Analyzing the Impact of Histogram-Based Image Preprocessing on Melon Leaf Abnormality Detection Using YOLOv7

Sahrial Ihsani Ishak, Sri Wahjuni, Karlisa Priandana*
School of Data Science, Mathematics and Informatics, IPB University, Bogor, Indonesia

***Abstract*—This study aims to analyze and implement image preprocessing techniques to improve the performance of melon leaf abnormality detection using the YOLOv7 algorithm. A total of 521 abnormal melon leaf images were processed using augmentation and three preprocessing methods: Averaging Histogram Equalization (AVGHEQ), Brightness Preserving Dynamic Histogram Equalization (BPDFHE), and Contrast Limited Adaptive Histogram Equalization (CLAHE), then compared with the original dataset. Modeling was conducted in three stages: initial training with an 80:20 split and default YOLOv7 augmentation; hyperparameter tuning via cross-validation using a 90:10 split without augmentation; and final training using the best parameters with augmentation reactivated. The models were evaluated using ensemble learning. Results showed mAP ranged from 58.6% to 66.3%, accuracy from 80.7% to 84.9%, and detection time from 9.8 to 20 milliseconds. Preprocessing improved mAP and detection time, though it had little effect on accuracy. The best performance was obtained with a kernel size of 3 and a learning rate of 0.001, while changes in activation function, pooling, batch size, and momentum had minimal impact. The top models, trained with maximum epochs and standard augmentation, achieved mAP of 84.12%, accuracy of 91.19%, and detection time of 4.55 milliseconds. Models using early stopping (patience = 300) reached mAP of 81.57%, accuracy of 92.23%, and detection time of 5.03 milliseconds. The best model outperformed previous works, which reported only 48.85% with Faster R-CNN, 33.16% with SSD, and 16.56% with YOLOv3. Although histogram-based preprocessing methods mainly enhanced inference speed, the overall improvements to YOLOv7 significantly boosted detection performance.**

Keywords*—Leaf abnormality; melon; image preprocessing; YOLOv7

I. Introduction

The agricultural sector significantly contributes to Indonesia's GDP by creating jobs and increasing export values [1], supports broader economic expansion through regional improvements in fruit commodity productivity [2], and has helped mitigate the negative impact of the COVID-19 pandemic on economic growth via strong export performance [3]. According to information from the Indonesian Central Statistical Agency, fruit commodity production in Indonesia during 2016-2022 increased [4]. Nevertheless, there are commodities whose production is fluctuating. One of them is the melon commodity [4]. Fluctuations in melon production stem from suboptimal growing conditions, such as nutrient allocation affected by pruning and fruit thinning practices [5] and adverse environmental factors like temperature and humidity variations [6].

There are two causes of malnutrition in melon crops: the lack of nutrient intake received by the plant and the presence of pest disorders and diseases that affect the melon plant periodically. This can lead to crop failure if physiological disorders during melon growth are not addressed promptly [7], emphasizing the importance of timely pruning and fruit regulation to support healthy development. Meanwhile, environmental factors such as temperature, light, and water availability also play a crucial role in plant growth [8]. Identifying such disorders often requires sending samples to laboratories for testing, which can be time-consuming [9] and further supports [10] the diagnostic process for plant abnormalities. Information technology is needed to help identify anomalies that occur in melon plants effectively and efficiently. Artificial intelligence technology could be the solution to this problem [11].

Artificial intelligence (AI) is a field of science that integrates machine and human intelligence, which has been applied in various sectors, including agriculture [11], contributes to the development of intelligent algorithmic systems [12], and provides the foundation for advanced deep learning methods [13]. Deep learning, which evolved from machine learning, builds algorithms from existing data by mimicking neural networks [14], using statistical and computational models [15], and emphasizing efficient learning processes for data classification and prediction [16]. Unlike conventional machine learning, which treats feature extraction as a separate process, deep learning enhances this by learning feature representations directly from raw data [17]. One of the prominent applications of deep learning is object detection, a task that involves identifying and localizing objects within an image. Current object detection approaches are categorized into one-stage methods, prioritizing faster inference speeds [18], and two-stage methods, offering higher detection accuracy through refined region proposals and classification [19].

The performance of object detection models can be influenced by the image preprocessing techniques applied beforehand, as specific preprocessing steps can improve the generalization ability of over-parameterized neural networks [20]. In contrast, others affect the accuracy of convolutional neural network-based recognition systems [21]. In the context of melon leaf analysis, differences between normal and abnormal leaves can be identified through leaf color, shape, and texture, which are often associated with anatomical resistance to fungal

infection [22] and market-related disease symptoms in melon and related crops [23]. An image itself is composed of a grid of pixels where each pixel encodes intensity or color information; this concept underpins many image analysis processes, as elaborated in foundational image processing literature [24], methods for machine vision image acquisition and preprocessing [25], and comprehensive digital image processing frameworks [26].

The frequency distribution of pixel intensity in an image can be analyzed using a histogram, a fundamental technique in image processing as described in works such as Gonzalez and Woods' Digital Image Processing [24], Sinha's treatment of machine vision systems [25], and Pratt's comprehensive guide on digital image analysis [26]. Histogram-based processing is widely used to enhance image contrast and emphasize key differences between objects and their backgrounds [24], beneficial for distinguishing between normal and abnormal melon leaves. Several adaptive histogram equalization techniques have been developed to improve pixel-level contrast and image quality, especially in plant health detection. One such method is AVGHEQ, which focuses on contrast enhancement while maintaining brightness consistency [27]. Another technique, CLAHE, prevents noise over-amplification in homogeneous areas by limiting the histogram contrast [28]. Additionally, BPDFHE employs fuzzy logic to enhance image contrast while preserving natural brightness levels [29]. These preprocessing methods are expected to increase the effectiveness of detection models by providing more apparent visual differentiation between healthy and unhealthy leaves.

Various image preprocessing techniques can significantly impact the performance of the object detection model. These techniques play an essential role in improving the quality of the input image, thereby increasing the accuracy and efficiency of the detection process. On the other hand, hyperparameters also influence the performance of the object detection model. The detection model used in this study is YOLOv7, which surpasses other algorithms such as YOLOR, YOLOX, Scaled-YOLOv4, YOLOV5, DETR, DETR Deformable, DINO-5scale-R50, ViT-Adapter-B, and many other objects detectors, especially in terms of detection speed [30]. This study investigates how picture preprocessing approaches and hyperparameter optimization affect the effectiveness of models for detecting melon leaves. Unlike many earlier studies, which frequently disregard these elements, this study investigates their significance in enhancing model accuracy and efficiency. By addressing these critical features, the study hopes to provide the groundwork for constructing a more effective and efficient plant anomaly detection model, particularly for melon crops, which have gotten less attention in previous studies.

The structure of this paper is as follows: Section II reviews relevant literature, Section III outlines the research methodology, Section IV presents the results and discussion, and Section V provides the conclusion.

II. Related Work

Research on the identification of anomalies in tomatoes, soy, cucumbers, apples, and melons shows that it is still necessary to use image preprocessing to improve model performance and model application into a mini-computer platform to evaluate models in real-time. A study by [31] demonstrated that the Faster R-CNN and Mask R-CNN methods effectively identified tomato crop diseases, although some misclassifications were still observed. Similarly, [32] identified pests and diseases in tomato plants and reported that an improved version of YOLOv3 achieved the highest accuracy of 92.39% with a detection speed of 20.39 milliseconds. In the case of soybean plants, disease detection using the Multi-Feature Fusion Faster R-CNN method yielded an optimal mean Average Precision (mAP) of 83.34% [33]. Another study [34] employed the MTC-YOLOv5n method with enhancements to reduce image noise and improve the detection of small objects. This approach resulted in a compact model size of 4.7 MB, an mAP of 84.9%, and a frame rate of 143 frames per second (FPS). In addition, [35] proposed the MGA-YOLO method for detecting apple diseases based on leaf images, achieving an mAP of 89.3%, a minimal model size of 10.34 MB, and a peak FPS of 84.1 on a GPU. The model was also tested on smartphones, reaching a frame rate of 12.5 FPS.

This research extends the work presented in [36], which compared three methods for plant disease detection: Faster R-CNN, SSD, and YOLOv3. The study found that Faster R-CNN achieved the highest mean Average Precision (mAP) at 48.85%, while YOLOv3 demonstrated the shortest inference time at 0.5 seconds. In terms of CPU usage, SSD performed most efficiently, whereas YOLOv3 offered a balance between fast inference and moderate CPU consumption. Additionally, the study highlighted that preprocessing techniques could enhance the performance of object detection models [36]. Building on these findings, the present research employs YOLOv7 for its efficient inference time and moderate CPU usage. It further investigates the impact of a histogram-based preprocessing technique on model performance.

III. Research Methodology

This research comprises of five main steps: (1) data preparation; (2) modeling and evaluation of phase 1; (3) modelling and evaluation of phase 2; (4) modelling and evaluation of phase 3; and (5) comparison with the previous research in [36]. Before delving into the main topic, it is crucial first to discuss the background of YOLOv7 and the image processing techniques used, including AVGHEQ, BPDFHE, and CLAHE.

A. YOLOv7

You Only Look Once (YOLO) is a one-stage object detection algorithm consisting of the backbone, neck, and head structure. The backbone layer is responsible for feature extraction from the input image. Then, the results are passed to the neck layer, which generates pyramid features, allowing the system to detect objects at different scales. The head layer is the final layer for detecting classes and bounding boxes. YOLO has an architecture based on convolutional networks. YOLO's detection network comprises 24 convolutional layers and 2 fully connected layers. Alternating 1 × 1 convolutional layers reduces the feature space from the previous layers. The convolutional layers have been pre-trained on the ImageNet classification task with an image resolution of 224 × 224 [30].

YOLOv7, released in 2022 [30], is an algorithm of the YOLO model. To be more precise, the author creates several

training techniques known as "bag-of-freebies" that consist of modules and optimization techniques that significantly boost detection accuracy without raising the cost of inference. The network architecture uses the E-ELAN technique, which manages the longest and shortest gradient pathways to help a deep model learn and converge more efficiently. The E-ELAN approach shuffles and merges schemes to integrate the features of the groups to improve the learned features. It also lowers the computation cost and parameter count. Furthermore, YOLO-v7 presents several other training bag-of-freebies, such as 1) designing the architecture of the planned re-parameterized convolution using Connection-Aware RepConv (RepConvN), 2) implementing a new labeling technique that directs the lead and auxiliary heads; and 3) normalizing data in conv-bn-activation topology 4) Convolution on a feature map that combines addition and multiplication 5) Making the final inference using the EMA model.

YOLOv7 is a version of YOLO released in 2022. It can outperform all existing object detectors in speed and accuracy, with a speed of 5 to 160 FPS and the highest accuracy of 56.8% AP on the V100 GPU [30]. YOLOv7 also beats various other detectors, such as YOLOR, YOLOX, and YOLOv5, in terms of speed and accuracy, as seen in Table I. The overall architecture of YOLOv7 can be seen in Fig. 1.

TABLE I. YOLOv7 JUSTIFICATION

Model	Inference Time (millisecond)	AP (%)
YOLOv7	~5–7	~57
YOLOR	~9–11	~56
PPYOLOE	~11–13	~54
YOLOX	~15–17	~53
Scaled-YOLOv4	~23–25	~52
YOLOv5 (r6.1)	~29–31	~51

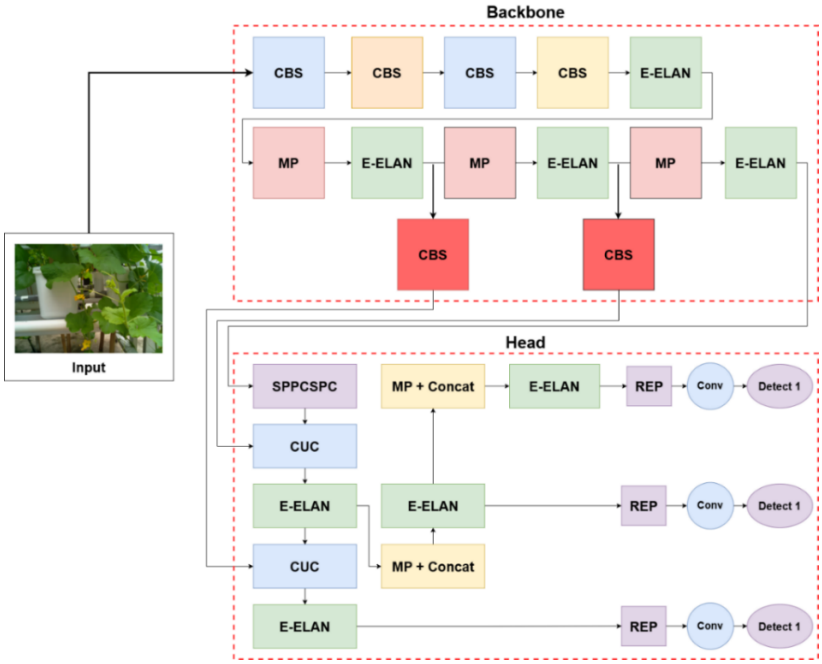


Fig. 1. YOLOv7 architecture.

B. AVGHEQ

AVGHEQ aims to increase image contrast while keeping the average brightness of the image output unchanged as much as possible. First, the input image is stretched on each color channel to correct any distortion from the unwanted environment. Then, it changed the color format from RGB to HSI [24]. After that the histogram was averaged before it was used in the equalization operation until the brightness error was reduced to a minimum and the entropy was maximized as much as possible. Next, normalization operations are performed, and last, the histogram and changes are remapped, and the HSI color format to RGB as the output image. The overall stages of AVGHEQ can be seen in Fig. 2.

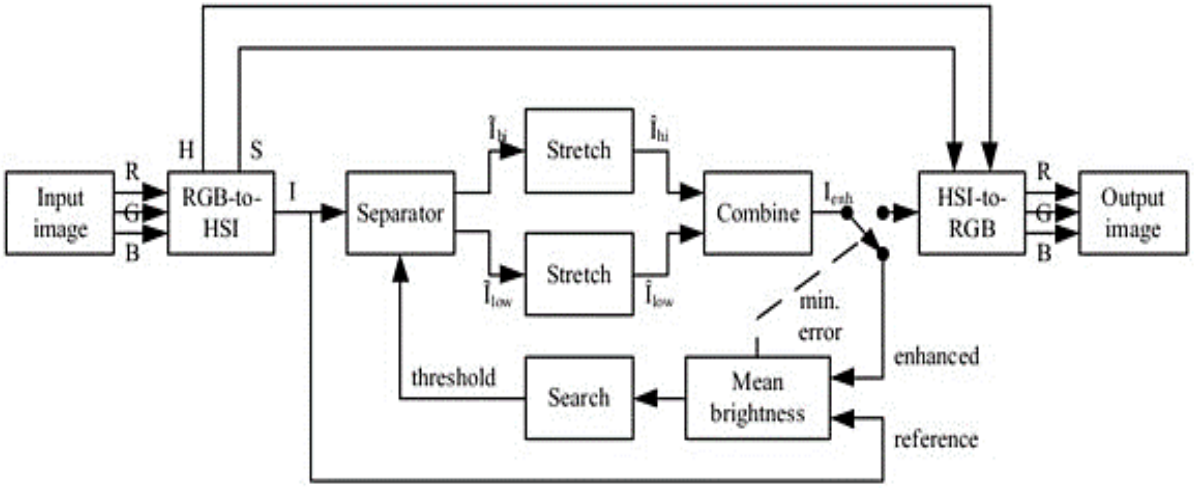


Fig. 2. AVGHEQ process [27].

C. BPDFHE

BPDFHE uses fuzzy statistics for digital image representation and processing. BPDFHE works in an abstract domain, allowing the process to be managed effectively at the end of the grayscale value, improving overall performance [29]. BPDFHE consists of four stages, namely, 1) Fuzzy Histogram computing, 2) Histogram Partitioning, 3) Dynamic Histogram Equalization of Partitioning, and 4) Image brightness normalization.

The process begins by calculating the fuzzy histogram, which uses the fuzzy membership function to handle the distribution of gray intensity values more subtly, resulting in a smoother and more informative histogram than traditional methods. Next, the histogram is partitioned based on the local maximum point found by derivative analysis, dividing the histogram into several sub-histograms. Each sub-histogram is then individually equalized using the Dynamic Histogram Equalization (DHE) technique to increase contrast without changing the average brightness of the image. This process includes adjusting the dynamic range and remapping the intensity of each partition. The final stage is brightness normalization to ensure the average brightness of the output image remains consistent with the input, resulting in a sharper image with a more balanced intensity distribution. The overall stages of BPDFHE can be seen in Fig. 3.

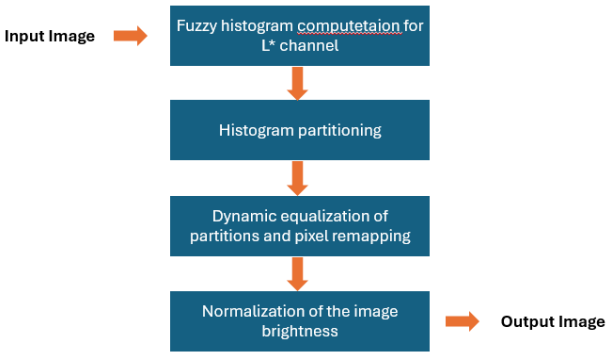


Fig. 3. BPDFHE process.

This technique effectively addresses the challenge of enhancing low-contrast images and optimizing the visual perception of details across different image regions. By carefully controlling the equalization process at the sub-histogram level, BPDFHE preserves the integrity of image information, minimizing the risk of over-enhancement or artifacts that may occur with other conventional techniques.

D. CLAHE

CLAHE is an adaptive method of *histogram equalization* followed by *thresholding* to aid in the dynamics of preservation of local contrast features of an image [37]. First, the *input* image is partitioned into several sub-images sized M × N. Then, calculate the histogram for each sub-image. Next control the contrast of the histogram with clips for each sub-image. The number of pixels present in the sub-image is distributed at each degree of grayness. Then calculate the *clip* limit from the histogram. On the original histogram, pixels will be limited if the number of pixels is greater than N_{CLIP}. The number of pixels is evenly distributed into each degree of grayness (N_d) defined by the total number of pixels constrained (N_{TC}) [37].

E. Data

This study uses secondary data from a melon leaf image from previous studies taken in 2022 [36]. The entire dataset consists of 522 images, which have been labeled and processed in TXT format. Each image has a resolution of 5 megapixels, with width and height of 2592 and 1944 pixels, respectively. The labels are divided into two classes: Abnormal and Normal. Fig. 4 shows a sample image for each class.

F. Data Preparation

This research contributes to the data preparation stage. Data preparation is a heavy challenge in deep learning technology to produce an optimal model so it can be used properly [38]. For that, the data that has been obtained needs to be well prepared. The data that has been acquired is then prepared further by preprocessing and augmentation.

Fig. 4. Sample data for abnormal (yellow bounding box) and normal (green bounding box).

The leaf image is an image that exists in a natural environment with a lot of background and noise. The usual use of histogram equalization cannot address the problem, so it requires a method that can adaptatively equalize the histogram and can also be applied to colored images that have a high background complexity. These methods include AVGHEQ (Averaging Histogram Equalization), CLAHE (Contrast Limited Adaptive Histogram equalization) [28], and BPDFHE (Brightness Preserving Dynamic Fuzzy Histogram Equalization) [29].

Augmentation based on geometry is rotation and shearing. Meanwhile, non-geometric augmentation is flipping. The training data was augmented using Python programming language on the server development environment of the IPB Computer Science study program.

G. Cross-Validation

Cross-validation is a technique that aims to estimate the generalized performance of a deep learning model, avoiding overfitting and underfitting [39]. The cross-validation technique used is k-fold cross-validation. Cross-validations will be used on the training data with a set of $k = 5$. This means that the data will be divided into five equal parts. The model will then be trained in four parts, called training data, and evaluated in the fifth part, called validation data. This process is repeated five times, and the average result is used to estimate the model's performance.

H. Modeling

This study uses the YOLOv7 nano training method. The object detection modeling process is carried out in three distinct phases, each producing different models based on a specific training strategy. In the first phase, a model is trained using the default hyperparameter configuration of YOLOv7 on the complete dataset. In the second phase, a new model is developed by performing hyperparameter tuning, as listed in Table II, using the best-performing image preprocessing technique identified for the corresponding dataset. In the third phase, the model is trained using the best-tuned hyperparameters, but this time with the original default hyperparameter file configuration of YOLOv7. This last modelling phase is conducted under two training scenarios: early stopping with a patience of 300 and full training for all epochs. As in the second phase, the best image preprocessing technique for each dataset is applied.

TABLE II. PARAMETERS ON MODELING STEP

Parameter		Value
Default hyperparameters		
Batch size		8
Epoch		5000
Image size		640 × 640 piksel
Pretrained model		YOLOv7-tiny
Tuning hyperparameter		
Kernel size		[3,5,7]
Activation function		[ReLU, LeakyReLU, SiLU]
Pooling layer		[AvgPool, MaxPool]
Learning rate		[0.1, 0.01, 0.001]
Batch size		[16, 32, 64]
Momentum		[0.9, 0.93, 0.96]
Epoch max		5000
Early stopping		300
Image size		640 × 640 piksel
Pretrained model		YOLOv7-tiny

I. Model Evaluation

The three object detection models produced at each training phase will be evaluated using a set of standard performance metrics to determine the most effective model. These metrics include Mean Average Precision (mAP), Intersection over Union (IoU), accuracy, precision, recall, and F1 score. In addition, detection time and training time are also considered in the evaluation process.

Mean Average Precision (mAP) assesses how well a model can predict accurate bounding boxes across all object classes in an image or video. mAP values range from 0 to 1, where higher values indicate better detection performance [40]. mAP is calculated by averaging the Average Precision (AP) across all

classes, where AP is derived from the precision-recall curve for each class [19]. High AP scores are achieved when both precision and recall are high across various confidence thresholds [41].

The Intersection over Union (IoU) metric measures the overlap between the predicted bounding box and the ground truth bounding box, divided by the area of their union. A higher IoU indicates a better match between the predicted and actual object locations [40].

The mathematical formulations for mAP, AP, and IoU are provided in Eq. (1), (2), and (3), respectively. Meanwhile, accuracy, precision, recall, and F1 score are defined using Eq. (4) through (7) [42].

$$mAP = \frac{1}{K} \sum_{i=1}^{K} AP(i) \quad (1)$$

where, K is the number of classes, AP is the Average Precision, and i is the index for the class.

$$AP(i) = \int_{r=0}^{1} p(r)dr \quad (2)$$

where, r is Recall and p is Precision

$$IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \quad (3)$$

where, B_p represents the bounding box on predicted area and B_{gt} represents the bounding box on ground truth area.

$$acc(i) = \frac{TP(i)+TN(i)}{TP(i)+FP(i)+TN(i)+TN(i)} \quad (4)$$

$$p(i) = \frac{TP(i)}{TP(i)+FP(i)} \quad (5)$$

$$r(i) = \frac{TP(i)}{TP(i)+FN(i)} \quad (6)$$

$$F1\_Score(i) = 2 \times \frac{p \times r}{p+r} \quad (7)$$

where, i is index for class, TP is True Positive, TN is True Negative, FP is False Positive, FN is False Negative, *acc* is Accuracy, r is Recall, and p is Precision.

IV. Results and Analysis

This section presents the experimental setup, followed by the training results and evaluation of the obtained YOLOv7 models.

A. Data Preparation Results

This research utilizes secondary data obtained from the study in [36]. Further details regarding the dataset are provided in the Data section. Data preparation was carried out through a combination of augmentation techniques and histogram-based image preprocessing. The augmentation techniques applied are listed in Table III. Table IV presents the results of the data augmentation process. To increase the quantity of training data, the original dataset was augmented threefold, resulting in a total of 1,176 training images and 130 test images. The number of objects per class in the training data also increased while maintaining a balanced class distribution.

TABLE III. Augmentation Techniques

No	Technique	Value
1	*Flipping*	Horizontal and Vertical
2	*Rotation*	-15° and +15°
3	*Shearing*	±15° Horizontal and ±15° Vertical

TABLE IV. Augmentation Results

Data Type	Total Data	Total Objects
Train data	1176	6,943 (abnormal), 6,450 (normal)
Test data	130	941 (abnormal), 549 (normal)

Histogram-based image processing techniques–AVGHEQ, BPDFHE, and CLAHE–are applied to the training data. These techniques do not alter the image classes but only modify the histogram distribution of the images. The resulting changes are illustrated in Fig. 5 and summarized in Table V. Based on the results, several observations can be made. First, although BPDFHE does not appear to significantly alter image contrast visually, the channel-wise histogram values indicate an enhancement. Second, AVGHEQ noticeably improves color contrast, as supported by the quantitative data in Table V. Third, visually, CLAHE effectively reduces image noise, resulting in sharper images. Although the enhancement is not substantial, this augmentation technique leads to a modest improvement in the image histogram.

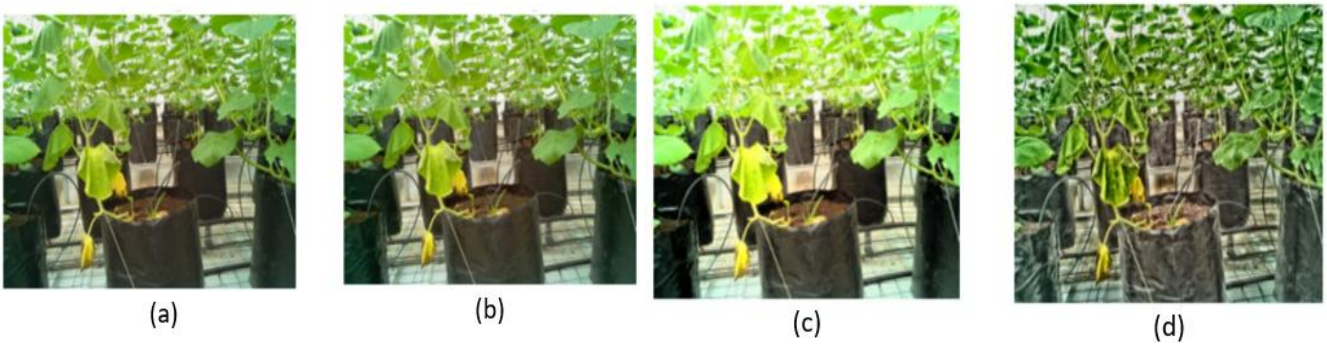


Fig. 5. (a) Comparison of the original image, (b) BPDFHE, (c) AVGHEQ, and (d) CLAHE.

TABLE V. DESCRIPTIVE ANALYSIS OF HISTOGRAM ON ALL DATASETS

Dataset	Mean (R)	Std Dev (R)	Mean (G)	Std Dev (G)	Mean (B)	Std Dev (B)
Original	96.20	51.1	120.30	54.56	77.20	59.08
BPDFHE	100.8	55.06	129.40	58.04	80.57	62.40
AVGHEQ	125.44	71.38	155.44	80.53	100.50	76.42
CLAHE	101.92	59.34	125.85	59.99	85.28	62.73

B. Cross-Validation Results

Each dataset generated through the cross-validation process is divided into training and validation sets. Fig. 6 illustrates the data distribution for each fold based on the original dataset. On average, each training set contains approximately 375 images, while each validation set contains about 94 images. In terms of object instances, the training data includes an average of 2,362 abnormal objects and 1,882 normal objects per fold. Meanwhile, the validation data includes an average of 590 abnormal objects and 471 normal objects per fold.

Fig. 7 illustrates the data distribution for each fold in the preprocessed dataset. On average, each training set contains 941 images, while each validation set includes 235 images. Regarding object instances, the training data contains an average of 5,554 abnormal objects and 5,160 normal objects per fold. In the validation data, the average is 1,389 abnormal objects and 1,291 normal objects per fold.

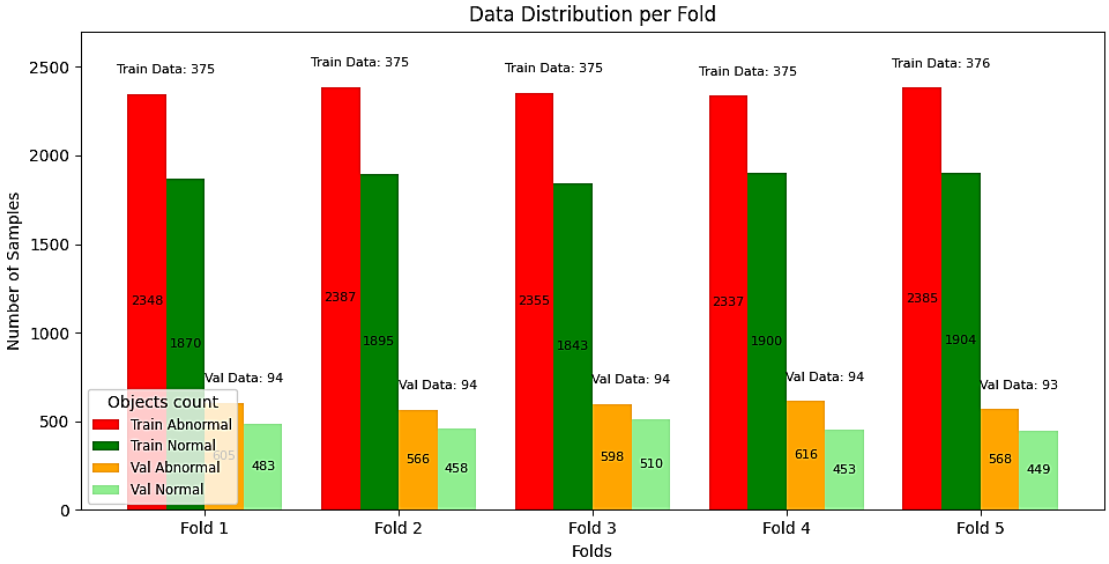


Fig. 6. Cross-validation result on original data.

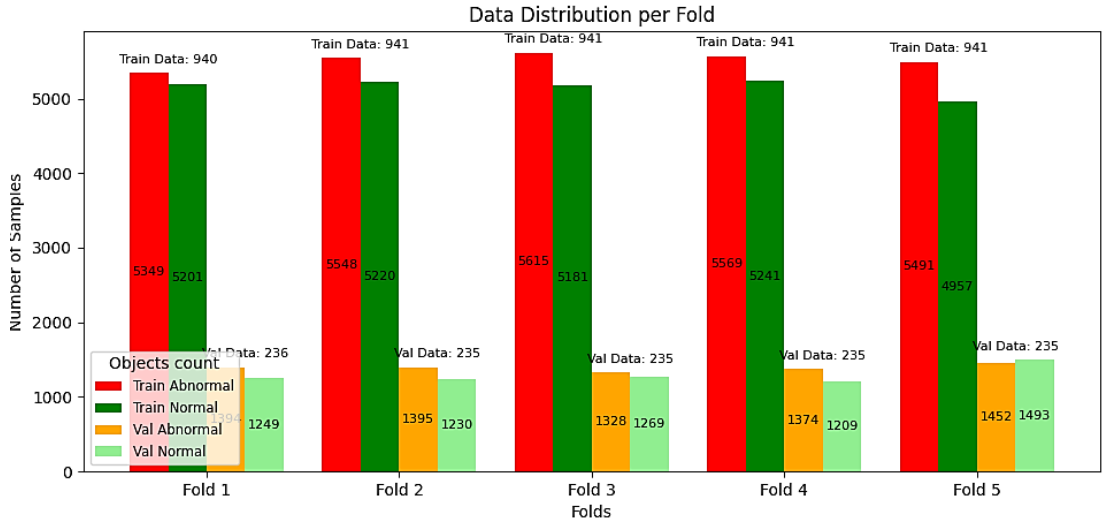


Fig. 7. Cross-validation result on preprocessed data.

C. Modelling Results

The object detection modeling process produced results related to model size and training time. In the first modelling phase (model 1), only the model size was recorded. In contrast, the second and third modelling phases (model 2 and model 3), both model size and training time were recorded.

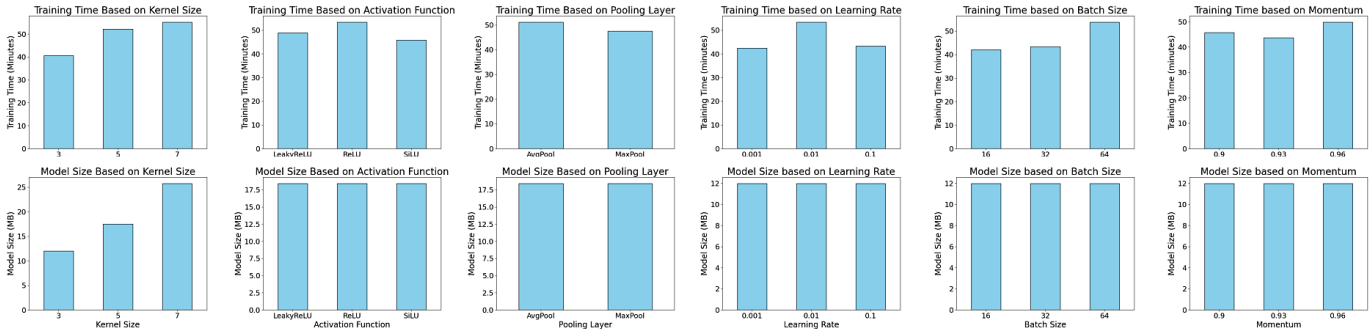
1) First phase modelling: Table VI shows that the use of augmented datasets yields the highest average mAP value of 72.80%, indicating a substantial improvement in model performance. The original dataset also performs reasonably well, achieving an mAP of 55.30%. In comparison, histogram-based preprocessing techniques such as BPDFHE, AVGHEQ, and CLAHE result in similar mAP scores, ranging from 52.89% to 54.44%. While these techniques offer slight improvements over the original dataset, their performance remains lower than that achieved with data augmentation. Notably, the average model size remains consistent across all preprocessing methods, ranging from 11.98 MB to 11.99 MB.

TABLE VI. COMPARISON OF MODEL TRAINING PERFORMANCE RESULTS IN STAGE 1

Num	Model	Average model Size (MB)	Average mAP (%)
1	Using original dataset	11.99	55.30
2	Using augmented dataset	11.99	72.80
3	Using BPDFHE dataset	11.98	52.89
4	Using AVGHEQ dataset	11.98	54.44
5	Using CLAHE dataset	11.98	53.70

2) Second phase modelling: Fig. 8(a) demonstrates that each hyperparameter value influences both the model's training time and size. Among the parameters, kernel size has the most significant impact—larger kernel sizes lead to increased training time and larger model sizes. Additionally, the ReLU activation function results in the longest training time compared to the other activation functions tested. Similarly, average pooling leads to longer training times than max pooling. However, activation functions and pooling types have a relatively minor impact on overall training time and model size. Models trained with image preprocessing techniques tend to produce slightly smaller model sizes, though the difference is not substantial.

Fig. 8(b) indicates that while hyperparameter values considerably affect training time, they do not influence model size. A learning rate of 0.01 results in the longest training time compared to other values. Likewise, a batch size of 64 leads to the longest training time, suggesting that larger batch sizes increase the computational load during training. Additionally, a momentum value of 0.96 yields the longest training time among the tested configurations.



(a) (b)

Fig. 8. Hyperparameter tuning training results on Stage 2.

3) Third phase modelling: Table VII presents the results of the third-stage modeling. The average training time for the model trained with the maximum number of epochs is approximately 20 hours. The model size remains constant at 12.3 MB for both training scenarios. As expected, increasing the number of epochs leads to longer training times; however, model size is influenced primarily by the input data and preprocessing methods rather than the number of epochs.

Notably, the model trained with early stopping using a patience of 300 epochs requires significantly less training time—only 1.8 hours—compared to the full 20 hours for the maximum epoch model. Despite the shorter training duration, the early-stopped model achieves a slightly higher average mAP of 53.37%, compared to 52.26% for the model trained for the maximum number of epochs.

TABLE VII. COMPARISON OF MODEL TRAINING PERFORMANCE RESULTS IN STAGE 3

Num	Model	Training Time (Hours)	Model Size (MB)	Average mAP (%)
1	Maximum *Epoch*	20	12.3	52.26
2	Patience 300	1.8	12.3	53.37

D. Evaluation

1) First phase modelling: Based on the evaluation results presented in Table VIII, the AVGHEQ model demonstrates consistent performance across mAP values, accuracy, and detection time, as indicated by its relatively low standard deviations. This consistency suggests that AVGHEQ is a stable and reliable choice compared to the other models. Conversely, while the BPDFHE models achieve the highest mAP, their accuracy and detection time exhibit greater variability and tend to be lower. The original model, on the other hand, achieves the highest accuracy but requires a longer detection time, which may be a critical factor for applications that prioritize rapid detection response. Therefore, selecting the optimal model requires balancing accuracy, stability of performance, and detection speed, depending on the specific needs of the application.

The model trained with the augmented dataset appears to suffer from overfitting, as evidenced by the average mAP on evaluation data (58.6%) being notably lower than the training mAP (72.80%). This overfitting likely results from the model becoming overly specialized to specific variations present in the augmented training data. Consequently, its performance may degrade when exposed to real-world data with broader variations. Additionally, since YOLOv7 itself incorporates augmentation during training, excessive augmentation in the dataset may reduce the model's adaptability to unseen variations. Despite this, the overall model performance remains acceptable, as the mAP on evaluation data is still reasonably high, though lower than the training results.

TABLE VIII. EVALUATION RESUTLS OF FIRST STAGE MODELLING

Model	mAP (%)		Accuracy (%)		Detection Time (ms)	
	Average	Rank	Average	Rank	Average	Rank
Original model	63.5 ± 1.3	2	84.9 ± 1.4	1	20.0 ± 7.3	5
Augmented model	58.6 ± 1.3	5	82.6 ± 2.1	3	9.8 ± 5.0	1
AVGHEQ model	**63.2 ± 2.4**	**3**	**83.1 ± 1.9**	**2**	**18.6 ± 8.4**	**3**
BPDFHE model	66.3 ± 1.2	1	82.6 ± 3.0	4	18.9 ± 12.2	4
CLAHE model	58.7 ± 3.1	4	80.7 ± 2.4	5	17.7 ± 11.0	2

2) Second phase modelling: After identifying the best preprocessing model, hyperparameter tuning was conducted using the AVGHEQ dataset. All augmentation processes in the default YOLOv7 configuration were disabled by setting the corresponding attributes to zero. To assess the significance of the hyperparameters on model performance, ANOVA analysis was initially performed [43]. Furthermore, MANOVA was employed to examine the simultaneous effects of multiple hyperparameters on the model performance metrics.

Table IX presents the results of the MANOVA analysis using Wilks' lambda criterion. The kernel size and learning rate hyperparameters yielded p-values less than 0.05, indicating statistically significant effects on model performance. In other words, these factors produce significantly different outcomes in the hyperparameter tuning process, allowing us to reject the null hypothesis (H_0). Conversely, the activation function, pooling layer, batch size, and momentum hyperparameters yielded p-values greater than 0.05, suggesting that their variations do not have a statistically significant impact on model performance. Thus, for these hyperparameters, the null hypothesis cannot be rejected.

TABLE IX. MANOVA TEST TO EXAMINE THE EFFECT OF EACH HYPERPARAMETER

Hyperparameter	Value	Num DF	Den DF	F Value	Pr > F
Kernel Size	**0.0137**	**8**	**24**	**22.6388**	**0.0000**
Activation Function	0.4487	8	24	1.4784	0.2168
Pooling Layer	0.8566	4	13	0.5442	0.7063
Learning Rate	**0.1487**	**8**	**42**	**8.3633**	**0.0000**
Batch Size	0.5913	8	42	1.5775	0.1608
Momentum	0.7804	8	44	0.7139	0.6779

3) Third phase modelling: After determining the optimal values for each hyperparameter—kernel size of 3, SiLU activation function, MaxPooling layer, learning rate of 0.01, batch size of 32, and momentum of 0.93—these settings were applied in the third modeling phase. The evaluation results from this phase are summarized in Table X. According to the table, the model trained with the maximum number of epochs achieved the best mAP values and shortest detection time compared to the model trained with patience set to 300. This finding aligns with previous research [44], which demonstrated that increasing the number of epochs generally enhances the performance of deep learning models.

Conversely, the model using patience 300 showed better accuracy compared to the maximum epoch model, although the differences between the two models were marginal. Both models outperform those from earlier modeling stages in terms of mAP, accuracy, and detection time. Moreover, neither model shows signs of overfitting, as indicated by the evaluation mAP being higher than the training mAP [45].

TABLE X. EVALUATION RESULTS OF THIRD STAGE MODELLING

Model	mAP (%)		Accuracy (%)		Detection Time (ms)	
	Average	Rank	Average	Rank	Average	Rank
Maximum epoch	**84.12 ± 7.0**	**1**	**91.19 ± 1.2**	**2**	**4.55 ± 3.3**	**2**
Patience *300*	81.57 ± 4.1	2	92.23 ± 2.4	1	5.03 ± 3.9	1

E. Comparison Between Our Results and Previous Research

According to the results shown in Table X, which presents only the model trained with the maximum number of epochs, the proposed model outperforms previous approaches. For example, the Faster R-CNN model achieved an mAP of 48.85%, the SSD model reached 33.16%, and the improved YOLOv3 model only 16.56%, as illustrated in Fig. 9. While the histogram-based image processing method does not improve accuracy—likely because it converts images by focusing on specific blocks rather than the entire image—it does contribute to increased inference speed. Enhancing the original YOLOv7 architecture also plays a critical role in boosting detection performance.

These findings indicate that models initialized with augmented data generally perform better than those without augmentation. Additionally, increasing the number of training epochs improves model robustness and yields the best detection results. Ultimately, an effective network balances having a relatively low number of parameters while efficiently extracting object features, thereby improving accuracy and inference speed simultaneously.

Furthermore, these results emphasize the importance of optimizing both the network architecture and the training strategy to achieve superior object detection performance. Although challenging, future improvements may involve fine-tuning the number of training epochs and applying targeted data augmentation techniques. Such enhancements will strengthen real-time inference capabilities and pave the way for further advances in model architectures and optimization strategies within deep learning.

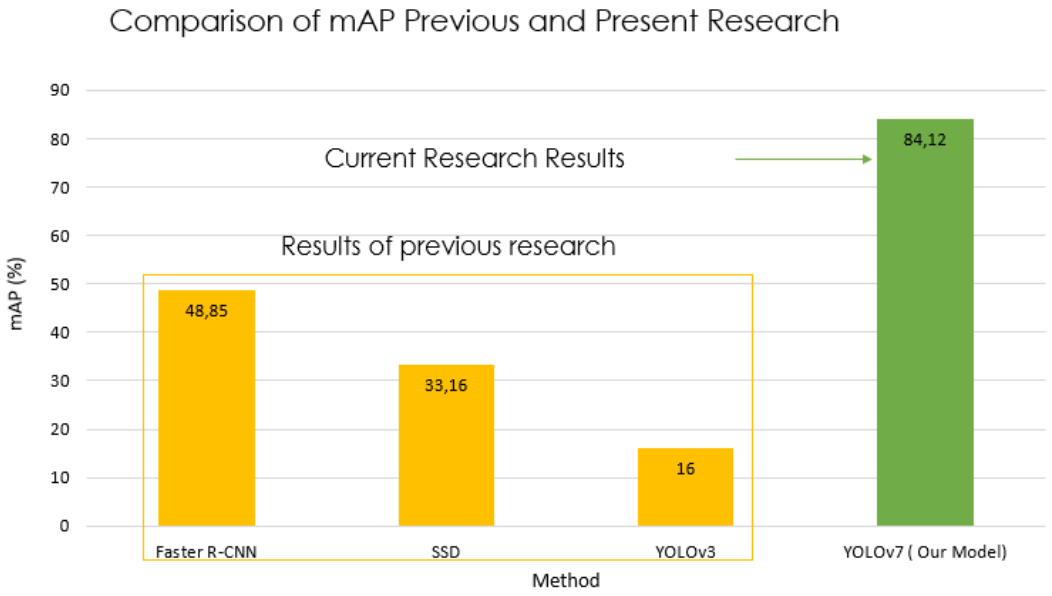


Fig. 9. mAP comparation between previous and our research.

V. CONCLUSION

This study analyzed a model for detecting deviations in melon leaf objects using several datasets: the original dataset, augmented datasets, and datasets preprocessed with AVGHEQ, BPDFHE, and CLAHE techniques. The analysis showed that the model's average mAP ranged from 58.6% to 66.3%, accuracy ranged from 80.7% to 84.9%, and detection time varied between 9.8 and 20 milliseconds. Image preprocessing improved the original model's performance in terms of mAP (particularly with BPDFHE) and detection time (across all preprocessing methods), but did not enhance accuracy. The BPDFHE model achieved the highest mAP value of 84.9%, while the fastest detection time of 9.4 milliseconds was observed with the model trained on augmented datasets. Overall, models trained on AVGHEQ-preprocessed datasets showed more balanced and stable results across all three variables—mAP (63.2%), accuracy (83.1%), and detection time (18.6 milliseconds)—compared to other models that performed well in only one or two metrics.

Hyperparameter tuning with the AVGHEQ dataset using the YOLOv7 algorithm revealed that kernel size and learning rate significantly impacted model performance. Specifically, a kernel size of 3 outperformed sizes 5 and 7, and a learning rate of 0.001 was superior to 0.1 and 0.01. Other hyperparameters, including activation functions, pooling layers, batch sizes, and momentum, showed no statistically significant effect on performance.

The best-performing models were obtained using the maximum number of training epochs combined with YOLOv7's default augmentation. Increasing the number of epochs contributed to more robust models and improved detection performance. However, excessive augmentation can potentially distort or obscure the original data patterns, making it harder for the model to generalize effectively.

Based on these findings, two recommendations are proposed for future research: first, to improve model accuracy by customizing network architectures, exploring more diverse preprocessing techniques, and expanding the hyperparameter tuning range; and second, to carefully balance augmentation to preserve critical data features. The model trained with maximum epochs achieved an average mAP of 84.12%, accuracy of 91.19%, and detection time of 4.55 milliseconds. In comparison, the model trained with patience set to 300 epochs achieved an average mAP of 81.57%, accuracy of 92.23%, and detection time of 5.03 milliseconds. These results suggest that increasing the number of epochs enhances model robustness and overall performance.

ACKNOWLEDGMENT

The computation in this study were conducted using the HPC facilities of Computer Science Study Program, School of Data Science, Mathematics and Informatics, IPB University. The dataset used in this research was obtained from [36], acquired from the IoT for Smart Urban Farming Laboratory (iSurf Lab), School of Data Science, Mathematics and Informatics, IPB University, and Agribusiness and Technology Park (ATP), IPB University.

REFERENCES

[1] S. I. Kusumaningrum, "Pemanfaatan Sektor Pertanian Sebagai Penunjang Pertumbuhan Perekonomian Indonesia," J. Transaksi, vol. 11, no. 1, pp. 80–89, 2019, [Online]. Available: http://ejournal.atmajaya.ac.id/index.php/transaksi/article/view/477

[2] G. Afriyanti, Ana Mariya, Charita Natalia, Sirat Nispuana, M. Farhan Wijaya, and M. Yoga Phalepi, "the Role of the Agricultural Sector on Economic Growth in Indonesia," Indones. J. Multidiscip. Sci., vol. 2, no. 1, pp. 167–179, 2023, doi: 10.59066/ijoms.v2i1.325.

[3] K. F. Arifah and J. Kim, "The Importance of Agricultural Export Performance on the Economic Growth of Indonesia: The Impact of the COVID-19 Pandemic," Sustain., vol. 14, no. 24, 2022, doi: 10.3390/su142416534.

[4] BPS, "Indonesian Fruit Plant Production from 2016 to 2022," 2022. [Online]. Available: https://www.bps.go.id/indicator/55/62/1/produksi-tanaman-buah-buahan.html

[5] S. R. Siregar, E. Hayati, and M. Hayati, "Respon Pertumbuhan dan Produksi Melon (Cucumis melo L.) Akibat Pemangkasan dan Pengaturan Jumlah Buah," J. Ilm. Mhs. Pertan., vol. 4, no. 1, pp. 202–209, 2020, doi: 10.17969/jimfp.v4i1.6419.

[6] O. S. University, "Environmental factors affecting plant growth," 2024.

[7] S. A. Avazovich, "Diseases of Melons: Reasons, Symptoms, and Methods of Control," Int. J. Life Sci. Agric. Res., vol. 01, no. 01, pp. 11–12, 2022.

[8] D. A. Fitria and M. I. Riyadi, "Strategi Coping Stres Pada Petani Melon Pasca Gagal Panen di Desa Maguwan, Kecamatan Sambit, Kabupaten Ponorogo," Rosyada Islam. Guid. Couns., vol. 3, no. 1, p. 51, 2022.

[9] A. Khakimov, I. Salakhutdinov, A. Omolikov, and S. Utaganov, "Traditional and current-prospective methods of agricultural plant diseases detection: A review," IOP Conf. Ser. Earth Environ. Sci., vol. 951, no. 1, 2022, doi: 10.1088/1755-1315/951/1/012002.

[10] L. Munk, D. B. Collinge, and A. M. Tronsmo, "Diagnosis of Plant Diseases," in Plant Pathology and Plant Diseases, 2020, pp. 164–181.

[11] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," Comput. Electron. Agric., vol. 147, no. July 2017, pp. 70–90, 2018, doi: 10.1016/j.compag.2018.02.016.

[12] C. S. R. Silva and J. M. Fonseca, Artificial Intelligence and Algorithms in Intelligent Systems, vol. 2. 2019. doi: 10.1007/978-3-319-91189-2_30.

[13] I. Goodfellow, Y. Bengio, and • Aaron Courville, "Deep Learning," Foreign Aff., vol. 91, no. 5, pp. 1689–1699, 2016.

[14] C. M. Bishop, Neural Network for Pattern Recognition. Birmingham: CLARENDON PRESS, 1995. doi: 10.1109/RusAutoCon49822.2020.9208207.

[15] B. Mehlig, "Machine Learning with Neural Networks," Mach. Learn. with Neural Networks, 2021, doi: 10.1017/9781108860604.

[16] M. Kubat, An Introduction to Machine Learning. 2017. doi: 10.1007/978-3-319-63913-0.

[17] F. Chollet, Deep Learning with Python. New York: Manning Publications Co., 2018. doi: 10.23919/ICIF.2018.8455530.

[18] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object Detection in 20 Years: A Survey," pp. 1–39, 2019, [Online]. Available: http://arxiv.org/abs/1905.05055

[19] 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.

[20] Z. Song, S. Yang, and R. Zhang, "Does Preprocessing Help Training Over-parameterized Neural Networks?," Adv. Neural Inf. Process. Syst., vol. 27, pp. 22890–22904, 2021.

[21] N. A. Simanjuntak, J. Hendarto, and Wahyono, "The effect of image preprocessing techniques on convolutional neural network-based human action recognition," J. Theor. Appl. Inf. Technol., vol. 98, no. 16, pp. 3364–3374, 2020.

[22] M. Maryani, R. L. Prabawani, and B. S. Daryono, "Struktur Anatomi Epidermis Daun Lima Kultivar Melon (Cucumis melo L.) Berdasarkan Resistensinya terhadap Jamur Tepung (Sphaerotheca fuliginea Poll)," Biota J. Ilm. Ilmu-Ilmu Hayati, vol. 14, no. 2, pp. 105–114, 2010, doi: 10.24002/biota.v14i2.2688.

[23] G. B. Ramsey and M. A. Smith, Market Diseases of Cabbage, Cauliflower, Turnips, Cucumbers, Melons, and Related Crops, no. 184. 1961.

[24] R. C. Gonzalez and R. E. Woods, Digital Image Processing, 4th ed. New York: Pearson, 2018.

[25] P. K. Sinha, Image acquisition and preprocessing for machine vision systems. 2012. doi: 10.1117/3.858360.

[26] William K Pratt, Digital Image Processing, IV., vol. 13, no. 1. Canada: John Wiley & Sons, Inc, 2007.

[27] S. C. F. Lin et al., "Image enhancement using the averaging histogram equalization (AVHEQ) approach for contrast improvement and brightness preservation," Comput. Electr. Eng., vol. 46, pp. 356–370, 2015, doi: 10.1016/j.compeleceng.2015.06.001.

[28] W. A. Mustafa and M. M. M. A. Kader, "A Review of Histogram Equalization Techniques in Image Enhancement Application," J. Phys. Conf. Ser., vol. 1019, no. 1, 2018, doi: 10.1088/1742-6596/1019/1/012026.

[29] D. Sheet, H. Garud, A. Suveer, M. Mahadevappa, and J. Chatterjee, "Brightness preserving dynamic fuzzy histogram equalization," IEEE Trans. Consum. Electron., vol. 56, no. 4, pp. 2475–2480, 2010, doi: 10.1109/TCE.2010.5681130.

[30] 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," pp. 1–15, 2022, [Online]. Available: http://arxiv.org/abs/2207.02696

[31] Q. Wang, F. Qi, M. Sun, J. Qu, and J. Xue, "Identification of Tomato Disease Types and Detection of Infected Areas Based on Deep Convolutional Neural Networks and Object Detection Techniques," Comput. Intell. Neurosci., vol. 2019, 2019, doi: 10.1155/2019/9142753.

[32] J. Liu and X. Wang, "Tomato Diseases and Pests Detection Based on Improved Yolo V3 Convolutional Neural Network," Front. Plant Sci., vol. 11, no. June, pp. 1–12, 2020, doi: 10.3389/fpls.2020.00898.

[33] K. Zhang, Q. Wu, and Y. Chen, "Detecting soybean leaf disease from synthetic image using multi-feature fusion faster R-CNN," Comput. Electron. Agric., vol. 183, no. February, p. 106064, 2021, doi: 10.1016/j.compag.2021.106064.

[34] S. Li, K. Li, Y. Qiao, and L. Zhang, "A multi-scale cucumber disease detection method in natural scenes based on YOLOv5," Comput. Electron. Agric., vol. 202, no. 17, p. 107363, 2022, doi: 10.1016/j.compag.2022.107363.

[35] Y. Wang, Y. Wang, and J. Zhao, "MGA-YOLO: A lightweight one-stage network for apple leaf disease detection," Front. Plant Sci., vol. 13, 2022, doi: 10.3389/fpls.2022.927424.

[36] H. Rahmat, S. Wahjuni, and H. Rahmawan, "Performance Analysis of Deep Learning-based Object Detectors on Raspberry Pi for Detecting Melon Leaf Abnormality," Int. J. Adv. Sci. Eng. Inf. Technol., vol. 12, no. 2, pp. 572–579, 2022, doi: 10.18517/ijaseit.12.2.13801.

[37] S. Shome and S. Vadali, "Enhancement of diabetic retinopathy imagery using contrast limited adaptive histogram equalization," Int. J. Comput. Sci. Inf. Technol., vol. 2, no. 6, pp. 2694–2699, 2011.

[38] S. R. Saufi, Z. A. Bin Ahmad, M. S. Leong, and M. H. Lim, "Challenges and opportunities of deep learning models for machinery fault detection and diagnosis: A review," IEEE Access, vol. 7, pp. 122644–122662, 2019, doi: 10.1109/ACCESS.2019.2938227.

[39] D. Berrar, "Cross-validation," Encycl. Bioinforma. Comput. Biol. ABC Bioinforma., vol. 1–3, no. January 2018, pp. 542–545, 2018, doi: 10.1016/B978-0-12-809633-8.20349-X.

[40] E. M., V.-G. L., W. C. K. I., W. J., and Z. A., "The Pascal Visual Object Classes (VOC) Challenge," Int. J. Comput. Vis., vol. 88, no. 2, pp. 303–338, 2010.

[41] A. Anwar, "What is Average Precision in Object Detection & Localization Algorithms and how to calculate it?," Towards Data Science, 2022. https://towardsdatascience.com/what-is-average-precision-in-object-detection-localization-algorithms-and-how-to-calculate-it-3f330efe697b

[42] A. Salazar-Gomez, M. Darbyshire, J. Gao, E. I. Sklar, and S. Parsons, "Beyond mAP: Towards practical object detection for weed spraying in precision agriculture," IEEE Int. Conf. Intell. Robot. Syst., vol. 2022-Octob, pp. 9232–9238, 2022, doi: 10.1109/IROS47612.2022.9982139.

[43] J. N. Van Rijn and F. Hutter, "Hyperparameter importance across datasets," Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min., pp. 2367–2376, 2018, doi: 10.1145/3219819.3220058.

[44] O. G. Ajayi and J. Ashi, "Effect of varying training epochs of a Faster Region-Based Convolutional Neural Network on the Accuracy of an Automatic Weed Classification Scheme," Smart Agric. Technol., vol. 3, no. August 2022, p. 100128, 2023, doi: 10.1016/j.atech.2022.100128.

[45] X. Ying, "An Overview of Overfitting and its Solutions," J. Phys. Conf. Ser., vol. 1168, no. 2, 2019, doi: 10.1088/1742-6596/1168/2/022022.