ISSN 2085-4552

Real-Time Tomato Quality Detection System Using You Only Look Once (YOLOv7) Algorithm

Isna Ayu Muarofah[1], Muhammad Jauhar Vikri[2], Ita Aristia Sa'ida[3]
[1,2,3] Informatics, Faculty of Sains and Technology, Universitas Nahdlatul Ulama Sunan Giri, Bojonegoro, Indonesia
[1] isnaayu2@gmail.com, [2] vikri@unugiri.ac.id, [3] itaaristia@unugiri.ac.id

Accepted 02 October 2023
Approved 29 November 2023

Abstract— **Real-time object detection is a crucial aspect of computer vision. With the increasing prominence of the big data field, it has become easier to gather data from various sources. Over the past few decades, computer vision inspection systems have become essential tools in agricultural operations, and their usage has seen a significant rise. Computer vision automation-based technology in agriculture is increasingly being employed to enhance productivity and efficiency. Tomato is a widely utilized crop commodity, finding applications in food, cosmetics, and pharmaceuticals. Consequently, tomato farming continues to evolve and has become one of the nation's export commodities. YOLO is an algorithm capable of real-time object detection and recognition. In this study, the YOLOv7-tiny architecture, which has lower computational overhead, was utilized. For quality detection of tomatoes, they were categorized into three classes: ripe, unripe, and defective. The trained model yielded a recall score of 0.97, precision of 1.0, a PR-curve of 0.838, and an F1-score of 0.81, indicating that the model learned effectively. The research achieved an accuracy of 90.6% on original images with an average IoU of 0.90 and a detection time of 2.7 seconds. In images with added light disturbance, the average accuracy was 91.2%. Images with reduced light yielded an average accuracy of 92%, while images with blur disturbance had an average accuracy of 78.2%. In real-time testing, ripe tomatoes were detected up to a maximum distance of 90cm, unripe tomatoes at 90cm, and defective tomatoes at 70cm.**

Index Terms— **Computer Vision; Detection, Localization; Recognition; Tomato; YOLO; YOLOv7**

I. Introduction

Real-time object detection is a crucial aspect of computer vision. Recently, there has been an increasing demand for the development of automated systems. Alongside the rising prominence of big data, it has become easier to acquire data from various sources. Many data processing techniques involving big data, such as machine learning, artificial intelligence, data mining, and others, are in play. The deluge of big data in the digital realm calls for immediate action from AI and the intelligent systems community to address how we manage knowledge. [1].

The demand for effective and safe agricultural food production methods is on the rise. Traditional agricultural management methods need to be complemented with innovative sensing technologies, driving technologies, and enhanced information and communication technology to accelerate agricultural productivity improvements more accurately. This will ultimately promote high-quality and high-yield agriculture.

Over the past few decades, computer vision inspection systems have become crucial tools in agricultural operations, and their usage has seen a significant rise. Expert systems and intelligent systems based on computer vision algorithms have become common components in agricultural production management. Automation-based computer vision technology in agriculture is increasingly being employed to enhance productivity and efficiency.

Tomatoes are a widely used horticultural commodity, finding applications in food, cosmetics, and pharmaceuticals. As a result, tomato farming continues to evolve and has become one of the nation's export commodities. According to the Central Statistics Agency (BPS), tomato production in Indonesia reached 1.11 million tons in 2021. This figure represents an increase of 2.72% compared to the previous year's production of 1.08 million tons. National tomato production has been on the rise since 2017 and reached its highest level in the past decade last year. In East Java alone, tomato production reached 1,021,085 quintals in 2022, a considerable quantity.

However, ensuring tomato quality for customer satisfaction is a challenge due to the large quantity. Additionally, it requires a significant amount of human resources. To facilitate efficient management, particularly for sorting, a model or system capable of detecting tomato quality is needed.

The research conducted by Paulina [2] regarding tomato quality suggests that the use of methods with continuously updated weights is recommended. This allows the model to keep learning and better recognize data. There are several methods for feature extraction, namely Histogram of Oriented Gradients (HOG), Haar feature, and Local Binary Pattern (LBP). All of these are standard feature extraction techniques. The study conducted by Kosasih & Daomara [3], which combines LBP and Histogram, demonstrated good accuracy. However, this method is sensitive to both lighting conditions and movement. Meanwhile, Firmansyah & Alfianto [4], conducted research by combining Haar and LBP. This study achieved reasonably good accuracy, but the objects failed to be detected when experiments involved obstacles.

Research was also conducted by Hafidhoh & Sukmana [5], who combined the SVM and HOG methods. This study achieved fairly good accuracy, but failed to detect due to factors related to HSV color space and consumed a considerable amount of time. There are also deep learning-based detections. One of the two-stage detection algorithms conducted by Hasma & Silfianti [6] had relatively longer detection times. Quoting from the creator's journal of the You Look Only Once version 1 algorithm [7], the R-CNN algorithm has a higher background error rate. However, R-CNN boasts higher accuracy rates, while YOLO offers faster detection speed. Up until now, YOLO has seen numerous developments. Although the original creator of the algorithm has ceased research due to ethical concerns, the research has been continued to this day. AlexeyAB, who is also a developer of YOLOv4, recently released YOLOv7. AlexeyAB claims that YOLOv7 outperforms its predecessors by being 120% faster [8]. Because of this, the researcher is interested in utilizing the YOLO algorithm with the YOLOv7 tiny architecture in this study.

II. METHODOLOGY

A. Research Objectives

In this research, the object used is tomatoes. The quality of the fruit, in this case, is determined by the level of freshness of these tomatoes. The level of freshness of a fruit can be determined by several factors, such as the physical condition of the fruit, which should be fresh with smooth skin without signs of wrinkles or wilting. The color of the skin should indicate that the fruit is sufficiently ripe, and there should be no signs of bruising or other damage, such as scratches, fungal growth, and so on [9].

The fruit has a round shape with a slight indentation at the base, and it has a thin, glossy skin. When young, the skin and flesh of the fruit are green and the taste is unpleasant. When ripe, the fruit's color turns to red, green, or orange [10]. There are various types of tomatoes such as cherry tomatoes, apple tomatoes, grape tomatoes, kumatoes, green tomatoes, and so on. Green tomatoes are harvested before they ripen. In this study, the Researcher categorizes the tomatoes into three parts: raw, ripe, and defective. Figure 1 serves as an example of the research object.

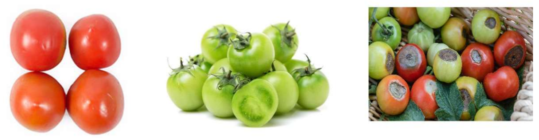


Fig. 1. Ripe tomato (left), unripe tomato (middle), and defective tomato (right).

B. Classification, Localization, Detection

Generic object detection, commonly referred to as generic object category detection, involves the classification and localization of object classes. There is currently a strong emphasis on detecting various natural categories, as opposed to predefined narrow interest categories. Thousands of objects inhabit the visual world we live in, and the research community is currently more interested in localizing highly structured objects (such as cars, bicycles) and articulated objects (such as humans, horses) rather than unstructured scenes (like sky, grass, and clouds). The spatial location and extent of an object can be coarsely determined using bounding boxes. The goal of object classification and object categorization is to assess the presence of objects from a certain set of object classes in an image, meaning assigning one or more object class labels to a specific image. Locating objects in an image makes detection a more challenging task than classification. The problem of object recognition presents a more general problem in identifying or localizing all objects present in an image [11]. Illustrations related to classification, localization, and detection can be found in Figure 2.2.

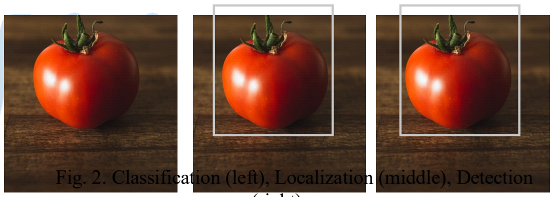


Fig. 2. Classification (left), Localization (middle), Detection (right)

C. You Only Look Once (YOLO)

You Only Look Once, commonly known as YOLO, is an object detection algorithm that employs a single neural network approach to predict bounding boxes and class probabilities through the entire image features in a single evaluation [7]. YOLO divides the image into an SxS grid. If there is an object inside it, then that grid cell is responsible for detecting the object within it. Each grid cell predicts B bounding boxes and confidence values. Each bounding box (B) contains 5 components: x, y, w, h, and class confidence. (x, y) are the coordinate points of the bounding box relative to the grid cell. (w, h) represent the width and height of the predicted box relative to the entire image. The confidence value is formulated as shown in Equation 1.

$$Pr(Object) * IOU\frac{truth}{pred} \quad (1)$$

The confidence value must be 0 if there is no object inside it. Each grid cell also predicts the class probabilities (C) within that cell.

$$\Pr(Class_i|Object) \quad (2)$$

The probabilities refer to the scenario when an object is present within the grid cell. YOLO predicts only one set of possible classes for each grid cell regardless of the number of boxes. Therefore, the class confidence score is utilized to gauge the confidence level in object classification and localization. This leads to the following equation:

$$\Pr(Class_i|Object) * Pr(Object) * IoU\frac{truth}{pred} = \Pr(Class_i|Object) * IoU\frac{truth}{pred} \quad (3)$$

D. YOLOv7-Tiny Architecture

In this study, the Researcher utilized the architecture of YOLOv7-tiny, which is optimized for edge devices. The YOLOv7-tiny architecture has undergone a reduction of 39% in parameters and 49% in computations, aimed at reducing both training and detection processes [8]. YOLOv7-tiny consists of three components: input, backbone, and head. The backbone layer is responsible for extracting feature maps, while the head is used for making predictions [12]. In YOLOv7-tiny, feature maps are generated at three different scales. Feature Pyramid Networks (FPN) are employed in this case to produce these different scales. This is highly beneficial for detecting objects of various sizes, be they small, medium, or large. The main idea behind FPN is to leverage the characteristics of convolutional layers that reduce the spatial dimension and increase the coverage of each feature on the original image to make predictions at different scales. In this study, the framework used is PyTorch, which has a default padding value of 1. The purpose of setting the padding value to none is to use auto-padding to ensure that the output of the convolution has the same dimensions as the input. The value of P is set to half of the kernel size. Concatenation, commonly referred to as concat, is the process of merging or combining previously extracted features to produce better information processing [13]. Each convolution layer plays a crucial role in processing information. However, on the other hand, larger layers and parameters also require greater computational resources. An illustration of the architecture used can be seen in Figure 3.

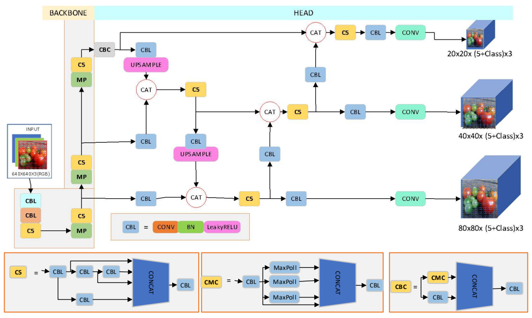


Fig 3. YOLOv7-tiny architecture

E. Loss Function

The loss function is used to estimate the level of inconsistency between the model's predicted values and the actual values. The primary task of training the model in this work is to use optimization methods to find model parameters that minimize the loss function. The loss function determines the optimal value of the model, so the performance of various object detectors is influenced by the loss function. The lower the loss value, the better the model learns. The loss function generally consists of bounding box regression and classification [14]. In YOLOv7, there are three components in determining the loss, which can be observed in Equation 4.

$$Loss = loss_{reg} + loss_{conf} + loss_{cls} \quad (4)$$

The regression loss of the object bounding box is the regression loss of the predicted bounding box calculated using the Complete Intersection Over Union (CIOU) [15]. As shown in Figure 4, CIOU takes into account three geometric measures: overlapping area, distance between center points, and aspect ratio, which make the regression of bounding boxes more stable [16].

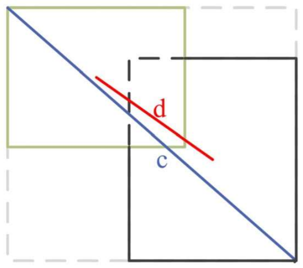


Fig. 4. Distance diagram frame

"c" represents the diagonal distance that encompasses both the predicted bounding box and the ground truth bounding box. "d" represents the Euclidean distance in the spatial domain between the center points of the predicted bounding box and the ground truth bounding box. The calculation of CIOU is formulated as follows:

$$CIOU = IOU - \frac{\rho 2(b, b^{gt})}{c^2} - \alpha v, \quad (5)$$

where,

$$IOU = \frac{area(b \cap b^{gt})}{area(b \cup b^{gt})}, \qquad (6)$$

$$\alpha = \frac{v}{(1 - IOU) + v}, \qquad (7)$$

$$v = \frac{4}{\pi 2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2, \qquad (8)$$

Where b and b^{gt} represent the predicted bounding box and the ground truth bounding box, respectively; w^{gt} and h^{gt} represent the width and height of the ground truth bounding box; $\rho 2(b, b^{gt})$ represents the Euclidean distance in the spatial domain between the center points of the predicted bounding box and the ground truth bounding box; c represents the diagonal distance of the area encompassing the predicted bounding box and the ground truth bounding box; α represents the weighting factor; v is used to consider the similarity or consistency of aspect ratios; IOU represents the Intersection over Union ratio between the predicted bounding box and the ground truth; $1 - CIOU$ represents the regression loss of the predicted bounding box.

$$loss_{CIOU} = 1 - CIOU = 1 - IOU + \frac{\rho 2(b, b^{gt})}{c^2} + \alpha v, \quad (9)$$

$$loss_{reg} = \lambda coord = \sum_{i=0}^{KxK} \sum_{j=0}^{M} I_{ij}^{obj} (2 - w_i \times h_i) loss_{CIOU}, \qquad (10)$$

Where $\lambda coord$ represents the weighting factor for positive samples. The input is segmented into K x K cells, and each cell produces M bounding box predictions. I_{ij}^{obj}= 1 when there is an object in the j predicted box in cell i. I_{ij}^{obj}= 0 when there is no target. w_i and h_i denote the width and height of the predicted bounding box, with $(2 - w_i \times h_i)$ representing the penalty term. Smaller bounding boxes have larger weights

Confidence loss consists of two parts: confidence loss for positive samples and confidence loss for negative samples. The loss value is calculated using binary cross-entropy.

$$loss_{conf} = -\sum_{i=0}^{KxK} \sum_{j=0}^{M} I_{ij}^{obj}[\widehat{C_i} \log(C_i) + (1 - \widehat{C_i}) \log(1 - C_i)], \quad (11)$$

$$-\lambda_{noobj} \sum_{i=0}^{KxK} \sum_{j=0}^{M} I_{ij}^{noobj}[\widehat{C_i} \log(C_i) + (1 - \widehat{C_i}) \log(1 - C_i)], \quad (12)$$

Where $\widehat{C_\iota}$ represents the predicted confidence value, C_i represents the ground truth confidence value and λ_{noobj} represents the weighting factor for negative samples. I_{ij}^{noobj} = 1 when there is no object in the j predicted box in cell i. I_{ij}^{noobj} = 0 hen there is an object.

Loss of predicted classes: This consists of the loss of positive sample classification, and its value is calculated using cross-entropy.

$$loss_{cls} = -\sum_{i=0}^{KxK} \sum_{j=0}^{M} I_{ij}^{obj} \sum_{c \in classes} [\widehat{p_\iota}(c) \log(\widehat{p_\iota})) + (1 - \widehat{p_\iota}(c)) \log(1 - p_i(c))], \quad (13)$$

Where $\widehat{p_\iota}(c)$ represents the predicted sample value of the class. $p_i(c)$ denotes the probability or likelihood of an object belonging to category c.

F. *Intersection Over Union (IoU)*

In general, Deep Learning algorithms combine classification and regression tasks in various ways. For example, in object detection, it requires predicting bounding boxes (regression) and a class within those bounding boxes (classification). IoU is a term used to describe the overlap level between two boxes. The larger the overlapping area, the greater the IoU. IoU is commonly used in object detection to train the model to produce accurate bounding boxes. IoU can be calculated using the following formula:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \qquad (14)$$

G. *Non Max Suppression (NMS)*

Non-Maximum Suppression (NMS) is a technique primarily used in object detection to select the best bounding box from a set of overlapping boxes. Because the goal of object detection bounding boxes is to have one box per object, NMS needs to be applied. For example, if a IoU threshold of 0.5 is set, it means that if the IoU is greater than the threshold, the bounding box with the lower confidence score will be removed.

H. *Data Augmentation*

Augmentation is a data preprocessing technique used to increase data diversity or generate new data from existing data. The augmentations used include mixup and mosaic. Additionally, augmentations in the form of HSV adjustments and flips are performed. HSV is a color space introduced by A.R. Smith in 1978, inspired by the intuitive nature of colors. In the field of data augmentation for tomato detection, it is used to enhance the color contrast of images by adjusting the hue, saturation, and value intensity ratios. The HSV color space can be visualized using a cone, as shown in Figure 5.

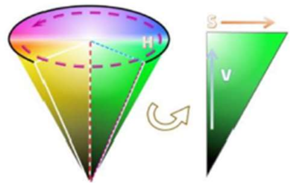


Fig. 5. HSV Augmentation for Data Augmentation

H (hue) on the cone represents the color phase angle, ranging from 0◦ to 360◦. S (saturation) stands for value ratio, which correlates with the purity of a specific color. Following the direction of the S arrow, the purity of the color witnesses a significant increase. V (value) represents the brightness of the color, ranging from 0 to

1. The V cone values range from 0 at the bottom for black to 1 at the top for white, with higher values indicating greater brightness [17]. In the case of 8-bit and 16-bit images, R, G, and B are converted to floating-point format and scaled to fit within the range of 0 to 1 using the following formula:

$$V \leftarrow max(R,G,B) \quad (15)$$

$$S \leftarrow \begin{cases} \frac{V \leftarrow min(R,G,B)}{V} & if\ V \neq 0 \\ 0 & otherwise \end{cases} \quad (16)$$

$$H \leftarrow \begin{cases} 60(G-B)/(V-min(R,G,B)) if\ V = R \\ 120+60(B-R)/(V-min(R,G,B)) if\ V = G \\ 240+60(R-G)/(V-min(R,G,B)) if\ V = B \end{cases} \quad (17)$$

Mixup is an unconventional data augmentation method based on the principle of independent data augmentation that uses linear interpolation to create new training samples and labels. The formula for processing the data labels is as follows:

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j \quad (18)$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j \quad (19)$$

Among them, two data pairs (x_i, y_i) and (x_j, y_j) are pairs of original training data (training sample and corresponding label); λ is a parameter that follows the division of β ; $\tilde{x}$ is the training sample from the mixup after the data enhancement operation; $\tilde{y}$ is the label o $\tilde{x}$. Figure 6 provides an illustrative example of mixup augmentation.

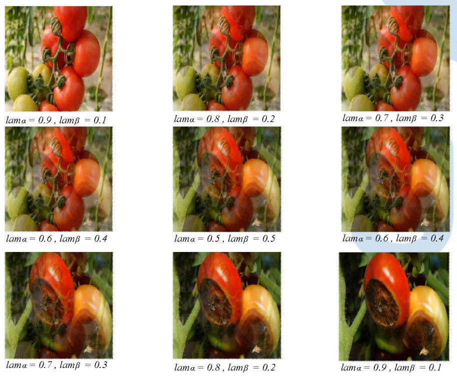


Fig. 6. Example illustration of mixup augmentation

The proportion of alpha and beta is lam\alpha\ + lam\beta\ =\ 1. The idea behind mosaic data augmentation is to randomly cut four images and combine them into a new training data, greatly enriching the detection data set, strengthening the network, and reducing GPU usage.

First, a set of images is randomly extracted from the tomato dataset. Then, four images are randomly selected, scaled randomly, distributed randomly, and then concatenated to form a new image. This operation is repeated for the batch size duration. Finally, the augmented mosaic data is fed into the neural network for training [18]. Figure 7 depicts an illustration of the mosaic data augmentation operation.

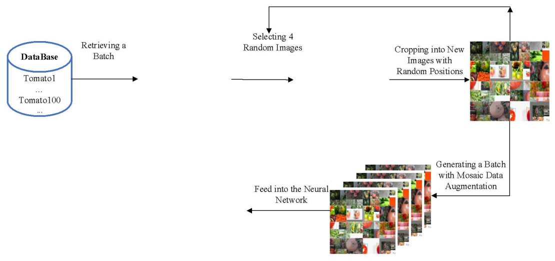

DataBase
Tomato1
...
Tomato100
...
Retrieving a Batch
Selecting 4 Random Images
Cropping into New Images with Random Positions
Generating a Batch with Mosaic Data Augmentation
Feed into the Neural Network

Fig. 7. Example of Mosaic Augmentation

I. *Evaluation Techniques*

Evaluation techniques are employed to assess the performance of the model. The evaluation metrics utilized include recall, precision, F1-score, mAP (mean average precision), and accuracy. Recall is defined as the ratio of the total number of true positive objects to the total number of actual positive objects. A high recall indicates that the class is correctly identified. Precision is calculated by dividing the total number of true positive samples by the total number of samples predicted as positive. The F1-score aims to compute a combination of precision and recall, using the harmonic mean of both. Mean average precision (mAP) is the average value of the average precision (AP), which constitutes an evaluation metric for assessing the performance of object detection [19].

$$Recall = \frac{TP}{TP+FN} \quad (20)$$

$$Precision = \frac{TP}{TP+FP} \quad (21)$$

$$F-measure = \frac{2*(Recall*Precision)}{Recall+Precision} \quad (22)$$

$$AP = \sum (Recall_{n+1} - Recall_n)\ x\ precision_{interp}\ x\ (Recall_n)) (23)$$

$$P_{interp}(R_{n+1}) = \underset{\tilde{r} \geq r_{n+1}}{max} P(\tilde{r}) \quad (24)$$

$$mAP = \frac{1}{N}\sum_{i=1}^{N} APi \quad (25)$$

J. *Research Phases*

The stages in this research process serve as the foundation and steps to address the research problem. The flowchart for this research can be seen in Figure 8.

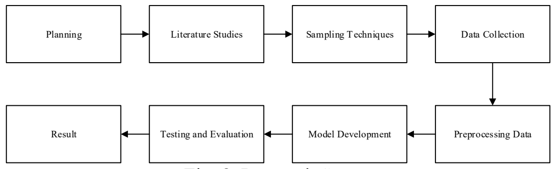


Fig. 8. Research Stages

1). Planning

By defining the research problem, it is possible to establish the object and limitations of this study.

Identifying issues and conducting an object identification makes this research more focused and provides clear objectives. As a result, limitations or scopes emerge to maintain the research's focus, with goals set as the targets to be achieved.

2). Literature Studies

The literature review phase involves searching for previous studies to be used as references in the research and reviewing theories to understand how the methods obtained from the research process work. In this stage, there is also an in-depth study of material related to image processing, as well as extracting and detecting objects using CNN and YOLO methods. The literature used includes discussions, reading books, journals, articles, and websites that discuss relevant materials.

3). Sampling Techniques

Probability sampling is a sampling technique that provides an equal chance for every element or member of the population to be selected as a sample. Stratified sampling is used to separate the dataset, reducing the problem of random sampling in datasets with imbalanced class distributions. In this method, subjects are initially grouped into different classifications.

4). Data Collection

The data sources consist of both secondary data and primary data. Secondary data refers to publicly available data that the researcher then uses and labels to fulfill the research data requirements. On the other hand, primary data refers to data that is directly collected by the researcher.

5). Preprocessing Data

The objects are selected through a categorization process, which involves adapting to the chosen objects. Next, the images are resized to a consistent size to facilitate the training process. Subsequently, data annotation is performed. The Annotation Stage involves providing information or labels for each image data. These labels include the coordinates, confidence level of the bounding boxes, and the type of tomato for each tomato object present in the image. This information is used for the training and testing processes. Images that may potentially introduce noise are not included in the labeling.

6). Model Development

The process of building this model involves training the data using both training and validation sets. The training set is a collection of data used to train the model to learn hidden features/patterns within the data. The validation set is a separate collection of data from the training set and is used to validate the performance of the model during training. The validation process provides information that helps adjust the model's hyperparameters and configurations appropriately. It's like feedback that tells whether the training is heading in the right direction or not. The main idea behind splitting the dataset into a validation set is to prevent the model from overfitting. The test set is a separate collection of data used to evaluate the model. Simply put, it answers questions about how well the model performs.

7). Testing and Evaluation

The testing process is carried out by testing the method using several new images. Real-time testing is conducted to determine the distance limit for objects to still be recognizable. Meanwhile, the evaluation process is performed to assess the accuracy level of the model's performance and also to analyze the results of the validation testing. Program performance validation is done by calculating the Intersection over Union (IoU) value between the predicted bounding box and the actual bounding box, as well as calculating the accuracy in object classification.

8). Result

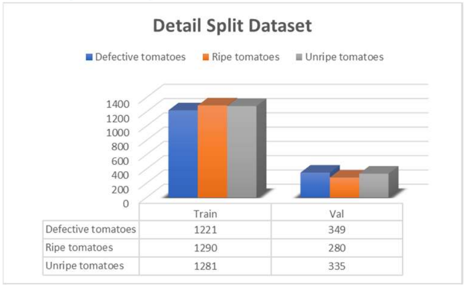
The result is information about the localization and recognition of the object, indicating whether it belongs to the categories of defective, ripe, or unripe.

III. Result And Discussion

In this section, the implementation and testing of the employed method are elucidated. Additionally, detailed results of the research are provided.

A. *Training Data*

The data was collected with three classes: ripe tomatoes, unripe tomatoes, and defective tomatoes. In total, the Researcher gathered a dataset consisting of 1,810 images. After the annotation process, the number of training data for ripe tomatoes was 1,290, unripe tomatoes were 1,281, and defective tomatoes were 1,221. For the validation set, there were 280 ripe tomatoes, 335 unripe tomatoes, and 349 defective tomatoes. The detailed data distribution can be seen in Figure 9, with the distribution ratio shown in Figure 10. The data was randomly split with 80% for training-set and 30% for validation-set. The data used for testing is obtained separately to directly assess the model's performance in recognizing new data. The testing data is also deliberately perturbed to thoroughly evaluate the model.



	Train	Val
Defective tomatoes	1221	349
Ripe tomatoes	1290	280
Unripe tomatoes	1281	335

Fig. 9. Detail Split Data

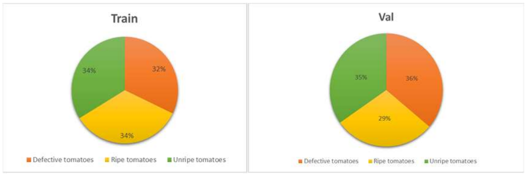


Fig. 10. Data Distribution Ratio

B. *Model Training*

The data is trained to learn patterns in the training data. The YOLOv7-tiny architecture is used with the initial weights from a pre-trained YOLOv7-tiny model in PyTorch format, and transfer learning is applied to new data. The weights are stored in the PyTorch model, with each weight being saved via Exponential Moving Average (EMA) based on the loss value calculated using Complete Intersection Over Union (CIOU).

TABLE I. TABLE HYPERPARAMETER

NO	Hyperparameter	
	Parameter	***Value***
1.	lr0	0.01
2.	lrf	0.01
3.	momentum	0.937
4.	weight_decay	0.0005
5.	warmup_epochs	3.0
6.	warmup_momentum	0.8
7.	warmup_bias_lr	0.1

The model was trained for a total of 300 epochs. From the training results, it can be observed that classification, precision, recall, mAP@0.5, mAP@0.5:95 continuously improved, indicating that the model is learning better over time. Validation box and validation classification experienced a decrease, indicating that the predictions and actual data are becoming relatively consistent. Meanwhile, in the case of validation objectness or confidence in objects, it started to increase, suggesting that the model is entering a memorization phase. Upon closer inspection, the following are the results for validation objectness, validation classification, and validation box.

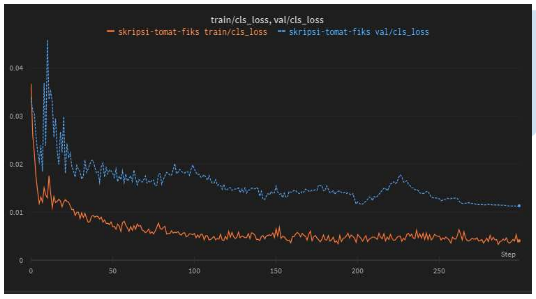


Fig. 11. Class loss

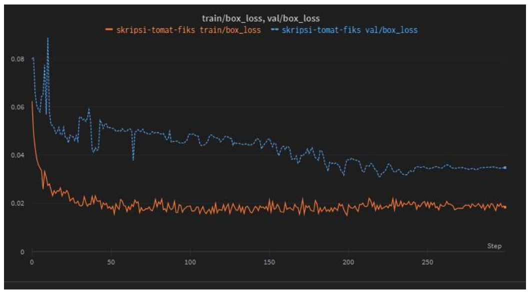


Fig. 12. Box loss

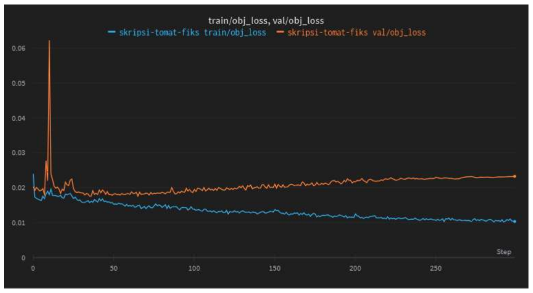


Fig. 13. Obj loss

In the above figure, it can be observed that the curve appears to be normal, and there are no indications of overfitting. Overfitting is a condition where the loss of training data and validation data start to diverge. The closer the distance between the loss of training data and validation data, the better the model performs because the loss values are decreasing. However, in the case of validation objectness, the curve starts to rise, but it is still in an early stage. This training obtained a confusion matrix value as shown in Figure 14.

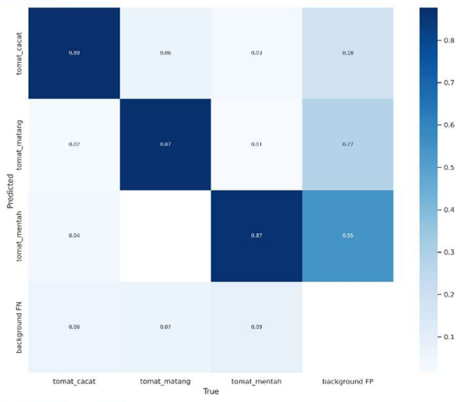


Fig. 14. Training Confusion Matrix

The above figure shows that the actual data predicted as background receives a relatively low value, indicating a low likelihood of objects being undetected, which is a positive outcome. However, there are instances where the background is predicted as having objects from the class of unripe tomatoes. This occurs because the unripe tomatoes in the training data were exposed to high light conditions, causing the model to learn from bright colors. As a result, a bright background resembling an object can confuse the model.

This model achieved a recall value of 0.97, precision of 1.0, a PR-curve value of 0.838, and an F1-score of 0.81, as depicted in the following figure.

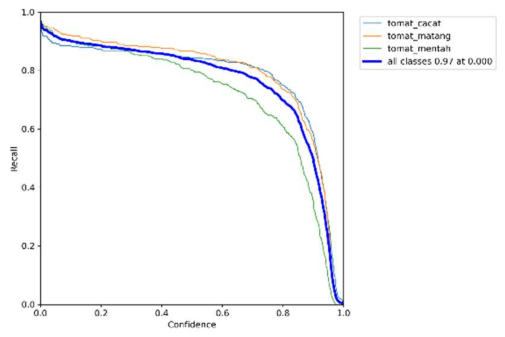


Fig. 15. Recall training-validation

In Figure 15, the model's recall performance during training and validation is depicted. The achieved recall value is 0.97, indicating that the model successfully identified 97% of the relevant instances in the dataset. This suggests a high sensitivity of the model in capturing instances of interest.

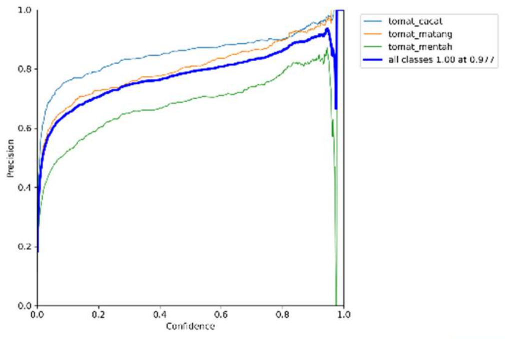


Fig. 16. Precision training-vatlidation

Figure 16 illustrates the precision performance of the model during training and validation. The precision value of 1.0 signifies perfect accuracy in positive predictions. Every instance predicted as positive by the model was indeed a true positive, demonstrating the high precision of the model.

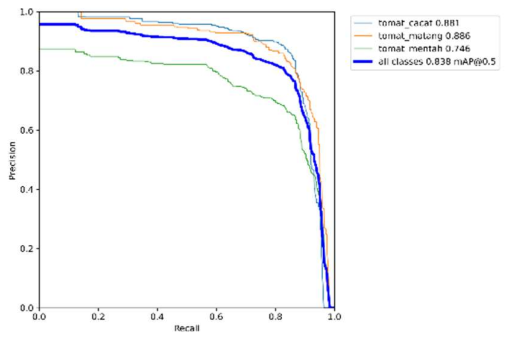


Fig. 17. PR-Curve training-validation

The Precision-Recall curve (PR-Curve) in Figure 17 provides a visual representation of the trade-off between precision and recall for different threshold values during training and validation. The PR-Curve value of 0.838 suggests a good balance between precision and recall, reinforcing the model's effectiveness in handling positive instances while minimizing false positives.

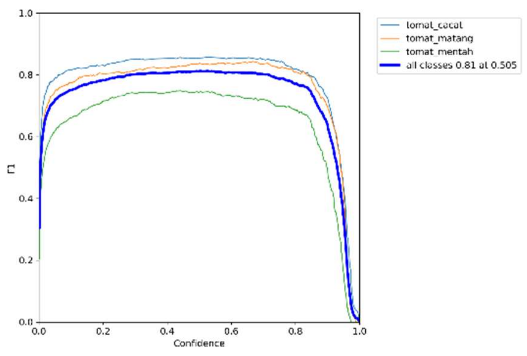


Fig. 18. F1-score training-validation

Figure 18 displays the F1-score performance of the model during training and validation. With an F1-score of 0.81, the model demonstrates a harmonious balance between precision and recall. This metric consolidates the model's ability to provide accurate positive predictions while effectively capturing relevant instances in the dataset.

C. *Method Testing Results*

New data has been added, consisting of 10 samples per class, for the testing process. These data have also been intentionally distorted to assess the model's performance. The data was captured with the camera facing the objects directly, under bright sunlight at 16:00 (GMT+7). The distortions applied include motion blur, increased brightness, and decreased brightness. Further details about the testing data can be found in Table II.

TABLE II. TESTING DATASET

Dataset	**Parameter**	
	intensity	***Data***
Original Images	-	30 Data
Darkened Images	Intensity dark (n) = 30	30 Data
	Intensity dark (n) = 50	30 Data
	Intensity dark (n) = 70	30 Data
	Intensity dark (n) = 90	30 Data
Brightened Images	Intensity bright (n) = 30	30 Data
	Intensity bright (n) = 50	30 Data
	Intensity bright (n) = 70	30 Data
	Intensity bright (n) = 90	30 Data
Motion Blurred Images	Intensity blur (n) = 0.3	30 Data
	Intensity blur (n) = 0.5	30 Data
	Intensity blur (n) = 0.7	30 Data
	Intensity blur (n) = 0.9	30 Data

In order to calculate the average IoU, a small threshold value is set, with an IoU threshold of 0.1. In addition to finding the average IoU, the detection speed of the model is also evaluated. To compute the average

IoU, it is necessary to determine the values of x_min, y_min, x_max, y_max based on the x_center, y_center, width, and height. Ground truth is also needed to calculate the IoU. Therefore, testing data is annotated and resized beforehand. Subsequently, the x_min, y_min, x_max, y_max dimensions are matched between the predicted bounding boxes and the ground truth. The results of the testing can be seen in Table III.

TABLE III. TESTING RESULT

Dataset	**Parameter**				
	n	***Accuracy***	***Average IoU***	***T (ms/s)***	***Average confidence***
Original Images	-	90,6%	0.90	2.7	0.83
Darkened Images	n=30	90,6%	0.90	2.6	0.82
	n=50	90,6%	0.89	2.6	0.80
	n=70	90,9%	0.89	2.5	0.80
	n=90	93,5%	0.88	2.4	0.76
Brightened Images	n=0.3	90,3%	0.89	2.6	0.71
	n=0.5	90,3%	0.87	2.2	0.79
	n=0.7	93,5%	0.90	1.9	0.81
	n=0.9	93,9%	0.90	2.4	0.83
Motion Blurred Images	n=30	82,3%	0.87	2.4	0.87
	n=50	81,4%	0.84	2.0	0.82
	n=70	74,2%	0.81	2.5	0.69
	n=90	75,2%	0.78	2.5	0.47

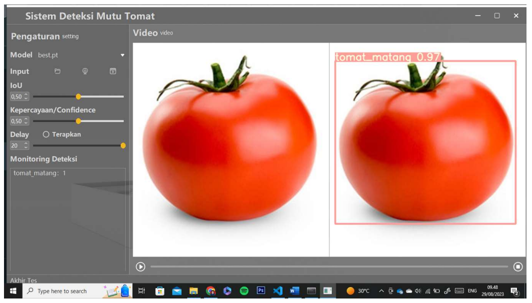
In addition to testing with added distortions, real-time testing was also conducted to determine the maximum distance at which objects can be reliably detected. The results of this testing can be found in Table IV. Real-time testing used an IoU threshold of 0.41 with a confidence threshold of 0.68. The use of relatively high values aims to tighten the model to minimize multiple bounding boxes. This testing was performed at distances of 20cm, 30cm, 40cm, 50cm, 60cm, 70cm, 80cm, and 90cm. The real-time testing utilized a Redmi 5A camera, conducted indoors at 09:30 AM (GMT+7).

TABLE IV. REAL-TIME TESTING RESULT

NO.	**Result**	
	Object classes	***Maximum Distance***
1.	Ripe Tomatoes	90Cm
2.	Unripe Tomatoes	90Cm
3.	Defective Tomatoes	70Cm

D. Interface Implementations

The system utilizes PyQt for its implementation. All features in the system are tailored to the needs of this research. This system is designed to facilitate ease of use and provide a more interactive way to view detection results. The system consists of a single page with various features or buttons utilized in the detection process, such as accessing directories and utilizing the camera directly. The system's page or GUI can be seen in Figure 15.



Model best.pt
IoU 0,50
Kepercayaan/Confidence 0,50
Delay 20 Terapkan
Monitoring Deteksi
tomat_matang: 1

Fig. 15. Interface Implementation

IV. CONCLUSION AND FUTURE WORK

This section contains the conclusions drawn from the conducted research, along with recommendations for future studies.

A. Conclusion

From the conducted research, the following conclusions can be drawn:

1). YOLO provides bounding boxes for each recognized object class. If bounding boxes overlap beyond a certain threshold, they are suppressed using non-maximum suppression. There can be multiple bounding boxes within a single object if the threshold is set to a lower value than the prediction score. In real-time object detection, confidence scores fluctuate due to moving objects. In real-time scenarios, frames undergo continuous movement, requiring the model to analyze rapidly and consistently. Real-time detection is also influenced by the computational power of the hardware. Higher GPU capabilities contribute to faster detection speeds;

2). The training model yielded a recall value of 0.97, precision of 1.0, a PR-curve score of 0.838, and an F1-score of 0.81. These results indicate that the model learned effectively;

3). The testing was conducted in two ways: through digital images and in real-time to measure distances. The results from original images showed an accuracy of 90.6%, with an average IoU of 0.90 and a detection time of 2.7 seconds. In images with added brightness, the average accuracy was 91.2%. Images with reduced brightness had an average accuracy of 92%, and images with blur interference had an average accuracy of 78.2%. These testing results indicate that the created model performs quite well. In real-time testing, ripe tomatoes were detected at a maximum distance of 90cm, unripe tomatoes at 90cm, and defective tomatoes at 70cm. Testing on non-tomato objects requires stricter parameter application because the training data lacks sufficient non-tomato object samples.

B. Future Work

The research provides valuable recommendations for the development of a YOLO-Tomat model. Firstly, if there is an intention to create this model, it is advisable to conduct a retraining process using this specific YOLO-Tomat model. Additionally, the integration of a more diverse dataset is crucial to significantly enhance detection accuracy. It is essential to exercise careful consideration in the selection of the training dataset, as the quality of the data has a profound impact on the model's overall performance.

Furthermore, the study supports the possibility of utilizing the latest versions of the YOLO architecture or even developing a custom, lighter architecture for this specific task. This architectural update can prove to be an effective strategy to enhance the model's proficiency in object detection. The development of the system can also be further refined to cater to a wider array of applications. This could include considering the integration of additional tools or relevant technologies to augment the system's capabilities. For detecting highly complex objects, it is recommended to expand the training dataset to include samples from object classes other than tomatoes. This augmentation process aids the model in comprehending and recognizing objects more effectively.

In future studies, it is advised to consider the inclusion of more complex object classes or the exploration of more challenging conditions to rigorously evaluate the model's performance. Additionally, research endeavors could delve into the possibility of detecting fruit flavors, providing invaluable insights for practical applications.

ACKNOWLEDGMENT

The authors would like to thanks Universitas Nahdlatul Ulama Sunan Giri for supporting this research.

REFERENCES

[1] F. Y. Wang, “A big-data perspective on AI: Newton, Merton, and analytics intelligence,” *IEEE Intell. Syst.*, vol. 27, no. 5, pp. 2–4, 2012, doi: 10.1109/MIS.2012.91.

[2] N. E. Paulina, Zi. emka Fitri, and A. M. N. Madjid, Abdul Imron, “Klasifikasi Kerusakan Mutu Tomat Berdasarkan Seleksi Fitur Menggunakan K-Nearest Neighbor,” *MIND J.*, vol. 6, no. 2, pp. 144–154, 2021, doi: 10.26760/mindjournal.v6i2.144-154.

[3] R. Kosasih and C. Daomara, “Pengenalan Wajah dengan Menggunakan Metode Local Binary Patterns Histograms (LBPH),” *J. Media Inform. Budidarma*, vol. 5, no. 4, p. 1258, 2021, doi: 10.30865/mib.v5i4.3171.

[4] R. A. Firmansyah and E. Alfianto, “Pembuatan Haar-Cascade Dan Local Binary Pattern Sebagai Sistem Pendeteksi Halangan Pada Automatic Guided Vehicle,” *Simetris J. Tek. Mesin, Elektro dan Ilmu Komput.*, vol. 9, no. 2, pp. 1073–1082, 2018, doi: 10.24176/simet.v9i2.2562.

[5] N. ul Hafidhoh and S. E. Sukmana, “Deteksi Pemain Basket Terklasifikasi Berbasis Histogram of Oriented Gradients,” *J. Inf.*, vol. 3, no. 1, pp. 6–11, 2018, doi: 10.25139/ojsinf.v3i1.635.

[6] Y. A. Hasma and W. Silfianti, “Implementasi Deep Learning Menggunakan Framework Tensorflow Dengan Metode Faster Regional Convolutional Neural Network Untuk Pendeteksian Jerawat,” *J. Ilm. Teknol. dan Rekayasa*, vol. 23, no. 2, pp. 89–102, 2018, doi: 10.35760/tr.2018.v23i2.2459.

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.

[8] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” no. July, 2022, doi: 10.48550/arXiv.2207.02696.

[9] A. Murdiati and Amaliah, *Panduan Penyiapan Pangan Sehat Untuk Semua*, 2nd ed. Prenada Media, 2013.

[10] E. Lubis Riyanty, *Bercocok Tanam Tomat Untung Melimpah*. Jakarta: Bhuana Ilmu Populer, 2020.

[11] L. Liu *et al.*, “Deep Learning for Generic Object Detection: A Survey,” *Int. J. Comput. Vis.*, vol. 128, no. 2, pp. 261–318, 2020, doi: 10.1007/s11263-019-01247-4.

[12] Y. Xia, M. Nguyen, and W. Qi Yan, *Image and Vision Computing: 37th International Conference, IVCNZ 2022*. 2023.

[13] A. B. Malayeri and M. B. Khodabakhshi, “Concatenated convolutional neural network model for cuffless blood pressure estimation using fuzzy recurrence properties of photoplethysmogram signals,” *Sci. Rep.*, vol. 12, no. 1, pp. 1–14, Apr. 2022, doi: 10.1038/s41598-022-10244-6.

[14] X. Wang and J. Song, “ICIoU: Improved Loss Based on Complete Intersection over Union for Bounding Box Regression,” *IEEE Access*, vol. 9, pp. 105686–105695, 2021, doi: 10.1109/ACCESS.2021.3100414.

[15] Z. Huang *et al.*, “An Improved Method for Ship Target Detection Based on YOLOv4,” *Appl. Sci.*, vol. 13, no. 3, p. 1302, 2023, doi: 10.3390/app13031302.

[16] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, “Distance-IoU loss: Faster and better learning for bounding box regression,” *AAAI 2020 - 34th AAAI Conf. Artif. Intell.*, no. 2, pp. 12993–13000, 2020, doi: 10.1609/aaai.v34i07.6999.

[17] Z. Qiu, S. Rong, and L. Ye, “YOLF-ShipPnet: Improved RetinaNet with Pyramid Vision Transformer,” *Int. J. Comput. Intell. Syst.*, vol. 16, no. 1, 2023, doi: 10.1007/s44196-023-00235-4.

[18] K. Jiang *et al.*, “An Attention Mechanism-Improved YOLOv7 Object Detection Algorithm for Hemp Duck Count Estimation,” *Agric.*, vol. 12, no. 10, 2022, doi: 10.3390/agriculture12101659.

[19] Y. Umar, S. M. S. Nugroho, and R. F. Rachmadi, “Deteksi Penggunaan Helm Pada Pengendara Bermotor Berbasis Deep Learning,” 2020. https://www.its.ac.id/komputer/wp-content/uploads/sites/28/2018/03/Buku-Yusuf-Umar-Hanafi-07211640000006-Supeno-M-Fuad-R.pdf (accessed Mar. 20, 2023).