YOLOv5n, and YOLOv8n. Table 12 summarizes the results.

CTB-YOLO achieved a stable inference rate of 1.52 FPS, with moderate power consumption (from 0.860V to 0.926V) and a peak temperature of 41.3°C, indicating safe thermal operation. The RAM usage was 266.96 MB, and the model size on disk was 8.07 MB, reflecting an efficient architecture that balances accuracy with edge-device constraints. Compared to larger or slower variants like GOLD-YOLO (1.09 FPS, 11.88 MB), CTB-YOLO demonstrates superior practicality for low-

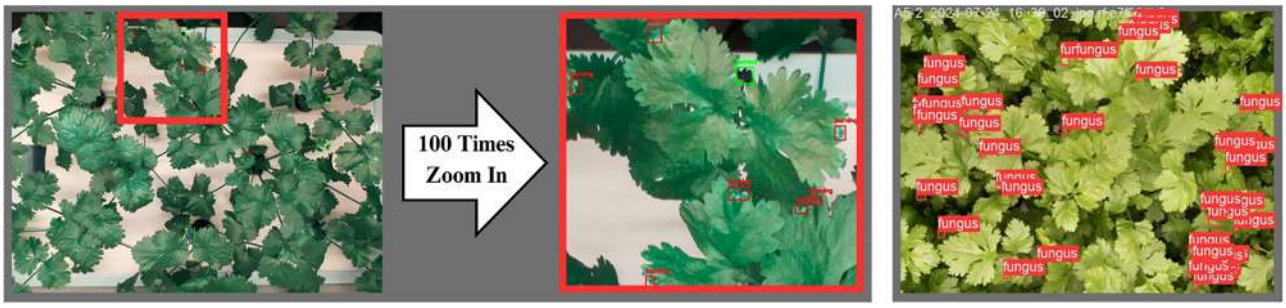


Fig 13. Detection results using the CTB-YOLO model. (a) Tip-burn detection (b) Powdery mildew detection.

Table 6
Summary of the detection results of six algorithms on the sample image.

Model	Labels	True Positive	False Negative	False Positive
YOLOv5n	Burning	7	6	1
	Burned	6	3	-
YOLOv8n	Burning	6	7	1
	Burned	6	3	-
ASF-YOLO	Burning	5	8	1
	Burned	7	2	1
MSFF-YOLO	Burning	5	8	1
	Burned	7	2	1
Gold-YOLO	Burning	6	7	-
	Burned	8	1	-
CTB-YOLO	Burning	**9**	**4**	**-**
	Burned	6	3	-

Table 7
Comparison of precision, recall and mAP between YOLOv8n and CTB-YOLO deep learning algorithms in detection models of powdery mildew.

Model	Precision	Recall	mAP50	mAP50-95
YOLOv5n	0.694	0.580	0.622	0.272
YOLOv8n	0.741	0.581	0.644	0.279
ASF-YOLO	0.694	0.594	0.630	0.270
MSFF-YOLO	0.699	0.595	0.631	0.273
Gold-YOLO	0.711	0.590	0.635	0.271
CTB-YOLO	**0.751**	**0.649**	**0.693**	**0.307**

power agricultural monitoring.

To evaluate the deployment efficiency of the proposed CTB-YOLO model on a real-world edge device, we conducted experiments on a Raspberry Pi 4 Model B using both the full-precision (FP32) and quantized (8-bit) versions exported in TorchScript format. The results are summarized in Table 13.

The full-precision CTB-YOLO model achieved an inference time of 6.438 seconds per image, yielding an estimated 0.16 frames per second (FPS), with moderate CPU and memory usage (~58.9% CPU and 139.7 MB RAM). To further investigate optimization potential, we applied dynamic quantization using 'PyTorch's quantize_dynamic()' method. However, due to the convolution-heavy structure of YOLO-based architectures—which are not significantly impacted by linear-only quantization—the quantized model exhibited no improvement in inference speed. In fact, the quantized version slightly increased in size (16.33 MB) and RAM usage (159.3 MB), while reducing FPS to 0.15. These results indicate that while quantization may benefit transformer- or MLP-heavy models, it offers limited gains for convolution-dominant architectures like CTB-YOLO. Therefore, our optimization strategy focuses on architectural streamlining rather than quantization for deployment on resource-constrained devices.

Table 8
Summary of the detection results of six algorithms on the sample image.

Model	True Positive	False Negative	False Positive
YOLOv5n	35	21	-
YOLOv8n	35	21	-
ASF-YOLO	33	23	2
MSFF-YOLO	33	23	1
Gold-YOLO	36	20	1
CTB-YOLO	**38**	**18**	**-**

4. Discussion

The detection of plant stress and disease is crucial not only for outdoor plants but also for indoor-grown vegetation. While the primary advantage of indoor cultivation is the potential for accelerated growth and faster yields, it also presents drawbacks, such as the occurrence of tip-burn and powdery mildew on leaves, which can be a subtle and early-stage issue. Farmers must be proactive in identifying the underlying causes, which may include nutrient imbalances, fluctuations in temperature and humidity, or inadequate air circulation.

4.1. Evaluation of coriander datasets

To contribute to research on indoor coriander cultivation, a novel dataset was established. This dataset was specifically considered by cultivating coriander, as it was compiled from 5 batches of plant growth in an indoor environment, where tip-burn and powdery mildew were induced by limiting airflow to the growing shelves. To comprehensively capture plant development, daily images were collected at 15-minute intervals via both Pi cameras and mobile phone cameras. This multi-angle approach allowed for a rich and diverse dataset, providing valuable insights into the characteristics of coriander plants throughout their growth cycle. The main challenge in this research was the accurate labeling of tip-burn and powdery mildew spots, which can be subjective and prone to human error, especially given the small size and very early stages of symptoms. To address this, tip-burn stages were divided into "burning" and "burned" categories, and powdery mildew was "fungus". While the data preparation process may have been affected by human bias and error, the findings of this study provide valuable insights into the potential of deep learning techniques for the early detection of plant stress and disease in indoor farming environments. Future research investigations need to explore the physical characteristics of tip-burns and powdery mildew at different stages, which could be facilitated through image processing techniques.

4.2. Small objects detection model (CTB-YOLO)

For coriander small tip-burn detection in complex background environments, we developed the CTB-YOLO detection algorithm on the basis of the YOLOv8 algorithm. We believe that the pyramid structure used in the neck part of the original YOLOv8 algorithm structure cannot effectively address the task of detecting small objects in complex backgrounds [39,40]. The CTB-YOLO algorithm improved the information transfer capability between different modules and, at the same time, enhanced the fusion efficiency of the deep information and the shallow information, reduced the loss of detail information in the cross-layer propagation of the features, and enhanced the algorithm performance [41] in detecting tiny objects [38]. This paper focuses on six different deep learning models based on the YOLO one-stage detector, including YOLOv5, YOLOv8, ASF-YOLO, MSFF-YOLO, Gold-YOLO, and the

Table 9
Ablation Study of CTB-YOLO Architectural Components.

Model	Precision	Recall	mAP50	mAP50-95	Speed(ms)	Params	GFLOPs
YOLOv8 (base)	0.6912	0.6736	0.726	0.373	3.39	3.01M	8.2
CA-only	0.6986	0.6559	0.706	0.357	4.45	3.04M	8.4
BF-only	0.7076	0.6695	0.727	0.373	5.97	2.92M	12.26
MSFF-only	0.6672	0.6737	0.712	0.359	3.65	3.04M	8.4
Extra Head	0.7086	0.6672	0.73	**0.385**	6.28	2.93M	12.36
CA + BF	0.6967	0.6734	0.726	0.379	5.83	2.92M	12.27
CA + Extra Head	0.7082	0.6762	0.727	0.378	5.71	2.96M	12.57
CTB-YOLO	**0.7078**	0.6609	**0.73**	0.382	7.29	3.88M	16.96

Table 10
Redundancy and Contribution Analysis of MSFF Module Across Variants.

Model	mAP50	mAP50–95	Precision	Recall	Params	GFLOPs	Speed (ms)
YOLOv8	0.726	0.373	0.691	0.674	3.01M	8.195	3.386
MSFF	0.712	0.359	0.667	0.674	3.04M	8.403	3.65
CA+MSFF	0.706	0.357	0.699	0.656	3.04M	8.404	4.449
BF+MSFF	0.727	0.373	0.708	0.67	2.92M	12.264	5.968
CTB-YOLO	**0.730**	**0.382**	**0.708**	0.661	3.88M	16.962	7.292

Table 11
Comparison between ASF-YOLO (Conv1D-based attention) and other attention-based YOLO variants in terms of accuracy and computational cost.

Model	Params	GFLOPs	Speed (ms)	Precision	Recall	mAP50	mAP50–95
YOLOv8n	3.01M	8.2	3.39	0.691	0.674	0.726	0.373
ASF-YOLO	3.05M	8.63	4	0.664	0.669	0.71	0.365
CBAM	5.95M	14.64	13.97	0.694	0.64	0.714	0.363
CTB-YOLO	3.88M	16.96	7.29	0.708	0.661	0.73	0.382

CTB-YOLO model, which was improved as part of this research. The results indicate that the CTB-YOLO model outperforms the other models, achieving the highest mean average precision (mAP50) of 76.1% in training. The CTB-YOLO model exhibited exceptional accuracy in detecting tip-burns in the coriander and to demonstrate the efficacy of small objects detection, we extended the application of our improved CTB-YOLO algorithm to detect powdery mildew on coriander leaves. CTB-YOLO achieved a mAP50 of 69.3%, a 5% improvement over the 62.0% achieved by the traditional YOLOv8 model. Notably, the model achieved a perfect accuracy rate, correctly identifying all instances of tip-burn without any false-positives (FPs). This error-free performance underscores the model's reliability and potential for practical application in precision agriculture and plant health monitoring [42]. A comparison of Hamidon & Ahamed's (2022) study on tip-burn detection in indoor lettuce via YOLOv5 revealed several false-positives and misidentifications, even though the accuracy reached 84.1% mAP50. We believe that the CTB-YOLO algorithm has potential and achieves better performance in terms of feature fusion. Meanwhile, to strengthen the deployment capability of the algorithm for edge computing devices, we will also continue to improve the structure of the algorithm in a lightweight way to improve the computational speed while guaranteeing the detection capability.

4.3. Implementation of an early warning system for coriander tip-burns and powdery mildew and user notifications

The study indicates that the enhanced CTB-YOLO model, when integrated with a Raspberry Pi board and a LINE notification system, still requires improvements in certain areas, such as camera quality. However, this system has the potential for application in commercial indoor farming because of its advantages of low investment and accuracy compared with those of conventional computing devices. The notification system through the LINE application allows direct communication with the user, enabling prompt and accurate interventions for the early detection of tip-burn and powdery mildew. These interventions may include modifying the growing environment with suitable humidity, temperature, lighting, and air movement, as well as providing additional calcium nutrients to plants in indoor farming systems. Notifications are transmitted directly to LINE users via the LINE Notify API within designated group chat rooms. This process is facilitated through a LINE token ID uniquely assigned to each user by the LINE Corporation for personal use.

5. Conclusions

This research aimed to develop a computer vision-based method for the early detection of tip-burn and powdery mildew in indoor-cultivated coriander. Tip-burn, a physiological disorder caused by a calcium deficiency, and Powdery Mildew caused by fungal pathogens pose a significant challenge to coriander production in controlled environments. Owing to the small size and clustered nature of tip-burn and powdery mildew symptoms, manual inspection is often not performed. To address this issue, we explored the adaptation of the CTB-YOLO algorithm, a modification based on YOLOv8, for tip-burn and powdery mildew detection. A key contribution of this research was the creation of a new image dataset specifically for coriander, a new improved model for detection and edge device implementation that is outlined as follows:

- **Novel Datasets.** This dataset, comprising images from indoor growth coriander batches, provides a valuable resource for training and evaluating deep learning models. The dataset's diversity and high-quality annotations were essential for ensuring the model's ability to accurately detect tip-burn and powdery mildew at various growth stages.
- **Results for improved CTB-YOLO.** Using this new dataset, we trained, validated, and tested deep learning models. CTB-YOLO consistently outperformed other YOLO-based algorithms, achieving a mean average precision (mAP50) of 76.1% on the test set. The high detection accuracy without error detected with CTB-YOLO demonstrated its potential as an effective early warning system for tip-burn and achieved 69.3% for powdery mildew detection.

Table 12
Edge Deployment Comparison of YOLO Variants on Raspberry Pi 4B.

Model	FPS	Power (V)	Temp (°C)	RAM (MB)	Size (MB)
CTB-YOLO	1.52	0.860 → 0.926	37.0 → 41.3	266.96	8.07
ASF-YOLO	1.69	0.860 → 0.926	37.0 → 40.9	263.8	6.11
GOLD-YOLO	1.09	0.860 → 0.926	37.4 → 41.3	284.28	11.88
MSFF-YOLO	1.72	0.860 → 0.926	37.4 → 40.4	262.15	6.08
YOLOv5n	1.7	0.860 → 0.926	37.4 → 39.9	259.52	5.08
YOLOv8n	1.72	0.860 → 0.926	37.0 → 38.9	262.65	6.02

Table 13
Deployment Performance Comparison of CTB-YOLO (FP32) and Quantized Models on Raspberry Pi.

Model	Format	Size (MB)	Inference Time (s)	FPS	CPU Usage (%)	RAM Usage (MB)
CTB-YOLO	TorchScript (FP32)	16.31	6.438	0.16	~58.9	139.7
Quantized	TorchScript (8-bit)	16.33	6.541	0.15	~59.8%	159.3

- **Edge Device Implementation.** To facilitate practical implementation, we explored the deployment of CTB-YOLO on low-cost edge devices such as Raspberry Pi. Integrating the system with mobile phone notifications (LINE) can enable growers to monitor their plants remotely and respond promptly to tip-burn and powdery mildew occurrences. To further increase the accuracy of tip-burn and powdery mildew detection, future research should focus on expanding the dataset size and improving the precision of annotations. Additionally, a deeper understanding of the physiological characteristics associated with different stages of tip-burn and powdery mildew is necessary to refine the model's ability to detect early-stage symptoms. The CTB-YOLO algorithm holds significant promise, and further optimization of the feature fusion component could lead to an improved performance.

To facilitate deployment on resource-constrained edge devices, efforts need to be directed toward developing more lightweight and computationally efficient versions of the algorithm without compromising detection accuracy. One limitation of the current study is the lack of spectral variability in the dataset. All data were collected under a consistent lighting setup, which may affect the generalizability of the model when deployed in different environments or using alternative horticultural lighting. Future research will address this limitation by expanding the dataset to include images captured under a variety of growing lights, such as red-blue LED, full-spectrum white, and far-red enriched lighting. This effort aims to improve model adaptability and ensure reliable performance under diverse cultivation scenarios. This research contributes to the development of automated detection systems for indoor farming environments, particularly for the early identification of tip-burn and powdery mildew symptoms. Future studies can explore the integration of advanced plant monitoring technologies, such as automation and robotics, to create more comprehensive and efficient indoor farming solutions.

CRediT authorship contribution statement

Parwit Chutichaimaytar: Writing – original draft, Methodology, Formal analysis, Data curation, Conceptualization. **Zhang Zongqi:** Validation, Methodology, Data curation. **Kriengkri Kaewtrakulpong:** Writing – review & editing, Resources, Formal analysis. **Tofael Ahamed:** Writing – review & editing, Resources, Investigation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The authors gratefully acknowledge the support of the Bioproduction and Machinery Laboratory and the University of Tsukuba for providing the necessary facilities to conduct this research. Additionally, the authors would like to express their sincere appreciation to the Ministry of Education, Culture, Sports, Science, and Technology (MEXT), Japan, for awarding scholarships that enabled this research to be pursued at the University of Tsukuba.

Data availability

Data will be made available on request.

References

[1] H. Lin, K. Lin, M. Yang, H.C. Nguyen, H. Wang, H. Huang, M. Huang, Physiological responses and antioxidant properties of coriander plants (Coriandrum sativum L.) under different light intensities of red and blue lights, Sci. Rep. 12 (1) (2022) 1–14, https://doi.org/10.1038/s41598-022-25749-3.
[2] R. Chapaneri, M. Desai, A. Goyal, S. Ghose, S. Das, Plant Disease Detection: A Comprehensive Survey. 2020 3rd International Conference on Communication System, Computing and IT Applications (CSCITA), 3-4 April, IEEE, Mumbai, 2020, https://doi.org/10.1109/cscita47329.2020.9137779.
[3] D. Ganeva, L. Filchev, E. Roumenina, R. Dragov, S. Nedyalkova, V. Bozhanova, Winter Durum wheat disease severity detection with field spectroscopy in phenotyping experiment at leaf and canopy level, Remote Sens. (Basel) 16 (10) (2023) 1–26, https://doi.org/10.3390/rs16101762.
[4] J. Ertle, Tipburn Management Through Controlled Environment for Indoor Vertical Farm Lettuce Production. PhD Dissertation, The Ohio State University, Columbus, Ohio, USA, 2023, pp. 1–158, https://doi.org/10.13140/RG.2.2.10282.39362.
[5] M.H. Hamidon, T. Ahamed, Detection of tip-burn stress on lettuce grown in an indoor environment using deep learning algorithms, Sensors 22 (19) (2021) 1–18, https://doi.org/10.3390/s22197251.
[6] Y. Zhang, M. Kacira, L. An, A CFD study on improving air flow uniformity in indoor plant factory system, Biosyst. Eng. 147 (2016) 193–205. https://doi.org/10.1016/j.biosystemseng.2016.04.012.
[7] G. Kumawat, D. Gothwal, R. Kunwar, A. Shivran, P. Kumawat, A. Meena, Screening of powdery mildew (Erysiphe polygoni DC.) tolerance in coriander (Coriandrum sativum L.) germplasm, Deleted J. 10 (4) (2021) 1112–1116. https://www.thepharmajournal.com/archives/2021/vol10issue4/PartP/10-4-137-670.pdf.
[8] J. Boulent, S. Foucher, J. Théau, Convolutional neural networks for the automatic identification of plant diseases, Front. Plant Sci. 10 (2019) 1–15, https://doi.org/10.3389/fpls.2019.00941.
[9] A. Jafar, N. Bibi, R.A. Naqvi, D. Jeong, Revolutionizing agriculture with artificial intelligence: plant disease detection methods, applications, and their limitations, Front. Plant Sci. 15 (2024) 1–20, https://doi.org/10.3389/fpls.2024.1356260.
[10] Redmon, J., Santosh, D.H.H., Ross, G., Farhadi, A. (2015). You only look once: unified, real-time object detection. arXiv preprint arXiv:1506.02640. https://doi.org/10.48550/arxiv.1506.02640.
[11] Redmon, J., Farhadi, A. (2018). YOLOv3: An incremental improvement. arXiv preprint arXiv:1804.02767. https://doi.org/10.48550/arXiv.1804.02767.
[12] A.M. Roy, R. Bose, J. Bhaduri, A fast accurate fine-grain object detection model based on YOLOv4 deep neural network, Neural Comput. Appl. 34 (2022) 3895–3921, https://doi.org/10.1007/s00521-021-06651-x.
[13] M.J. Soeb, M.F. Jubayer, T.A. Tarin, M.R. Al Mamun, F.M. Ruhad, A. Parven, N. M. Mubarak, S.L. Karri, I.M. Meftaul, Tea leaf disease detection and identification based on YOLOv7 (YOLO-T), Sci. Rep. 13 (1) (2023) 1–16, https://doi.org/10.1038/s41598-023-33270-4.
[14] A. Abdullah, G.A. Amran, S.M. Tahmid, A. Alabrah, A. Ali, A deep-learning-based model for the detection of diseased tomato leaves, Agronomy 14 (7) (2024) 1–13, https://doi.org/10.3390/agronomy14071593.
[15] Y. Zhang, B. Ma, Y. Hu, C. Li, Y. Li, Accurate cotton diseases and pests detection in complex background based on an improved YOLOX model, Comput. Electronics Agric. 203 (2022) 107484. https://doi.org/10.1016/j.compag.2022.107484.
[16] E. Önler, N.D. Köycü, Wheat powdery mildew detection with YOLOv8 object detection model, Appl. Sci. 14 (16) (2024) 7073, https://doi.org/10.3390/app14167073.
[17] P. Temniranrat, K. Kiratiratanapruk, A. Kitvimonrat, W. Sinthupinyo, S. Patarapuwadol, A system for automatic rice disease detection from rice paddy images serviced via a Chatbot, Comput. Electron. Agric. 185 (2021) 1–7, https://doi.org/10.1016/j.compag.2021.106156.
[18] J. Chen, W. Lin, H. Cheng, C. Hung, C. Lin, S. Chen, A smartphone-based application for scale pest detection using multiple-object detection methods, Electronics. (Basel) 10 (4) (2020) 1–24, https://doi.org/10.3390/electronics10040372.
[19] M. Carvajal-Yepes, K.F. Cardwell, A. Nelson, K.A. Garrett, B. Giovani, D.G. O Saunders, S. Kamoun, J. Legg, V. Verdier, J. Lessel, R.A. Neher, R. Day, P. G. Pardey, M.L. Gullino, A.R. Records, B. Bextine, J.E. Leach, S. Staiger, J. Tohmé, A global surveillance system for crop diseases, Science (1979) 364 (6447) (2019) 1237–1239, https://doi.org/10.1126/science.aaw1572.
[20] M. Griffith, G. Buss, P. Carroll, X. Yang, J. Griffis Jr, G. Papkov, S. Bauer, K. Jackson, A. Singh, The comparative performance of nutrient-film technique and deep-water culture method of hydroponics for GREENBOX technology, Agric. Sci. 14 (8) (2023) 1108–1120, https://doi.org/10.4236/as.2023.148074.
[21] F. Wang, Q. Gao, G. Ji, J. Wang, Y. Ding, S. Wang, Effects of light intensity and photoperiod on morphological development and photosynthetic characteristics of Coriander, Horticulturae 10 (3) (2024) 1–18, https://doi.org/10.3390/horticulturae10030215.
[22] H.M. Resh, Hydroponic Food Production: A Definitive Guidebook for the Advanced Home Gardener and the Commercial Hydroponic Grower (8th ed.), CRC Press, 2022, https://doi.org/10.1201/9781003133254.
[23] N. Santhosh, R. Shankar, R. Narendranath, K. Srinivasan, Research of production and growth of Coriander in various seasons using K-means algorithm, Int. J. Innov. Technol. Expl. Eng. (IJITEE) 8 (12S) (2019) 518–521, https://doi.org/10.35940/ijitee.l1129.10812s19.
[24] H. Noh, J. Lee, The effect of vapor pressure deficit regulation on the growth of tomato plants grown in different planting environments, Appl. Sci. 12 (7) (2021) 1–18, https://doi.org/10.3390/app12073667.

[25] J. Flusser, M. Lébl, F. Šroubek, M. Pedone, J. Kostková, Blur invariants for image recognition, Int. J. Comput. Vis. 131 (9) (2023) 2298–2315, https://doi.org/10.1007/s11263-023-01798-7.
[26] Kumar, T., Mileo, A., Brennan, R., Bendechache, M. (2023). Image data augmentation approaches: A comprehensive survey and future directions. arXiv preprint arXiv:2301.02830. https://doi.org/10.48550/arXiv.2301.02830.
[27] L. Jiao, C. Xie, P. Chen, J. Du, R. Li, J. Zhang, Adaptive feature fusion pyramid network for multi-classes agricultural pest detection, Comput. Electron. Agric. 195 (2022) 1–9, https://doi.org/10.1016/j.compag.2022.106827.
[28] Terven, J., Cordova-Esparza, D. (2023). A comprehensive review of YOLO architectures in computer Vision: from YOLOv1 to YOLOv8 and YOLO-NAS, arXiv preprint arXiv:2304.00501. https://doi.org/10.48550/arxiv.2304.00501.
[29] Wang, C., He, W., Nie, Y., Guo, J., Liu, C., Han, K., & Wang, Y. (2023). Gold-YOLO: efficient object detector via gather-and-distribute mechanism. arXiv preprint arXiv: 2309.11331. https://doi.org/10.48550/arXiv.2309.11331.
[30] M. Kang, C. Ting, F.F. Ting, R.C.W. Phan, ASF-YOLO: A novel YOLO model with attentional scale sequence fusion for cell instance segmentation, Image Vis. Comput. 147 (2024) 1–9, https://doi.org/10.1016/j.imavis.2024.105057.
[31] N. Chumuang, S. Hiranchan, M. Ketcham, W. Yimyam, P. Pramkeaw, S. Tangwannawit, Developed credit card fraud detection alert systems via notification of LINE application, in: 2020 15th International Joint Symposium on Artificial Intelligence and Natural Language Processing (iSAI-NLP), IEEE, Bangkok, 2020, pp. 18–20, https://doi.org/10.1109/isai-nlp51646.2020.9376829.
[32] Messaging API reference | LINE Developers, n.d. Available at, Accessed 13 April, https://developers.line.biz/en/reference/messaging-api/, 2024. Accessed 13 April.
[33] N. Ketkar, Introduction to PyTorch, Deep Learning with Python. Apress (2017) 195–208, https://doi.org/10.1007/978-1-4842-2766-4_12.
[34] A. Khan, Z. Rauf, A. Sohail, A.R. Khan, H. Asif, A. Asif, U. Farooq, A survey of the vision transformers and their CNN-transformer based variants, Artif. Intell. Rev. 56 (2023) 2917–2970, https://doi.org/10.1007/s10462-023-10595-0.
[35] Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2019). Distance-IoU Loss: Faster and Better Learning for Bounding Box Regression. arXiv preprint arXiv: 1911.08287. https://doi.org/10.48550/arXiv.1911.08287.
[36] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, A. Zisserman, The Pascal visual object classes (VOC) Challenge, Int. J. Comput. 88 (2010) 303–338, https://doi.org/10.1007/s11263-009-0275-4.
[37] L. Fischer, P. Wollstadt. (2023). Precision and Recall Reject Curves for Classification. arXiv preprint arXiv:2308.08381. https://doi.org/10.48550/arxiv.2308.08381.
[38] Y. Tian, S. Wang, E. Li, G. Yang, Z. Liang, M. Tan, MD-YOLO: multi-scale dense YOLO for small target pest detection, Comput. Electron. Agric. 213 (2023) 1–12, https://doi.org/10.1016/j.compag.2023.108233.
[39] C. Guo, S. Zheng, G. Cheng, Y. Zhang, J. Ding, An improved YOLO v4 used for grape detection in unstructured environment, Front. Plant Sci. 14 (2023) 1–19, https://doi.org/10.3389/fpls.2023.1209910.
[40] CL. Ji, T. Yu, P. Gao, F. Wang, R.Y. Yuan, Yolo-tla: an efficient and lightweight small object detection model based on YOLOv5, J. Real-Time Image Process. 21 (141) (2024) 1–16, https://doi.org/10.1007/s11554-024-01519-4.
[41] Feng, R., Guan, W., Qiao, Y., Dong, C. (2020). Exploring multi-scale feature propagation and communication for image super resolution. arXiv preprint arXiv: 2008.00239. https://doi.org/10.48550/arXiv.2008.00239.
[42] J. Qi, X. Liu, K. Liu, F. Xu, H. Guo, X. Tian, M. Li, Z. Bao, Y. Li, An improved YOLOv5 model based on visual attention mechanism: application to recognition of tomato virus disease, Comput. Electron. Agric. 194 (2022) 1–12, https://doi.org/10.1016/j.compag.2022.106780.