VISION-BASED TOMATO RIPENESS DETECTION USING DIGITAL IMAGE PROCESSING

/

डिजिटल इमेज प्रोसेसिंग का उपयोग करके टमाटर के पकने का दृष्टि-आधारित पता लगाना

Bibek ISHORE, Sanjay Kumar PATEL, Jaya SINHA, Subhash CHANDRA, and Sanjay KUMAR
Department of Farm Machinery and Power Engineering, College of Agricultural Engineering and Technology,
Dr. Rajendra Prasad Central Agricultural University, Pusa, Samastipur, Bihar, India
Corresponding Author: Bibek Ishore, Email: bibekishore5@gmail.com
DOI: https://doi.org/10.35633/inmateh-78-18

Keywords: *Image Preprocessing, Machine Learning, Ripeness Detection, Computer Vision*

ABSTRACT

Tomatoes (Solanum lycopersicum) are not only a staple in cuisines worldwide but also a subject of scientific interest due to their health benefits and distinct ripening process. Recognizing the ripest and most flavorful tomatoes has led to innovative research combining technology and agriculture. In this context, image processing emerges as a promising tool to discern the quality of tomatoes, particularly through color analysis. This study explores the effectiveness of a region-based image processing system in identifying red, ripe tomatoes. Currently, this process is done by hand, which takes time and can lead to mistakes-developed a machine learning-based device that utilizes computer vision and image processing techniques to detect ripe tomatoes with high accuracy. By employing algorithms that analyze color, texture, and shape, our technology can identify the optimal harvest time, making the process faster, more efficient, and more cost-effective. Automating tomato harvesting is crucial to addressing the labor crisis and enhancing the effectiveness of the present harvesting process. The actualization of automated harvesting depends on the ability to precisely recognize fruits. Fruit that is harvested at its peak maturity has the maximum levels of taste, vitamins, and sale value, which optimizes financial gains. There is now an inadequate rate of identification and failure to identify because of the blockage of specific fruits by vegetation and unwanted fruits, as well as the color change brought on by light. In order to identify tomato fruits in difficult circumstances, this research suggests a tomato identification system using the enhanced YOLOv8 framework. According to the model's test evaluation, the YOLOv8-Tomato model's mAP0.5 was 86.9%, its recall rate was 98%, and its accuracy and precision were 94% and 90%, respectively.

सारांश:

टमाटर (सोलनम लाइकोपर्सिकम) न केवल दुनिया भर के व्यंजनों में एक प्रधान है, बल्कि उनके स्वास्थ्य लाभों और विशिष्ट पकने की प्रक्रिया के कारण वैज्ञानिक रुचि का विषय भी है। सबसे पके और सबसे स्वादिष्ट टमाटरों को पहचानने से प्रौद्योगिकी और कृषि के संयोजन में नवीन अनुसंधान हुआ है। इस संदर्भ में, छवि प्रसंस्करण टमाटर की गुणवत्ता को समझने के लिए एक आशाजनक उपकरण के रूप में उभरता है, विशेष रूप से रंग विश्लेषण के माध्यम से। यह अध्ययन लाल, पके टमाटरों की पहचान करने में क्षेत्र-आधारित छवि प्रसंस्करण प्रणाली की प्रभावशीलता की पड़ताल करता है। वर्तमान में, यह प्रक्रिया हाथ से की जाती है, जिसमें समय लगता है और गलतियाँ हो सकती हैं-एक मशीन लर्निंग-आधारित उपकरण विकसित किया गया है जो उच्च सटीकता के साथ पके टमाटर का पता लगाने के लिए कंप्यूटर दृष्टि और छवि प्रसंस्करण तकनीकों का उपयोग करता है। रंग, बनावट और आकार का विश्लेषण करने वाले एल्गोरिदम को नियोजित करके, हमारी तकनीक इष्टतम फसल समय की पहचान कर सकती है, जिससे प्रक्रिया तेज़, अधिक कुशल और अधिक लागत प्रभावी हो जाती है। टमाटर की कटाई को स्वचालित करना श्रम संकट को दूर करने और वर्तमान कटाई प्रक्रिया की प्रभावशीलता को बढ़ाने के लिए महत्वपूर्ण है। स्वचालित कटाई की प्राप्ति फलों को ठीक से पहचानने की क्षमता पर निर्भर करती है। जो फल अपनी चरम परिपक्वता पर काटा जाता है, उसमें स्वाद, विटामिन और बिक्री मूल्य का अधिकतम स्तर होता है, जो वित्तीय लाभ को अनुकूलित करता है। अब वनस्पति और अवांछित फलों द्वारा विशिष्ट फलों की रुकावट के साथ-साथ प्रकाश द्वारा लाए गए रंग परिवर्तन के कारण पहचान और पहचान करने में विफलता की अपर्याप्त दर है। कठिन परिस्थितियों में टमाटर के फलों की पहचान करने के लिए, यह शोध उन्नत YOLOv8 *ढांचे का उपयोग करके टमाटर की पहचान प्रणाली का सुझाव देता है। मॉडल के परीक्षण मूल्यांकन के अनुसार,* YOLOv8-*टमाटर मॉडल का* mAP0.5 *86.9% था, इसकी रिकॉल दर 98% थी, और इसकी सटीकता और सटीकता क्रमशः 94% और 90% थी।*

INTRODUCTION

Automation and computer vision technologies have shown a great deal of promise in agriculture in recent decades for savings in costs and increased efficiency. By accurately determining the ripeness of fruits and vegetables, these technologies maximize the harvesting period and guarantee higher-quality produce *(Moya et al., 2025)*. Furthermore, early detection and diagnosis of plant diseases by computer vision, which helps retrieve data from pictures *(Mishra et al., 2022)*, enables prompt interventions that reduce crop losses while lowering labor costs and boosting productivity in operations. These developments support more economic and environmentally friendly farming practices by improving harvesting and disease control effectiveness. Although tomatoes are admired for their profitability on a worldwide level, their restricted hereditary variation, that is, the range of inherited characteristics within the tomato species, presents serious difficulties. Despite their outward distinctions, the majority of mass-produced tomato varieties are derived from a limited hereditary resource, making them more vulnerable to illnesses, pests, and environmental stress *(Delices et al., 2019)*.

Tomatoes (*Solanum lycopersicum*) are a significant commercial crop that is grown all over the world because of their distinct flavor, plenty of nutrients, and high tolerance for the environment (*Bac et al*., *2014*). However, the majority of tomato harvesting still involves hand picking, and issues including poor harvesting productivity, significant human involvement, high labor intensity, and inadequate harvest performance cannot be assured *(Fu et al., 2020)*. Thus, the key to encouraging the expansion of the tomato planting industry is investigating more effective orchard harvesting, where precise fruit recognition via technology serves as a crucial assurance for timely, injury-free, and effective harvesting in addition to realizing the crucial premise of smart harvesting *(Goldenberg et al., 2018; Qing et al., 2023)*. An essential component of the robotic plucking procedure is accurately identifying when a tomato fruit is mature.

The fruit guarantees the best possible nutritional composition and selling price only when it is at the ideal age for harvesting. The primary indicator of tomato ripeness is the fruit's surface appearance; ripe tomatoes often have a consistent reddish or orange hue (environmental factors during shipping might cause orange tomatoes to mature). However, autonomous harvesting robots have numerous difficulties in determining ripeness in real-world scenarios. One of these is that the accuracy of the color values in the photograph is impacted by the fact that tomato color also varies greatly due to various light circumstances brought on by the climate. The fruit surface color may show in the image with varying brightness and hues due to varying light concentrations and angles, making recognition more difficult. Furthermore, the desired fruit may get partially obscured by nearby leaves, branches, and other unwanted items, making detection much more challenging *(Zheng et al., 2024)*. Modern computational imaging and machine vision methods are required for tackling these obstacles and increasing the precision and identification rate of mature tomatoes.

Currently, the division of images, image attribute removal, color division, and other visual machine learning-based traditional object recognition methods have drawbacks such as inaccurate results, inadequate reliability, and a laborious process in unorganized expanding surroundings where the object being detected is obstructed by noise, illumination, and obstructions *(Sun et al., 2019; Lu et al., 2018; Hayashi et al., 2014)*. It is difficult to satisfy real-time requirements. To construct a robust classifier for identifying red, ripe tomato fruits through learning, a color-independent tomato fruit identification method based on AdaBoost was adopted. This approach trains multiple weak classifiers obtained through threshold-based judgment of Haar-like features, which are then combined to form a strong classifier *(Zhao et al., 2016)*.

When there are severe fruit occlusions and insufficient illumination, this method's detection rate for ripe tomato fruits is low. Agriculture has also made substantial use of deep learning approaches for recognizing objects in recent years, which have successfully addressed the issues with conventional target recognition techniques. Most of these techniques rely on Convolutional Neural Networks *(Fukushima, 1980; Lecun et al., 1998)*. Processing-wise, deep learning is frequently separated into two groups: one-stage detection and two-stage detection.

Two-phase recognition is best exemplified by the R-CNN algorithm series *(Girshick et al., 2014; Girshick, 2015; Ren et al., 2016; He et al., 2017)*. A model for apple recognition during the fruit reduction phase that can be applied to recognizing small objects in near-scene color was created based on YOLOv7 *(Long et al., 2023)*. When compared to the initial model, the mAP0.5 value, precision, and recall increased by 2.3%, 0.9%, and 1.3%, respectively. By using the Depthwise Separable Convolution (DSConv) technique, the conventional convolution procedure is replaced based on the YOLOv8s model *(Yang et al. 2023)*. This was accomplished by designing a Dual Path Attention Gate (DPAG) module and introducing the Feature Enhancement Module (FEM), which successfully increased the model's detection success rate to 93.4% in challenging

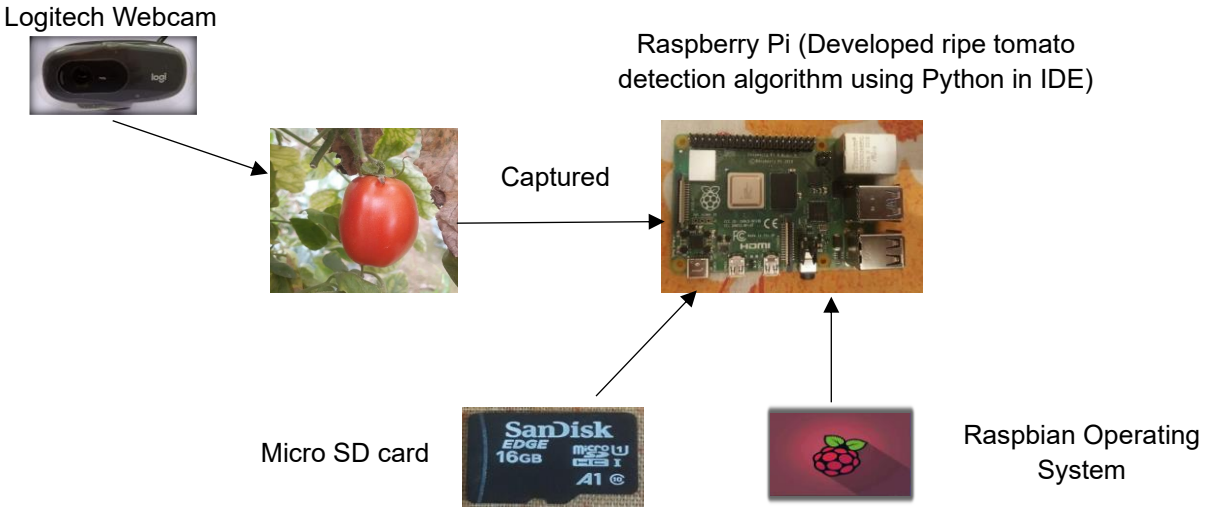
circumstances. The present research improves the YOLOv8n model in several ways while concentrating on the recognition and maturity recognition requirements and difficulties of cherry tomatoes in environmental conditions.

To satisfy the demands of the detection task while preserving detection accuracy, the model's parameter count and processing resource requirements are reduced. The efficiency of cherry tomato recognition and the identification effectiveness of fully developed cherry tomatoes in the environment of nature were assessed through tests and experiments, which offered a useful guide for the sensible distribution of work and precise targeting in the automated fruit harvesting process *(Liang et al., 2025)*. The hypothesis of the study is that integrating advanced image processing with an enhanced YOLOv8 framework will significantly improve the accuracy, precision, and recall of tomato ripeness detection under challenging field conditions, thereby enabling efficient harvesting automation and maximizing both quality and economic returns.

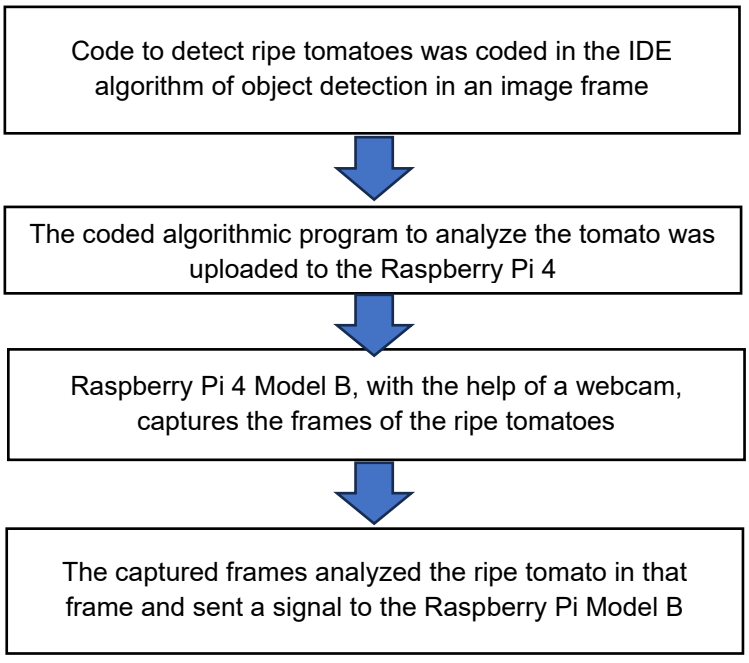
MATERIALS AND METHODS

Ripeness tomato detection

The ripeness tomato detection required a Raspberry Pi 4 Model B board, a Webcam (resolution: 480×640), and other necessary hardware components to capture, process, and analyze the image frames of ripeness tomatoes, as shown in Fig. 1(a). The process flow chart for detecting ripe tomatoes is shown in Fig. 1(b).



(a) System components



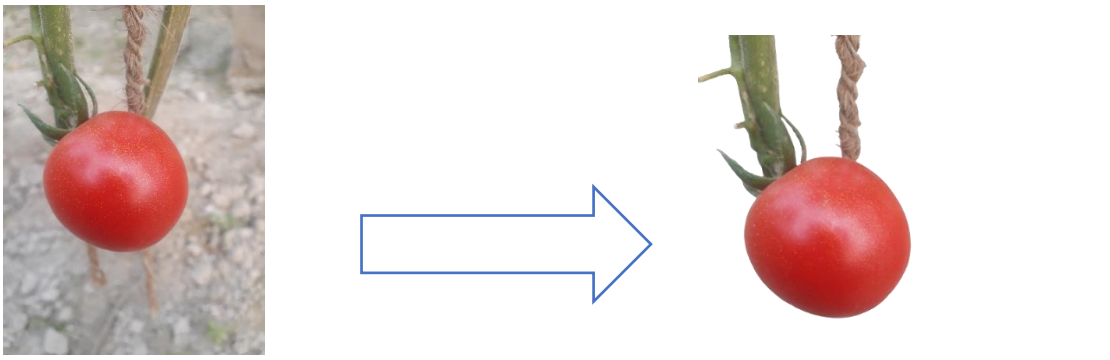
(b) Operational flowchart

Fig. 1 - System components and operational workflow of the vision-based ripe tomato detection system

Dataset preparation

A database was created from images captured on the campus of Dr. Rajendra Prasad Central Agricultural University. During preprocessing, the background and image size were reduced to remove background noise, enabling the network to focus on learning relevant features.

In addition, the large image size and substantial memory requirements of images acquired by the smartphone made model training difficult. The dataset preparation procedure is illustrated in the following figures. First, each image was scaled to a fixed resolution. After scaling, the images were further processed to remove the background, leaving only the pixels of interest, as shown in Fig. 2(a) and Fig. 2(b). The background removal was performed using an open-source web application.



(a) - Actual tomato image **(b) - Processed image after background removal**

Fig. 2 - Visual comparison of the tomato image across image processing stages

The processed image was labelled using the LABELING tool for model training. The tool's primary purpose is to provide a unique name and generate a bounding box around the target area in the image. The LABELING window is shown in Fig. 3.

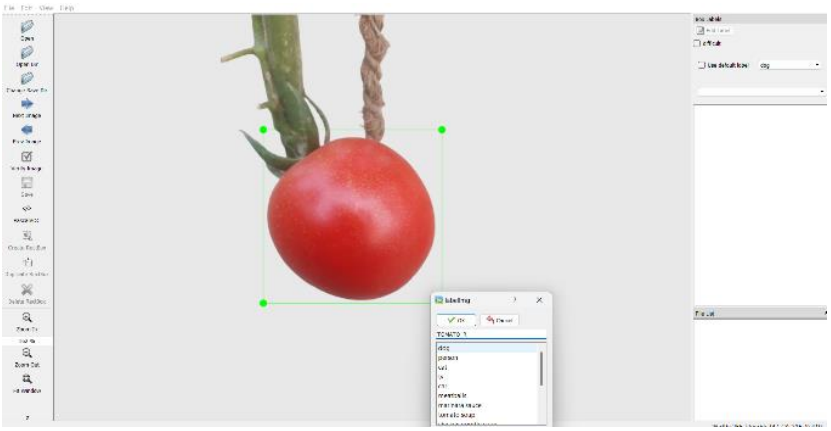


Fig. 3 - LABELIMG window

Each processed image was manually labelled, and then the file was saved. Using several photos of tomatoes in various stages of maturation, a specially customized database for various plant targets was created, and from there, the Python programming language, deep learning, and machine learning were used to produce an image processing system for important plant targets. The finished file contained two objects: the image and its associated tagged file in XML format, as shown in Fig. 4. The class of the object, label name, bounding box size, and file location were stated in the XML file.

Fig. 4 - View of the image containing and associated labels

The labelled dataset was further fragmented into two sections, namely the train and test sections. The number of images in each section is shown in Table 1. The block diagram for tomato detection is shown in Fig. 5.

Table 1

Dataset for tomato detection

Class	Total Number of Images
Train	1864
Test	466

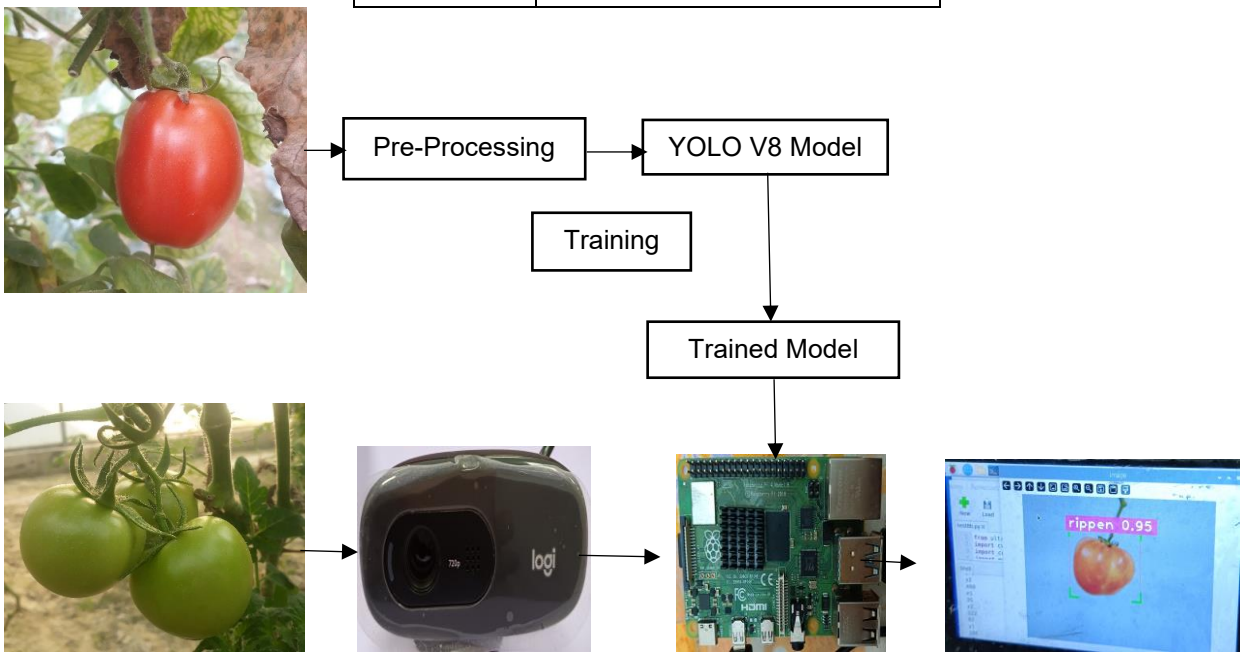


Fig. 5 - Block diagram for real-time tomato detection

Build a convolutional neural network for ripeness detection in tomato

Creating a virtual environment was necessary because it isolated all of the many Python libraries and dependencies that the object identification model required. Hence, a separate virtual environment was built for this purpose, and all the object detection components were installed inside of it. This also made sure that all the major dependencies and libraries that had previously been installed within Python were not in conflict. Using the import command, incorporating library bundles of code added additional instructions to enhance Python's functionality in its applications.

Creating folders and subfolders

For simple comprehension and access construction after the virtual environment is created and activated. The basic folder remained subfolders like the virtual environment folder and the training folder with Python code. Another set of four subfolders was made in the training folder. The object detection models of Ultralytics YOLOv8 were installed and located in the model's folder. YOLO scripts are located in the scripts folder. These are required to produce YOLOv8 records for model training and the effective installation of the Object identification API. There were four subfolders in the workspace folder. The main folder where the training results were kept was called the workspace folder. The. pb file, test, and train record file is all located in the annotation folder. The class label name and unique ID are contained in the pb file. The neural network is trained using the train and test record files. Test and Train folders are located inside the images folder. Images with appropriate labels are stored in these folders. The trained model is loaded for real-time detection using checkpoints that are stored in the trained model's folder.

Installation of object detection API and download pre-trained model

The YOLOv8 object detection API was installed on the local machine after the folder structure was configured. The pre-trained model was downloaded into the workspace directory.

Training of the model

Immediately following the successful installation of the object detection API, a label map (.pbtxt) file was generated and saved in the annotations folder. The .pbtxt file contains the class labels and unique IDs for each class. Subsequently, YOLO records were created separately for the training and testing images and saved as train record and test record files in the annotations folder. At the next stage, the pipeline configuration was adjusted. The paths to the train record, test record, and label map (.pbtxt) files were specified in the pipeline settings. The configuration also included the number of classes and image augmentation parameters to resize the images to match the input shape of the pre-trained model. The pipeline configuration file was stored in the model folder within the workspace subfolder. The model subfolder also contained additional folders, including train, test, eval, and export. After completing these steps, the pre-trained model was trained iteratively, and the training results were stored in the same folder as the model checkpoints. Following successful training, the model was evaluated to determine training and evaluation losses. Finally, the trained model was converted into a weight file and saved as *best_version-name.pt* in the model folder.

```
Starting training for 130 epochs...

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      1/130      3.36G      1.368      3.009      1.616        115        736: 100%|████████| 9/9 [00:06<00:00,  1.38it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|████████| 2/2 [00:01<00:00,  1.11it/s]
                   all         36        193

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      2/130      3.28G      1.319        2.7      1.433         93        736: 100%|████████| 9/9 [00:02<00:00,  4.21it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|████████| 2/2 [00:00<00:00,  5.25it/s]
                   all         36        193

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      3/130      3.23G      1.298      2.372      1.425        105        736: 100%|████████| 9/9 [00:01<00:00,  4.50it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|████████| 2/2 [00:00<00:00,  5.55it/s]
                   all         36        193

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      4/130      3.24G      1.319      2.015      1.451        110        736: 100%|████████| 9/9 [00:01<00:00,  5.14it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|████████| 2/2 [00:00<00:00,  5.50it/s]
                   all         36        193

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      5/130      3.27G      1.372      1.653      1.456        152        736: 100%|████████| 9/9 [00:01<00:00,  4.88it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|████████| 2/2 [00:00<00:00,  5.29it/s]
                   all         36        193

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      6/130      3.28G      1.363      1.474       1.48        117        736: 100%|████████| 9/9 [00:01<00:00,  4.76it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|████████| 2/2 [00:00<00:00,  5.62it/s]
                   all         36        193
```

Fig. 6 - Execution of 130 epochs

Transfer to Raspberry Pi

The weight file from the final training step was transferred to a folder in the Raspberry Pi operating system. A *labels.txt* file and a Python script containing the object detection code were also placed in the same folder. The *labels.txt* file lists the class names corresponding to the target objects. The program code for tomato ripeness detection was developed and executed using the Geany IDE. Essential libraries and packages were employed, including OpenCV for image processing tasks, *math* and *NumPy* for mathematical operations and array manipulation, TensorFlow for implementing machine learning models, and YOLO for object detection. These tools were integrated to create a system capable of accurately identifying ripe tomatoes based on the trained model.

Experimental setup for vision-based ripe tomato detection

The vision-based ripe tomato detection system was integrated for ripe tomato recognition. The circuit diagram of the ripe tomato detection system is shown in Fig. 7. Testing was conducted on the combined test set (466 images) to evaluate robustness. The trained model was deployed on the developed hardware, with inference optimized to ensure real-time ripe tomato detection performance. The detection unit consists of a webcam and its associated electronic circuitry for ripe tomato detection. The developed closed-loop circuit receives a feedback signal from the ripe tomato detection unit.

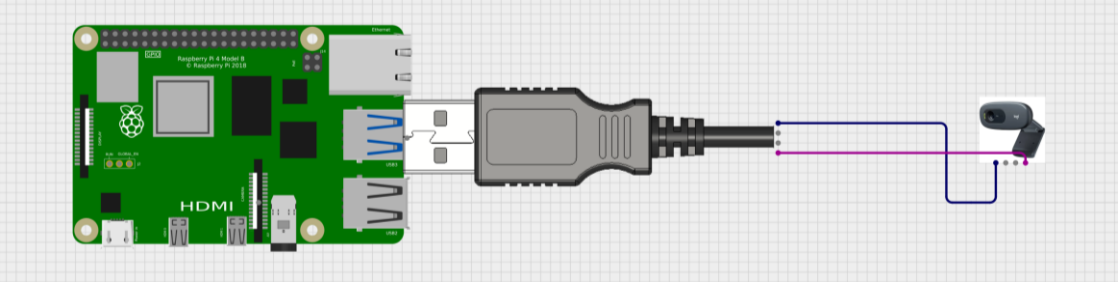


Fig. 7 - Electronic circuit diagram of the vision-based ripe tomato detection

Evaluation metrics computation

Evaluation is a critical stage, as it determines the effectiveness and reliability of the trained model in real-world scenarios. After model training was completed, the system was assessed using a separate test dataset that was not involved in either training or validation, ensuring objective performance evaluation. The evaluation of ripe tomato detection involved a comprehensive assessment of model performance. This process included loading the saved model, plotting validation metrics, and computing multiple evaluation indicators. For evaluation, the trained model was loaded from the specified file path using Python commands, and the same validation dataset used during training was employed to ensure consistency. Key performance metrics included precision, recall, and F1-score. Precision represents the proportion of correctly identified ripe tomatoes among all detected tomatoes, while recall measures the proportion of actual ripe tomatoes that were successfully detected. The F1-score provides a balanced measure of precision and recall. In addition to quantitative metrics, system robustness was evaluated through real-time testing. These tests involved capturing live video or image feeds and assessing the system's ability to detect ripe tomatoes. System response time, detection success rate, and false positive and false negative rates were also recorded.

Accuracy

Model accuracy, also referred to as test accuracy, is defined as the accuracy achieved on a separate test dataset that was not used during training. Accuracy represents the percentage of correctly classified instances, including both true positives and true negatives, relative to the total number of samples. In the context of tomato ripeness detection, accuracy reflects the proportion of tomatoes correctly classified as ripe or unripe.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (Sun\ X.\ 2024) \tag{1}$$

where: *True Positive (TP):* The number of predictions that are correctly forecasted as a specific class.

True Negative (TN): The number of predictions that are correctly forecasted as not belonging to a specific class.

False Positive (FP): The number of predictions that are incorrectly forecasted as a specific class.

False Negative (FN): The number of predictions that are incorrectly forecasted as not belonging to a specific class.

Precision

Precision, also known as Positive Predictive Value, measures the proportion of true positive predictions among all positive predictions made by the model. It indicates how many tomatoes predicted as a specific class (ripe or unripe) actually belong to that class, reflecting the accuracy of the model's positive predictions.

$$Precision = \frac{TP}{TP+FP} \quad (Yang\ Z.\ et\ al.,\ 2024) \tag{2}$$

Recall

Recall measures the proportion of true positive instances correctly identified by the model. It indicates how many actual instances of a specific class (ripe or unripe) are successfully detected, reflecting the model's ability to identify all positive instances of that class.

$$Recall = \frac{TP}{TP+FN} \text{ (Zheng S. et al., 2024)} \quad (3)$$

F_1 Score

The *F1 Score* is calculated as the harmonic mean of precision and recall. It provides a single metric that balances false positives and false negatives, making it particularly useful when the dataset is imbalanced, such as when one class significantly outnumbers the other. In such cases, accuracy may be misleading, whereas the *F1 Score* offers a more impartial performance assessment.

$$F_1 \, Score = \frac{2 \times Precision \times Recall}{Precision + Recall} \text{ (Sun X. 2024)} \quad (4)$$

Training and validation losses

Box Loss

Box loss measures the difference between the predicted and actual bounding box positions of the detected object. Box loss was calculated using Eq. (5) *(Maurya et al., 2025)*.

$$\text{Box Loss} = \sum_{i=1}^{n}(L1 \text{ or } IoU \text{ loss between predicted and actual box positions}) \quad (5)$$

where:

L1 loss (Mean Absolute error) measures the difference between the predicted bounding box coordinates and the actual (ground truth) bounding box coordinates.

IoU Loss (Intersection over Union) measures the overlap between the predicted bounding box and the ground truth bounding box.

Classification loss (Cls Loss)

This loss measures the error in predicting the correct class (ripe vs. unripe). Classification Loss was calculated using Eq. (6) (*Maurya et al., 2025*).

$$\text{Classification Loss} = \frac{1}{n} \sum_{i=1}^{n} yi \log(\widehat{y1}) \quad (6)$$

where:

yi is the ground truth class label (ripe or unripe)

ŷi is the predicted probability of the class.

Mean average precision (mAP)

Mean average precision (mAP) combines precision and recall across different IoU thresholds to measure overall detection performance and was calculated using Eq. (7) (Maurya *et al.,* 2025).

$$\text{mAP50} = \frac{1}{n} \sum_{i=1}^{n} APi \quad (7)$$

where:

n is the number of detection classes;

AP is the Average Precision, calculated as the area under the precision-recall curve.

The metric *mAP50-95* accounts for IoU thresholds ranging from 50% to 95%, providing a more comprehensive assessment of model performance across varying levels of detection overlap.

RESULTS

Image preprocessing

During the image preprocessing stage, the collected dataset was resized from its original resolution of 1920×1920 pixels to a standardized dimension of 224×224 pixels, as shown in Fig. 8.

Class	Original images (1920 * 1920)	Resized Images (224 * 224)	Background-Removed Images
Ripe			
Unripe			

Fig. 8 - Image Preprocessing

The transformation was necessary to align with the input requirements of the YOLOv8 Convolutional Neural Network (CNN) model used for tomato ripeness. The resizing operation was carried out using OpenCV, which provided an effective and reliable method for converting images to the required dimensions. Standardizing the image size ensured consistency across the dataset, which was vital for maintaining stable model training and achieving accurate classification results. Background removal was carried out to isolate the tomatoes from distracting or inconsistent backgrounds, aiming to improve the model's ability to focus on essential features. The process was implemented using OpenCV. The preprocessing contributed to clearer object boundaries, which helped enhance the accuracy of the YOLOv8 model during training.

RGB distributions

The ripe color frequency of the image is presented in three histograms in Fig. 9. The red channel (left) on the x-axis represents the pixel values, ranging from 0 to 255. The y-axis represents the frequency of occurrence of each pixel value. The graph shows a high concentration of pixel values in the higher range (around 200-255), indicating a strong presence of red color in the image. This is expected for ripe tomatoes, which are predominantly red.

Similar to the red channel, the green channel (middle) of the x-axis represents pixel values, and the y-axis represents frequency. The graph shows a broader distribution of pixel values compared to the red channel, with a peak around 100-150. This indicates a moderate presence of green color in the image, likely due to the green areas of the tomato or reflections.

The blue channel (right) of the x-axis and y-axis represent pixel values and their frequencies, respectively. The graph shows a relatively low frequency of pixel values across the entire range, with a slight peak around 50-100. This indicates a low presence of blue color in the ripe tomato image, as expected.

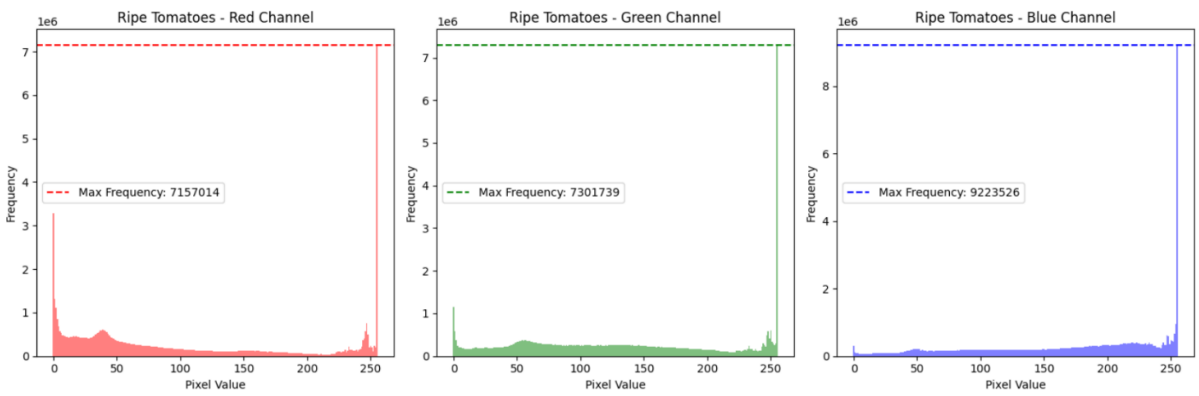


Fig. 9 - Color frequency histograms for ripe tomatoes

The unripe color frequency of the image presents three histograms in Fig. 10.

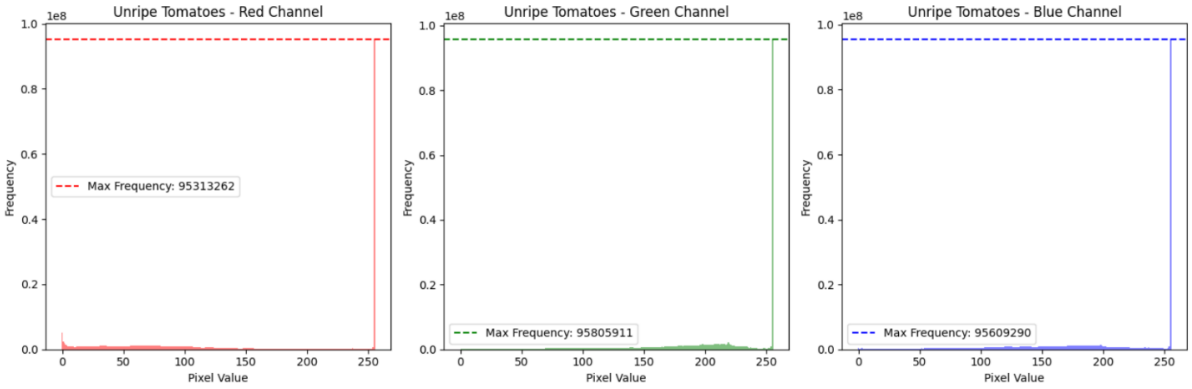


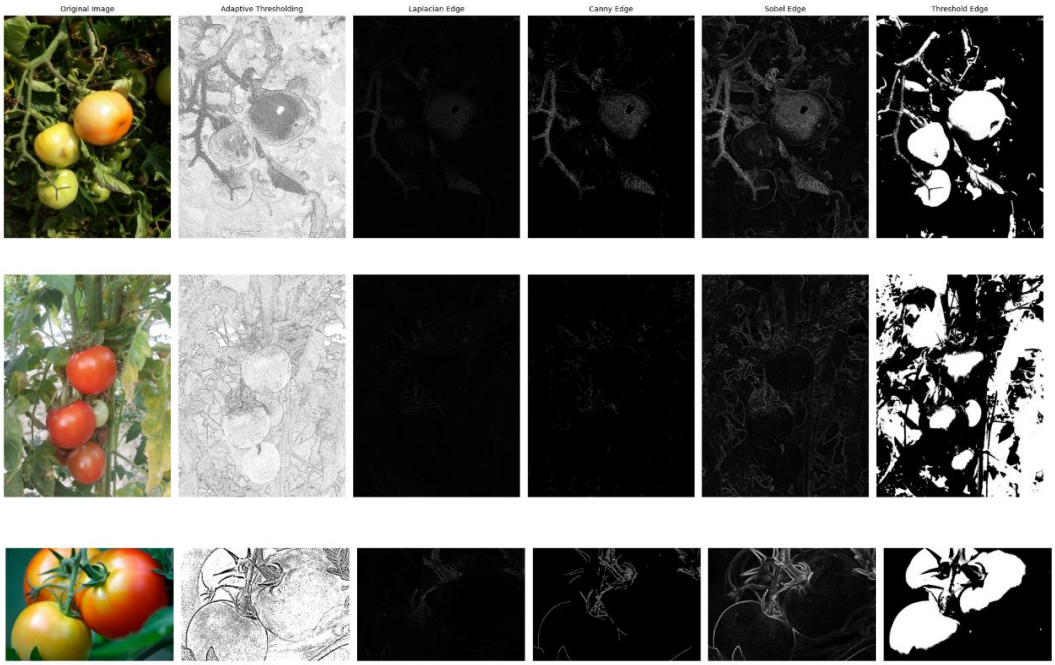
Fig. 10 - Color frequency histograms for unripe tomatoes

The red channel (Left) of the x-axis represents the pixel value (intensity) ranging from 0 to 255. The y-axis represents the frequency (number of pixels) with that particular intensity value. The red line shows a high frequency of pixels concentrated around the lower intensity values, indicating that unripe tomatoes have a lower red component.

Similar to the red channel, the green channel (Middle) of the x-axis represents pixel intensity, and the y-axis represents frequency. The green line shows a higher frequency of pixels around the middle range of intensity values, suggesting that unripe tomatoes have a more prominent green component compared to red. The blue Channel (Right) of the x-axis and y-axis represent pixel intensity and frequency, respectively. The blue line shows a similar pattern to the red channel, with a high frequency of pixels at lower intensity values, indicating a lower blue component in unripe tomatoes.

The edge detection of the picture shows a comparison of different image processing techniques applied to five images of tomatoes in Fig. 11. Adaptive thresholding displays the result of applying adaptive thresholding, which converts the image to black and white based on local pixel intensities, highlighting edges and details.

The Laplacian edge detection algorithm is used in the column, emphasizing edges and sharp transitions in the image. The canny edge of this column shows the output of the Canny edge detector, a more sophisticated edge detection algorithm that produces thinner and more accurate edges. The Sobel edge detection algorithm is applied in this column, highlighting edges and intensity gradients in the image. The thresholded edge column presents the final output after applying a thresholding operation to the edge-detected images, resulting in binary images that retain only edges and relevant features.



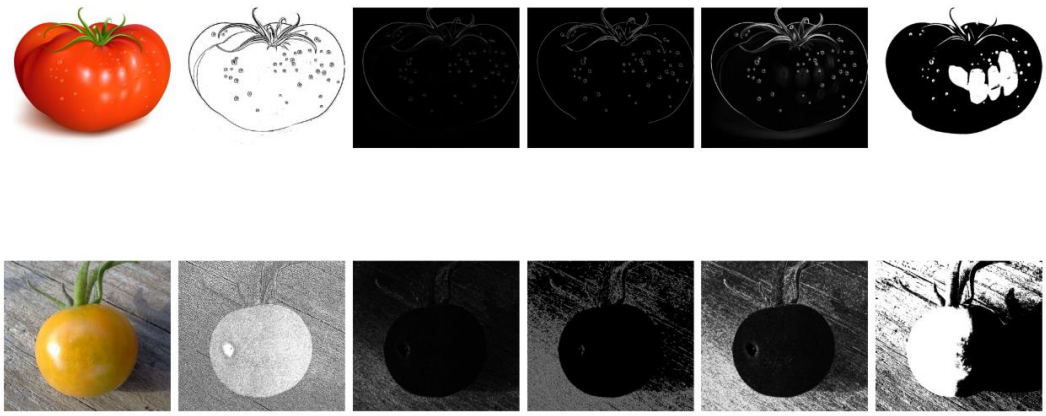


Fig. 11 - Edge detection

Figure 12 presents a histogram comparing the edge density of four edge detection methods: adaptive threshold, Laplacian, Canny, and Sobel. The adaptive threshold and Laplacian methods produced similar results, with most images exhibiting low edge densities in the range of 0-25. The Canny method shows a more evenly distributed range of edge densities, with a slight peak around 10-15. The Sobel method generated the highest edge densities, with most images falling within the ranges of 180-200 and 210-220.

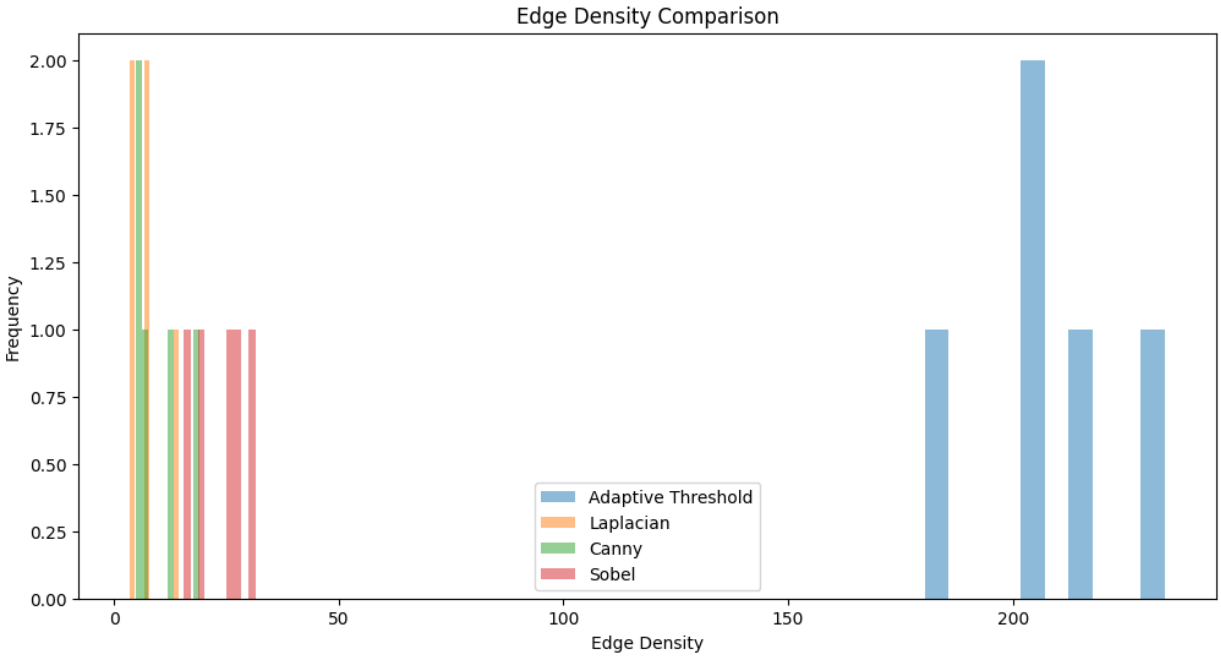


Fig. 12 - Edge density comparison

Model training

The model training process was conducted over 130 epochs, allowing the YOLOv8 convolutional neural network to iteratively learn from the dataset. This extended training phase enabled the model to progressively improve its ability to distinguish different maturity stages of tomatoes. The entire training cycle was completed in 0.108 h, demonstrating the efficiency of the optimized training pipeline and hardware resources. The reported processing speeds indicate the time required by the YOLOv8 model for tomato ripeness detection to preprocess input data, perform inference, and generate outputs. An inference time of 3.0 ms indicates rapid ripe tomato recognition, while a preprocessing time of 0.2 ms reflects efficient data preparation. The learning rate was set to 0.01, and the batch size was 16. During each epoch, model weights were updated based on the computed error, resulting in a gradual improvement in classification accuracy. This efficient training configuration was essential for achieving high-performance, real-time detection in agricultural applications.

Table 2

Model performance

Training steps	Total loss	mAP
20	0.45	0.40
50	0.42	0.50
100	0.38	0.39
130	0.36	0.41

The above table provides insight into the model's performance during the training process. As the number of training steps increases, the total loss decreases, indicating that the model is learning and improving, as lower total loss values generally correspond to better performance.

Model evaluation

The training and validation performance of the tomato ripeness detection model over 130 epochs shows a notable improvement in detecting ripe tomatoes, as illustrated in Fig. 13. The loss metrics, including box loss, classification loss, and distribution-focused loss, consistently decrease over time, indicating effective learning across epochs. Box loss, which measures the error in estimating bounding boxes around tomatoes, is a key indicator of object localization accuracy. As the model improves its ability to predict precise bounding boxes, box loss decreases rapidly during the early epochs, as shown in the training curve (Fig. 13a) and validation curve (Fig. 13b). By enabling the model to suppress irrelevant background information and focus on salient features in the ripe tomato regions, attention mechanisms enhance convergence. The attention-enhanced YOLOv8 model generalizes effectively to previously unseen images, as evidenced by the consistently low box loss during the validation phase.

As shown in Fig. 13(c) and Fig. 13(d), classification loss reflects the model's ability to correctly identify whether a detected object corresponds to a ripe tomato. This capability is particularly important in agricultural visual environments, where tomatoes may be partially occluded, unevenly illuminated, or deformed. By improving feature extraction from complex visual patterns, attention mechanisms substantially reduce classification loss during training. The consistently decreasing validation loss indicates reliable predictions and reduced overfitting. Without attention mechanisms, the model may incorrectly classify surface defects or background textures as ripe tomatoes, leading to an increased number of false positives.

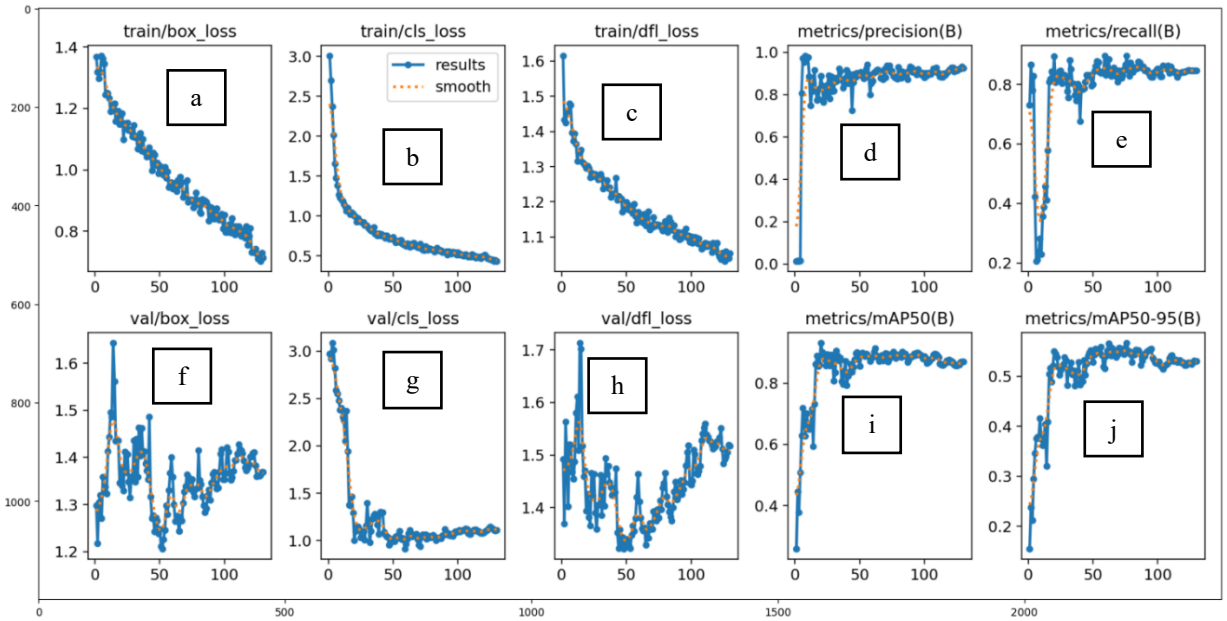


Fig. 13 - Graph of model evaluation performance

Dynamic focal loss is employed to enhance training by emphasizing difficult cases, including small, dimly illuminated, or occluded tomatoes, as shown in Fig. 13(e) and Fig. 13(f). When attention mechanisms are incorporated, the dynamic focal loss consistently decreases, indicating improved performance in detecting challenging instances. In the deeper layers of the network, attention modules help preserve fine details, such

as small protuberances on tomato surfaces, thereby reducing the impact of class imbalance and improving loss convergence. The precision and recall curves shown in Fig. 13(g) and Fig. 13(h) further demonstrate the effectiveness of the attention mechanism. A reduced false positive rate leads to a notable improvement in precision during the initial epochs, resulting from the network's enhanced ability to distinguish ripe tomatoes from visually similar background elements. As recall gradually increases, the network exhibits greater sensitivity in recognizing ripe tomatoes, including those that are partially occluded or located near image boundaries. By strengthening the receptive field and enhancing edge-related feature extraction, attention mechanisms improve recall while maintaining accuracy. The mean Average Precision (mAP) curves provide a comprehensive evaluation of detection performance. Under a 50% IoU threshold, Fig. 13(i) (mAP@0.5) shows a rapid increase followed by a plateau above 88%, indicating reliable localization and classification of ripe tomatoes. Figure 13(j) (mAP@0.5–0.95) reports consistently high values across stricter IoU thresholds, demonstrating strong localization accuracy at varying overlap levels. Overall, the results highlight the effectiveness of attention modules in improving both bounding box quality and classification accuracy. The integration of attention mechanisms significantly enhances the performance of the YOLOv8-based ripe tomato detection system, as evidenced by reduced loss values and increased precision, recall, and mAP across training and validation phases.

Confusion matrix

The confusion matrix presents metrics including true positives, true negatives, false positives, and false negatives. A higher concentration of values along the diagonal indicates better model performance, reflecting accurate classification. The results show that the model performs effectively in identifying unripe and ripe tomatoes, achieving accuracies of 94% and 86%, respectively. These results indicate that the features extracted through the image processing approach are effective in distinguishing different tomato ripening stages, as shown in Fig. 14.

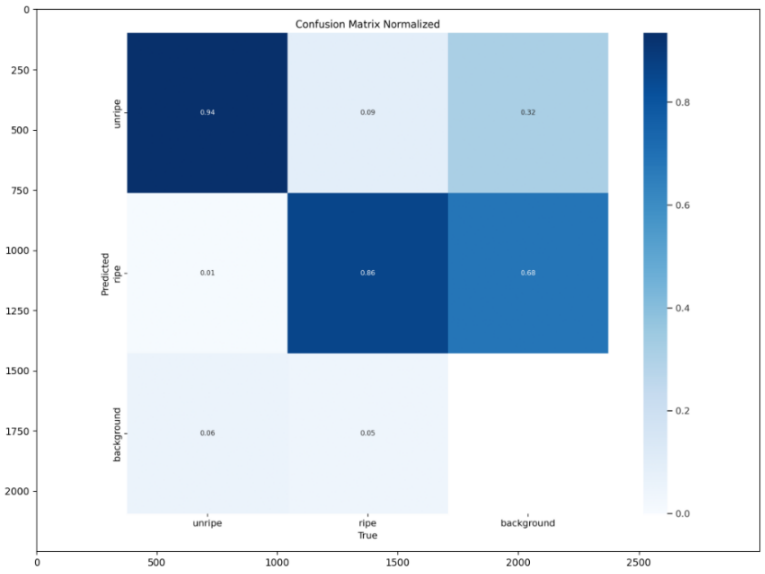


Fig. 14 - Confusion matrix

Evaluation matrix

In the context of detection tasks, the terms TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative) are used. They are computed by comparing the model's projected results with the actual results (ground truth). The metrics give a comprehensive view of the model's ability to distinguish between the "ripe" and "unripe" tomato classifications. True Negatives (TN) display accurate identifications of tomatoes that do not belong to any particular class, whereas True Positives (TP) illustrate instances where the model correctly classifies a tomato as belonging to a certain class. Predicted TP, TN, FP, and FN for each class are given below in Table 3.

Table 3

Evaluation matrix

Predicted (No)	**Predicted (Yes)**	
TN=0.94	FP=0.09	**Actual (No)**
FN=0.01	TP=0.86	**Actual (Yes)**

A bar chart in Fig. 15 was plotted using NumPy and Matplotlib to visualize the True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) for each class. This provided a clear comparison of the classification performance across different categories.

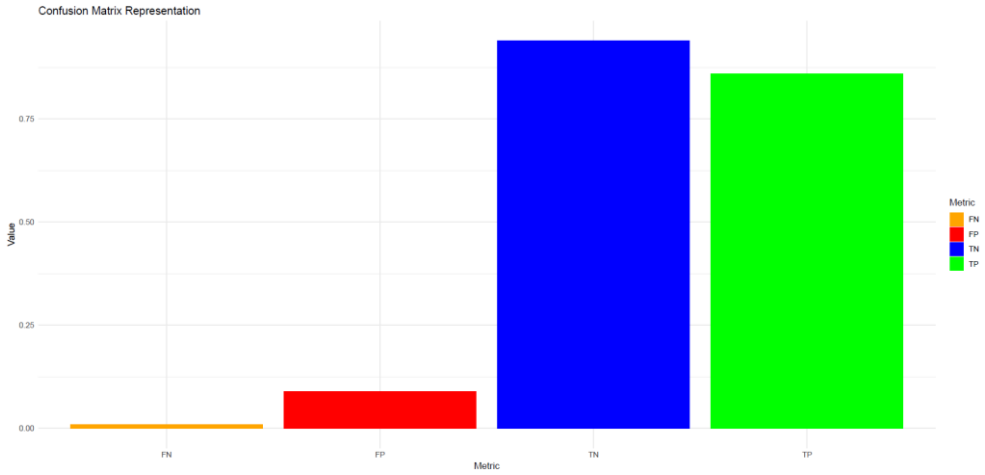


Fig. 15 - Confusion matrix

Accuracy

Accuracy is defined as the ratio of correct predictions to the total number of predictions and is a key indicator for evaluating model performance. Training accuracy reflects the model's ability to learn patterns from the training dataset and indicates how well the model fits the training data.

Validation accuracy provides insight into model performance on unseen data during the development phase and is essential for model tuning and for preventing overfitting.

The most objective assessment of a model's performance is provided by its test accuracy, which reflects how well the model generalizes to new, unseen data. Because it offers a realistic indication of expected real-world performance, overall model accuracy is sometimes used interchangeably with test accuracy. Test accuracy is critical for determining a model's reliability and practical usefulness and is regarded as the gold standard for evaluating a model's generalization capability.

$$\text{Accuracy} = \frac{0.86+0.94}{0.86+0.94+0.09+0.01} = 0.94 = 94\% \text{ (Sun, X. 2024)}$$

Precision and Recall

Precision evaluates the model's ability to make accurate predictions, reflecting the proportion of correct positive detections. Recall emphasizes the completeness of positive detections for a given ripening stage. The precision curve remains stable at a high level, indicating that the model consistently identifies ripe tomatoes with minimal false positives. Although the recall value exhibits slight fluctuations, it demonstrates effective detection of actual ripe tomatoes with few false negatives.

As shown in Fig. 16, the curve begins with a precision value of 1.0 (perfect precision) when recall is zero. As recall increases, precision decreases sharply. This steep decline indicates that as the model attempts to detect a larger number of ripe tomatoes (higher recall), precision decreases due to an increase in false positives. This behavior represents a typical trade-off observed in object detection models between precision and recall.

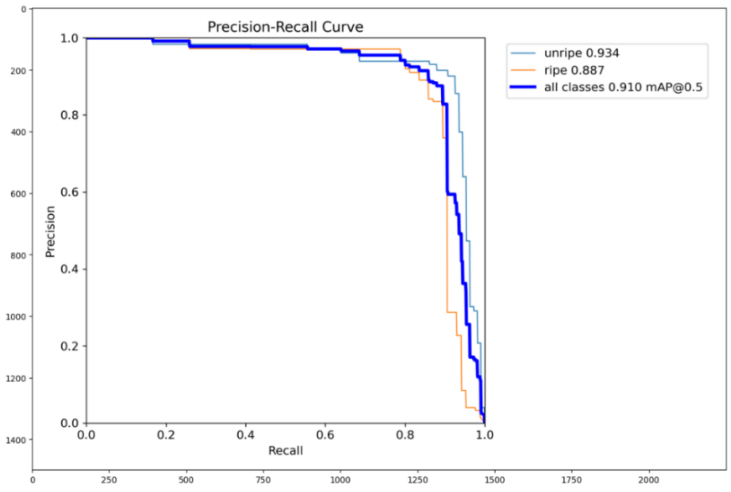


Fig. 16 - Graphical representation of Precision-Recall Graph

$$\text{Precision} = \frac{0.86}{0.86+0.09} = 0.90 = 90\% \text{ (Yang Z. et al., 2024)}$$

$$\text{Recall} = \frac{0.86}{0.86+0.01} = 0.98 = 98\% \text{ (Zheng S. et al., 2024)}$$

F1 score

The F1-score offers an impartial assessment of the model's performance. It considers both measures together by taking the harmonic mean of precision and recall. A high F1 score indicates that the model does an excellent task of detecting ripe tomatoes. Because of this, the F1-score is a useful indicator for confirming if the model is accurate and efficient. Section 3.2.6.4 in Chapter III contains the formula for calculating the F1 Score.

$$\text{F1 score} = \frac{2\times0.90\times0.98}{0.90+0.98} = 0.93 = 93\%$$

CONCLUSIONS AND FUTURE WORK

The developed image-processing-based network is capable of detecting mature red tomato fruits on plants. The proposed technique achieved an accuracy of approximately 94%, demonstrating the effectiveness of image processing approaches in agricultural applications. The method can be readily applied to color-based quality assessment of tomato crops, where accurate ripeness identification is essential for optimized harvesting and marketing. Precise fruit production assessment plays a critical role in improving harvesting efficiency and economic returns. The present study suggests that integrating characteristic pigment intensity with additional features can further enhance the determination of tomato ripeness stages. To support this objective, a digital imaging–based ripeness assessment system was developed for tomato image acquisition. Using machine learning techniques, ripe tomatoes were successfully identified from the processed images.

For future work, research will focus on integrating the YOLOv8-based tomato detection model with depth sensors and automated harvesting arms for deployment in commercial harvesting robots, enabling efficient mechanized harvesting operations. Further improvements will involve expanding the dataset, incorporating additional environmental and contextual parameters, and adapting the model to different tomato varieties to advance precision agriculture applications. In addition, more advanced deep learning architectures and data augmentation strategies will be explored to further enhance detection accuracy and predictive capability.

Declaration

All authors have read, understood, and complied as applicable with the statement on "Ethical responsibilities of Authors" as found in the Instructions for Authors.

Data Availability

The image dataset used in this research is accessed from https://github.com/bi641/Dataset-of-picture.

ACKNOWLEDGEMENT

All authors have read, understood, and complied as applicable with the statement on “Ethical responsibilities of Authors” as found in the instructions for authors.

REFERENCES

[1] Delices, G., Leyva Ovalle, O.R., Mota-Vargas, C., Nunez Pastrana, R., Gamez Pastrana, R., Meza, P.A., & Serna-Lagunes, R. (2019). Biogeografía del tomate *solanum lycopersicum* var. *cerasiforme* (solanaceae) en su centro de origen (sur de américa) y de domesticación (méxico), *Rev. Biol. Trop.*, 67 (4), 1023—1036, Costa Rica. http://dx.doi.org/10.15517/rbt.v67i4.33754

[2] Fu, L., Gao, F., & Wu, J. (2020). Application of consumer RGB-D cameras for fruit detection and localization in field: A critical review. *Comput. Electron. Agric.*, 177, 105687, 14-22, Netherlands. https://doi.org/10.1016/j.compag.2020.105687

[3] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybern.*, 36, 193–202, Japan. DOI:10.1007/BF00344251

[4] Girshick, R. (2015). Fast R-CNN. *In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, 1440-1448, USA. https://ieeexplore.ieee.org/document/7410526

[5] Girshick, R., Donahue, J., & Darrell, T. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, 1-8, USA. https://ieeexplore.ieee.org/document/6909475

[6] Goldenberg, L., Yaniv, Y., & Porat, R. (2018). Mandarin fruit quality: A review. *J. Sci. Food Agric*., 98 (1), 18–26, Israel. https://doi.org/10.1002/jsfa.8495

[7] Hayashi, S., Yamamoto, S., & Saito, S. (2014). Field operation of a movable strawberry-harvesting robot using a travel platform. *Japan Agricultural Research Quarterly*, 48 (3), 307–316, Japan. https://doi.org/10.6090/jarq.48.307

[8] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, 2961-2969, USA. https://doi.org/10.1109/ICCV.2017.322

[9] LeCun, Y., Bottou, L., & Bengio, Y. (1998). Gradient-based learning applied to document recognition. *Inst. Electr. Electron. Eng. Proc*., 86 (11), 2278–2324, USA. https://ieeexplore.ieee.org/document/726791

[10] Liang, X., Jia, H., Wang, H., Zhang, L., Li, D., Wei, Z., You, H., Wan, X., Li, R., Li, W., & Yang, M. (2025). ASE-YOLOv8n: A Method for Cherry Tomato Ripening Detection. *Agronomy*, 15 (5), 1088, 1-29, China. https://doi.org/10.3390/agronomy15051088

[11] Long, Y., Yang, Z., & He, M. (2023). Recognizing apple targets before thinning using improved YOLOv7. *Trans. Chin. Soc. Agric. Eng*., 39 (14), 191–199, China. DOI: 10.11975/j.issn.1002-6819.202305069

[12] Lu, J., Lee, W.S., & Gan, H. (2018). Immature citrus fruit detection based on local binary pattern feature and hierarchical contour analysis. *Biosyst. Eng*., 171, 78–90, USA. https://doi.org/10.1016/j.biosystemseng.2018.04.009

[13] Maurya, M., Kumar, S., Kumar, S., Kumari, S., Sahni, R.K., Patel, S.K., Chandra, S., & Kumar, N. (2025). Smart Vision-based Sugarcane Bud detection and Cutting System for seed generation. *Results in Engineering*, 27, 106993, 1-31, India. https://doi.org/10.1016/j.rineng.2025.106993

[14] Mishra, P., Kumar, S., Chaube, M.K., & Shrawankar, U. (2022). Evaginating scientific charts: recovering direct and derived information encodings from chart images, *Journal of visualization*. 39, 343—359, India. https://doi.org/10.1007/s12650-021-00800-z

[15] Moya, V., Guerra, M., Pazmino, K., Abedrabbo, F., Chicaiza, F.A., & Pozo-Espin, D. (2025). Tomato classification with YOLOv8: Enhancing automated sorting and quality assessment. *Smart Agricultural Technology*, 12, 101221,1-18, Ecuador. https://doi.org/10.1016/j.atech.2025.101221

[16] Qing, C., Chengkai, Y., Ziliang, G., Jinpeng, W., Hongping, Z., & Xuesong, J. (2023). Current status and future development of the key technologies for apple-picking robots. *Nung Yeh Kung Ch'eng Hsueh Pao*, 39 (4), 1–15, China. http://tcsae.org/en/article/doi/10.11975/j.issn.1002-6819.202209041

[17] Ren, S., He, K., & Girshick, R. (2016). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Anal*., 39 (6), 1137–1149, USA. DOI: 10.1109/TPAMI.2016.2577031

[18] Sun, S., Jiang, M., He, D., Long, Y., & Song, H. (2019). Recognition of green apples in an orchard environment by combining the GrabCut model and Ncut algorithm. *Biosyst. Eng*., 187, 201–213, China. https://doi.org/10.1016/j.biosystemseng.2019.09.006

[19] Sun, X. (2024). Enhanced tomato detection in greenhouse environments: a lightweight model based on S-YOLO with high accuracy. *Frontiers in Plant Science*, 1-17, China. https://doi.org/10.3389/fpls.2024.1451018

[20] Yang, G., Wang, J., Nie, Z., Yang, H., & Yu, S. (2023). A lightweight YOLOv8 tomato detection algorithm combining feature enhancement and attention. *Agronomy*, 13 (7), 1824, China. https://doi.org/10.3390/agronomy13071824

[21] Yang, Z., Li, Y., Han, Q., Wang, H., Li, C. & Wu, Z. (2024). A Method for Tomato Ripeness Recognition and Detection Based on an Improved YOLOv8 Model. *Horticulture*, 11 (15), 1-21, China. https://doi.org/10.3390/horticulturae11010015

[22] Zhao, Y., Gong, L., Zhou, B., Hua, Y., Niu, Q., & Liu, C. (2016). Object recognition algorithm of tomato harvesting robot using a non-colour coding approach. *Trans. Chin. Soc. Agric. Mach*., 47 (7), 1–7, China. DOI: 10.6041/j.issn.1000-1298.2016.07.001

[23] Zheng, S., Jia, X., He, M., Lin, T., & Weng, W. (2024). Tomato Recognition Method Based on the YOLOv8-Tomato Model in Complex Greenhouse Environments. *Agronomy*, 14 (8), 1-21, China. https://doi.org/10.3390/agronomy14081764