

Comparative Analysis of CNN and YOLOv11 Algorithms for Detecting Oyster Mushroom Maturity and Harvest Readiness

Juhar Riski Ahmadi ^{1*}, Nurdin ^{2**}, Mukti Qamal ^{3*}

^{*} Departement of Informatics, Universitas Malikussaleh, Lhokseumawe, Indonesia

^{**} Departement of Information Technology, Universitas Malikussaleh, Lhokseumawe, Indonesia
nurdin@unimal.ac.id ²

Article Info

Article history:

Received 2026-08-09

Revised 2026-09-11

Accepted 2026-10-09

Keyword:

CNN,
ResNet50,
YOLOv11,
Oyster Mushroom,
Harvest Readiness,
Deep Learning,
Image Classification,
Object Detection.

ABSTRACT

Determining the optimal harvest time for oyster mushrooms (*Pleurotus* sp.) is a critical challenge in cultivation practice, as conventional methods rely on subjective visual inspection by farmers, which is prone to human error and inconsistency. This study presents a comparative analysis of two deep learning approaches, namely a Convolutional Neural Network (CNN) based on the ResNet50 architecture and YOLOv11, for classifying oyster mushroom maturity and harvest readiness into two categories: "harvest-ready" and "not harvest-ready." A dataset of 1,200 balanced images was collected directly from an oyster mushroom cultivation facility and preprocessed through auto-orientation, resizing to 640×640 pixels, and data augmentation applied exclusively to the training set. Model performance was evaluated using accuracy, precision, recall, F1-score, and training time as evaluation metrics. The comparative evaluation was conducted on 30 newly acquired images that were entirely excluded from the training, validation, and testing datasets. The experimental results show that YOLOv11s achieved superior and more balanced performance, with accuracy, precision, recall, and F1-score each reaching 93%. In comparison, the CNN (ResNet50) model achieved an accuracy of 90%, a precision of 100%, a recall of 83.3%, and an F1-score of 90.9%. In terms of training efficiency, CNN (ResNet50) required 71.9 minutes, while YOLOv11s required 89.3 minutes. These findings indicate that YOLOv11s is a more effective model for automated oyster mushroom harvest readiness detection and has the potential to be integrated into smart farming systems.



This is an open access article under the [CC-BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

I. INTRODUCTION

Oyster mushroom (*Pleurotus* sp) is an agricultural commodity with high economic value and serves as an important source of nutritious food. However, determining the optimal harvesting time remains a significant challenge for farmers because it is generally performed manually, which may lead to suboptimal harvesting decisions and consequently affect the quality of the harvested mushrooms. Detecting mushroom maturity and harvest readiness is a critical aspect of oyster mushroom cultivation, as it directly influences harvest quality, market value, and production efficiency. Traditionally, harvesting decisions rely on farmers' visual observations, which are inherently subjective and prone to errors due to their dependence on individual

experience. Furthermore, this manual approach is labor-intensive and time-consuming, particularly in large-scale cultivation systems [1].

Recent advances in deep learning-based computer vision have created significant opportunities for developing automated solutions for oyster mushroom maturity and harvest readiness detection. In recent years, deep learning methods, particularly Convolutional Neural Networks (CNNs), have become widely adopted for agricultural image classification and detection tasks, including disease identification and oyster mushroom classification. Recent studies have demonstrated that CNN-based models can achieve an accuracy of up to 97.14% in classifying diseases affecting oyster mushroom cultivation bags (baglogs) by leveraging transfer learning techniques, which accelerate the

training process while improving classification performance [2]. In another study, a CNN model achieved an accuracy of 82% for mushroom genus classification using a locally collected dataset [3].

In addition to CNN-based approaches, one-stage object detection models such as You Only Look Once (YOLO) have demonstrated superior inference speed and real-time detection capabilities, particularly for detecting small objects in complex real-world environments. An international study reported that the YOLOv5s model achieved a high mean Average Precision (mAP) while maintaining satisfactory inference speed for detecting *Agaricus bisporus* mushrooms [4]. Furthermore, a recent study published in 2024 demonstrated that a lightweight YOLO variant specifically optimized for oyster mushroom detection achieved an mAP of 94.95% while requiring fewer model parameters and shorter inference time than conventional YOLO architectures [5].

Moreover, the latest developments in the YOLO family continue to improve the trade-off between detection accuracy and inference speed for agricultural applications. However, studies specifically investigating the application of YOLOv11 for oyster mushroom maturity and harvest readiness detection remain limited. Given its promising capabilities, YOLOv11 has considerable potential for integration into smart farming systems alongside Internet of Things (IoT) technologies, enabling automated harvesting processes that improve production efficiency and harvest quality.

Therefore, a comparative analysis of CNN and YOLOv11 for detecting oyster mushroom maturity and harvest readiness is relevant and important to conduct, particularly due to the limited number of deep learning studies focusing on oyster mushrooms as an agricultural commodity cultivated by farmers. This research aims to address the existing research gap by evaluating the performance of these two deep learning approaches and identifying the most suitable algorithm for supporting intelligent, efficient, and technology-driven oyster mushroom cultivation. The comparative analysis in this study is conducted by evaluating the performance of CNN and YOLOv11 using accuracy, precision, recall, F1-score, and model training time as the primary evaluation metrics.

II. METHODOLOGY

The research methodology consisted of a series of interconnected stages, starting from data collection and preprocessing to model evaluation and performance comparison. Two deep learning models, namely a Convolutional Neural Network (CNN) with the ResNet50 architecture and YOLOv11, were employed to analyze oyster mushroom image data. Each stage was carefully designed to ensure high quality data preparation and optimal model development. After both models were trained, their performance was evaluated and compared to identify the model that achieved the best performance in detecting oyster mushroom maturity and harvest readiness.

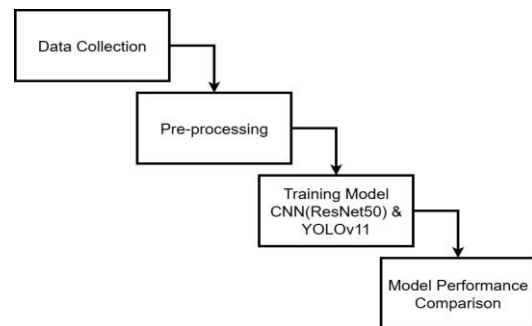


Figure 1. Research Stages

A. Data Collection

Data collection represents the initial stage of this study and aims to acquire the dataset used for training and evaluating the deep learning models. In this study, the dataset was collected by capturing images of oyster mushrooms directly using a smartphone camera. The image acquisition process was conducted in an oyster mushroom cultivation facility under various conditions, including different camera angles, object to camera distances, and lighting conditions. These variations were intentionally introduced to ensure that the resulting dataset accurately represents real-world cultivation environments. All captured images were subsequently reviewed to ensure image quality and were categorized into predefined classes.

The dataset consisted of 1,200 images divided into two classes: harvest-ready and not harvest-ready. Class determination was based on several visual characteristics of the oyster mushrooms, including cap size, color, and the density of mushrooms within a single cluster. Each class contained 600 images, resulting in a balanced dataset. Maintaining an equal number of images for each class helps reduce class imbalance during the training process, allowing the models to learn representative features from both categories more effectively. The collected dataset was subsequently used in the preprocessing stage before being employed to train the CNN (ResNet50) and YOLOv11 models. The following shows the distribution of the dataset.

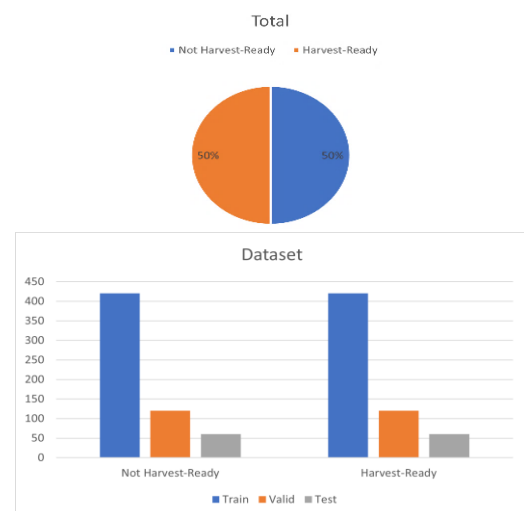


Figure 2. Distribution Dataset

B. Preprocessing

Data preprocessing is an important stage performed before the model training process to prepare the dataset for subsequent analysis. The primary objective of this stage is to produce a standardized and high quality dataset that meets the input requirements of both the CNN (ResNet50) and YOLOv11 models. In this study, all images first underwent an auto-orientation process, in which the image orientation was automatically corrected based on the metadata embedded in the images to ensure consistent image alignment. Subsequently, all images were resized to 640×640 pixels using the stretch resizing method. This method resizes each image to a predetermined resolution without preserving the original aspect ratio, ensuring that all images have input dimensions that are compatible with the requirements of the deep learning models. The 640×640 pixel resolution was selected because it is the standard default input size commonly used by the YOLO model. Therefore, the same 640×640 pixel resolution was also applied to the CNN model to maintain consistency and ensure that the training data used for both models were balanced and comparable.

After the resizing process was completed, the dataset was divided into three subsets: 70% for training, 20% for validation, and 10% for testing. This data partitioning allows the models to be trained using the training set, validated during the training process to monitor their generalization capability, and subsequently evaluated using the testing set, which contains data that have not been previously used by the models. The same data partitioning strategy was applied to both models, including the number and specific images used in each subset, to ensure that both models were trained, validated, and tested using the same dataset and to minimize the potential for unwanted bias. Thus, the evaluation results can provide a more objective and fair assessment of the performance of both models.

After the preprocessing stage was completed, data augmentation was applied exclusively to the training set to increase dataset diversity without requiring additional image acquisition. The augmentation process generated various versions of the original images while preserving the essential characteristics of the oyster mushrooms, allowing the models to learn more robust and representative features under different environmental conditions. At this stage, several augmentation techniques were applied to both models, including flip, rotation, brightness, exposure, blur, and noise. Applying augmentation only to the training set can improve the models' generalization capability, reduce the risk of overfitting, and enhance their robustness when encountering variations in real-world conditions. In contrast, no augmentation was applied to the validation or testing sets, ensuring that model evaluation was performed using unaltered data and could more accurately reflect the models' generalization performance. After the preprocessing stage was completed, *data augmentation* was applied exclusively to the training set to increase dataset diversity without requiring additional image acquisition. The augmentation process generated various versions of the original images while

preserving the essential characteristics of the oyster mushrooms, allowing the models to learn more robust and representative features under different environmental conditions. At this stage, several augmentation techniques were applied to both models, including *flip*, *rotation*, *brightness*, *exposure*, *blur*, and *noise*. Applying augmentation only to the training set can improve the models' generalization capability, reduce the risk of *overfitting*, and enhance their robustness when encountering variations in real-world conditions. In contrast, no augmentation was applied to the validation or testing sets, ensuring that model evaluation was performed using unaltered data and could more accurately reflect the models' generalization performance.

C. Convolutional Neural Network (ResNet50)

Convolutional Neural Network (CNN) are among the most widely used deep learning methods for image classification tasks. Although CNNs share the same fundamental principles as conventional artificial neural networks, they employ a specialized architecture consisting of two main components: the Feature Extraction Layer and the Fully Connected Layer (Multilayer Perceptron, MLP) [6]. The feature extraction layer automatically learns hierarchical representations from input images through convolution and pooling operations, while the fully connected layer performs the final classification based on the extracted features. CNNs process image data by analyzing the spatial arrangement of pixels organized in a two-dimensional grid, where each pixel contains intensity values representing brightness and color. This enables the network to effectively learn complex visual patterns and semantic features from the input images [7] [8].

In this study, the CNN model was implemented using the ResNet50 architecture. ResNet50 is a deep convolutional neural network consisting of 50 layers and was specifically designed to address the vanishing gradient problem commonly encountered in very deep neural networks through the use of residual connections. The architecture comprises 49 convolutional layers followed by a fully connected layer for classification. By incorporating identity shortcut connections, ResNet50 provides an alternative pathway for gradient propagation during backpropagation, thereby facilitating more stable and efficient training. This residual learning strategy enables the network to maintain high classification performance while achieving an effective balance between network depth and computational efficiency [9].

The key feature that distinguishes ResNet from conventional CNN architectures is the incorporation of skip connections (also known as identity shortcut connections) between network layers. These connections bypass one or more intermediate layers, allowing information and gradients to flow directly across the network. As a result, important feature representations are preserved, and gradient propagation during backpropagation remains stable, effectively mitigating the vanishing gradient problem. This design enables the successful training of substantially deeper neural networks while maintaining high learning efficiency and improving overall model performance [10].

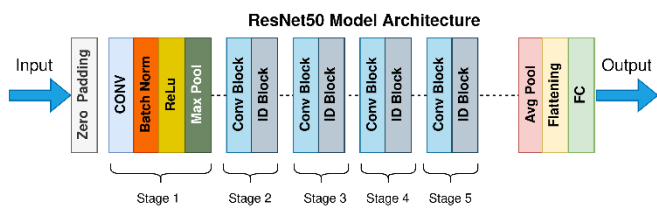


Figure 3. Architecture ResNet50

D. You Only Look Once (YOLOv11)

You Only Look Once (YOLO) is a deep learning-based object detection algorithm designed for real-time applications. YOLO adopts a single-stage detection approach by formulating object detection as a unified regression problem, in which a single neural network simultaneously predicts object bounding boxes and class probabilities directly from the entire input image in a single forward pass. Unlike two-stage object detectors, which rely on a separate region proposal generation process followed by object classification using more complex architectures, single-stage detectors such as YOLO perform object localization and classification in a single step across the entire image. This unified framework significantly improves detection speed while maintaining competitive detection accuracy, making YOLO substantially faster than conventional approaches such as R-CNN and Faster R-CNN. Since its initial introduction, the YOLO family has undergone continuous architectural improvements, resulting in successive versions with enhanced accuracy, computational efficiency, and detection performance. These advancements have culminated in the latest release, YOLOv11, which is employed in this study [11] [12].

The YOLO detection process begins by dividing the input image into a grid of cells, where each cell is responsible for predicting multiple bounding boxes along with the probability of object presence. Each predicted bounding box is assigned a confidence score, which reflects the likelihood that an object is present as well as the accuracy of its predicted location. The model also predicts class probabilities for each bounding box to determine the object's category. Once predictions from all grid cells have been generated, Non-Maximum Suppression (NMS) is applied to eliminate redundant overlapping bounding boxes while retaining the most confident detections. This one-stage detection pipeline enables YOLO to achieve efficient and accurate object detection with low inference latency [13] [14].

Since its introduction, the YOLO family has undergone substantial architectural advancements, evolving from YOLOv1 to the latest YOLOv11. Each successive version has introduced improvements such as more powerful backbone networks and advanced data augmentation techniques, including Mosaic augmentation and adaptive anchor matching, which enhance detection accuracy for objects with varying sizes, scales, and spatial locations. In particular, YOLOv11 incorporates multi-scale attention modules and an adaptive detection head, enabling the model to better capture

complex object features while improving recall and mean Average Precision (mAP). Although these architectural enhancements substantially improve detection performance, they also increase computational requirements, making hardware capability an important consideration for practical deployment [15].

YOLOv11 is the most recent generation of the YOLO object detection framework, officially introduced in 2024. Compared with earlier versions, YOLOv11 incorporates several architectural enhancements, including improved multi-scale feature learning, a refined anchor mechanism, and a more computationally efficient backbone network. These improvements enable the model to achieve higher detection accuracy while maintaining real-time inference capability. The progression of the YOLO family and the performance improvements achieved across different versions are illustrated in Figure 3.

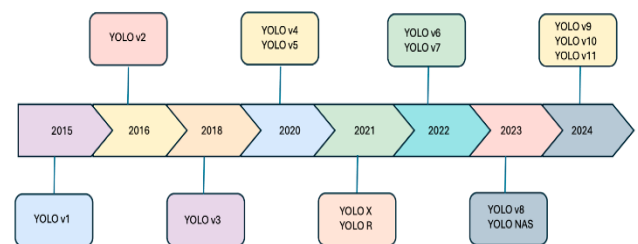


Figure 4. YOLO Evolution and model Performance from 2015-2024

The development of the YOLO framework from its initial release to the latest version can be summarized as follows:

1. YOLOv1 (2015): Introduced the first single-stage object detection framework capable of detecting objects by processing an entire image in a single forward pass [16].
2. YOLOv2 (2016): Enhanced detection performance through the introduction of anchor boxes and upsampling techniques, resulting in improved localization accuracy [17].
3. YOLOv3 (2018): Added Incorporated a more powerful backbone network, multi-scale feature extraction, and multiple anchor boxes to improve detection of objects at different scales [18].
4. YOLOv4 (2020): Improved both training efficiency and detection accuracy by integrating Mosaic data augmentation and several architectural optimizations [19].
5. YOLOv5 (2020): Released by Ultralytics as a Python-based implementation emphasizing flexibility, ease of use, and practical deployment [20].
6. YOLOv6 (2022): Designed as a lightweight, open-source object detection model optimized for edge computing applications [21].
7. YOLOv7 (2022): Introduced additional architectural refinements to achieve improved detection accuracy while maintaining computational efficiency [22].
8. YOLOv8 (2023): Further refined the detection architecture by optimizing the backbone, neck, and head

components, together with several new modules that enhanced both accuracy and inference efficiency [23].

9. YOLOv9 (2024): Introduced the Programmable Gradient Information (PGI) and Generalized Efficient Layer Aggregation Network (GELAN) techniques to improve feature representation in deep neural networks [24].
10. YOLOv10 (2024): Developed by Tsinghua University with an emphasis on computational efficiency through the adoption of large-kernel convolutions and partial self-attention mechanisms [24].
11. YOLOv11 (2024): Represents the latest evolution of the YOLO family, incorporating advanced architectural optimizations that enhance both detection performance and computational efficiency for increasingly complex computer vision tasks[25].

E. Model Performance Comparison

Model performance comparison represents the final stage of this study and aims to evaluate and compare the performance of the CNN (ResNet50) and YOLOv11 models in classifying the maturity level of oyster mushrooms. The comparison was conducted using 30 newly acquired images that were excluded from the training, validation, and testing datasets. This evaluation strategy was adopted to assess the ability of each model to generalize to previously unseen data under real-world conditions.

The performance of both models was evaluated based on the confusion matrix generated during the testing process. Several evaluation metrics were derived from the confusion matrix, including accuracy, which measures the overall correctness of the model's predictions; precision, which evaluates the proportion of correctly predicted instances among all positive predictions; recall, which measures the model's ability to correctly identify all instances belonging to each class; and the F1-score, which represents the harmonic mean of precision and recall. In addition to these classification metrics, the training time required by each model was also analyzed as an indicator of computational efficiency. The results obtained from these evaluation metrics were subsequently compared to determine the model that demonstrated the best overall performance in classifying oyster mushrooms into the harvest-ready and not harvest-ready categories.

The following evaluation metrics were used to compare the performance of the CNN (ResNet50) and YOLOv11 models:

- a. Accuracy represents the overall proportion of correctly classified instances, including both positive and negative samples, relative to the total number of observations.

$$Accuracy = \frac{TP+TN}{T+TN+FP+FN} \quad (1)$$

- b. Recall measures the ability of a classification model to correctly identify all relevant instances within a dataset.

It represents the proportion of actual positive samples that are correctly classified by the model.

$$Recall = \frac{TP}{TP+FN} \quad (2)$$

- c. Precision evaluates the reliability of the model's positive predictions by measuring the proportion of correctly predicted positive instances among all instances predicted as positive.

$$Precision = \frac{TP}{TP+FP} \quad (3)$$

- d. The F1-score is a performance metric that combines precision and recall into a single measure by calculating their harmonic mean. It is particularly useful when a balanced assessment of both metrics is required.

$$F1-Score = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall} \quad (4)$$

TABLE I.
CONFUSION MATRIX

Confusion Matrix		Predicted	
		Positive	Negative
Actual	Positive	TP (True Positive)	FN(False Negative)
	Negative	FP (False Positive)	TN (True Negative)

This table presents the confusion matrix used to evaluate the performance of the detection models. The matrix serves as the basis for calculating key evaluation metrics, including precision, recall, F1-score, and accuracy, by comparing the actual class labels with the predicted outcomes.

III. RESULT AND DISCUSSION

Classifying the maturity level of oyster mushrooms is a critical stage in the cultivation process, as it determines the optimal harvesting time and directly influences harvest quality. Conventional maturity assessment relies heavily on farmers' experience and visual inspection, making the decision-making process subjective and prone to human error. To address this limitation, deep learning approaches, including a Convolutional Neural Network (CNN) based on the ResNet50 architecture and YOLOv11, were employed to automate the classification of oyster mushroom maturity.

This study presents a comparative performance analysis of the two models to identify the most effective approach for classifying oyster mushrooms into "harvest-ready" and "not harvest-ready" categories. The comparison was conducted using several evaluation metrics, including accuracy, precision, recall, F1-score, and training time, providing a comprehensive assessment of the strengths and limitations of each model.

A. Data Description

In this study, the dataset was collected by capturing images of oyster mushrooms directly at a cultivation facility using a smartphone camera. Image acquisition was performed

under various conditions, including different camera angles, object-to-camera distances, and lighting conditions, to ensure that the dataset accurately represented real-world cultivation environments.



Figure 5. Research Dataset

The dataset consisted of 1,200 images, evenly distributed into two classes: “harvest-ready” and “not harvest-ready”, with 600 images in each class. Maintaining a balanced number of samples for each category helps minimize class imbalance during model training, allowing the models to learn representative features from both classes more effectively. Before model development, all images underwent a preprocessing pipeline consisting of auto-orientation, resizing to 640×640 pixels, dataset partitioning into 70% training, 20% validation, and 10% testing subsets, followed by data augmentation applied exclusively to the training set to increase dataset diversity.

In addition to the primary dataset used for model training and evaluation, this study employed 30 newly acquired images that were excluded from the training, validation, and testing datasets. These images were used during the model performance comparison stage to evaluate the generalization capability of the CNN (ResNet50) and YOLOv11 models on previously unseen data. Consequently, the evaluation results provide a more objective assessment of the ability of both models to classify oyster mushrooms into the “harvest-ready” and “not harvest-ready” categories.

B. Model Configuration and Training

Before the training process, the CNN (ResNet50) and YOLOv11 models were prepared using the same training configuration to ensure a fair and consistent comparison. Both models were trained using the AdamW optimizer with a maximum of 100 epochs and an early stopping patience of 15 epochs. The same training dataset, validation dataset, image resolution, and augmentation strategy were also applied to both models. By maintaining consistent training configurations and input data across the two models, differences arising from variations in the training setup could be minimized. Therefore, the performance differences observed during the evaluation were expected to primarily

reflect the characteristics and capabilities of the respective model architectures rather than differences in the experimental configuration. This approach was adopted to improve the fairness, consistency, and objectivity of the comparative evaluation between CNN (ResNet50) and YOLOv11.

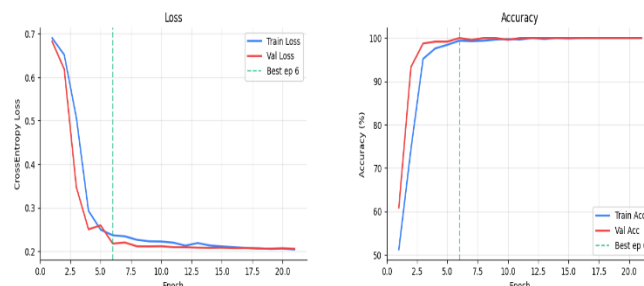


Figure 6. Training Curves Model ResNet50

Based on the accuracy and loss curves, the CNN (ResNet50) model demonstrates a well performing learning process. During the initial epochs, both training and validation accuracy increased rapidly. The training accuracy increased from approximately 51% in the first epoch to nearly 100%, while the validation accuracy increased from approximately 61% to around 99–100%. After approximately epoch 6, both accuracy values remained relatively stable at a high level until the end of training. Meanwhile, the training loss and validation loss decreased significantly from approximately 0.69 at the beginning of training to around 0.20 toward the end. After epoch 6, both loss values remained relatively stable with only a small difference between them. The training process stopped at epoch 21 due to the implementation of early stopping with a patience value of 15 epochs. After achieving the best validation performance at epoch 6, the model did not show any significant improvement in validation performance for the following 15 epochs. Therefore, the training process was automatically terminated at epoch 21 to prevent unnecessary training and reduce the risk of overfitting. Overall, these results indicate that the model was able to learn the characteristics of the dataset effectively and maintain consistent performance on both the training and validation data.

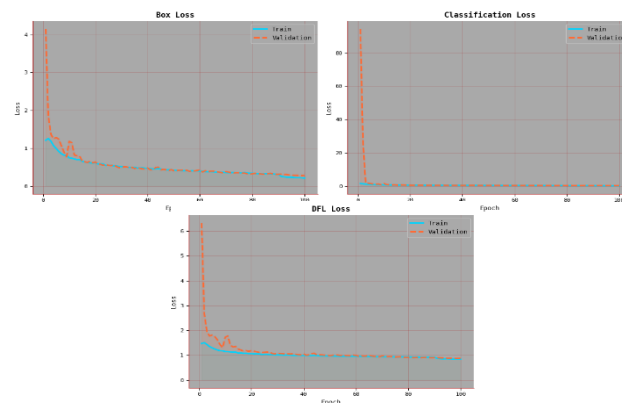


Figure 7. Training Curves Model ResNet50

Based on the YOLOv11 training results, the Box Loss, Classification Loss, and DFL Loss decreased significantly during the initial epochs and then gradually stabilized toward the end of training at epoch 100. The decrease in Box Loss indicates an improvement in the model's ability to predict the location of the bounding boxes, while the decrease in Classification Loss indicates that the model became better at distinguishing between the harvest-ready and not harvest-ready classes. Meanwhile, the decrease in DFL Loss indicates improved accuracy in predicting the bounding box coordinates. The relatively close training and validation curves indicate a consistent learning process with no significant performance gap between the two datasets. Overall, these results show that YOLOv11 was able to learn the characteristics of the objects effectively and achieve relatively stable performance throughout the training process.

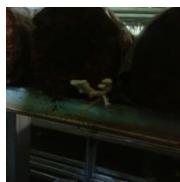
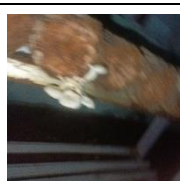
C. Model Performance Comparison

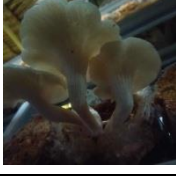
A comparative performance analysis was conducted to determine which model, CNN (ResNet50) or YOLOv11, is more effective in classifying the maturity level of oyster mushrooms. The evaluation was performed using 30 newly acquired images captured directly from the cultivation environment. These images were excluded from the training, validation, and testing datasets to ensure an unbiased assessment of the models' generalization capability. Evaluating the models on previously unseen data provides a more realistic measure of their performance under practical deployment conditions.

The predictions generated by each model were compared with the corresponding ground-truth labels and summarized using a confusion matrix. The confusion matrix reports the numbers of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN), providing the basis for quantitative performance evaluation. Based on these values, several evaluation metrics were calculated, including accuracy, which measures the overall proportion of correctly classified samples; precision, which quantifies the reliability of positive predictions; recall, which evaluates the model's ability to correctly identify all relevant instances; and the F1-score, which represents the harmonic mean of precision and recall.

In addition to classification performance, the training time required by each model was analyzed as an indicator of computational efficiency. The results obtained from these evaluation metrics were subsequently compared to identify the model that achieved the best overall performance in classifying oyster mushrooms into the harvest-ready and not harvest-ready categories while also demonstrating efficient model training.

TABLE II.
DETECTION RESULTS OF THE CNN (RESNET50) AND YOLOV11 MODELS

No	Mushroom Image	Ground Truth	Model	Prediction Result
1.		Not Harvest-Ready	CNN	Harvest-Ready 53.00%
			YOLO	Not Harvest-Ready 94.00%
2.		Not Harvest-Ready	CNN	Harvest-Ready 52.6%
			YOLO	Not Harvest-Ready 95.3%
3.		Not Harvest-Ready	CNN	Not Harvest-Ready 72.5%
			YOLO	Not Harvest-Ready 97.5%
4.		Not Harvest-Ready	CNN	Not Harvest-Ready 70.2%
			YOLO	Not Harvest-Ready 95.4%
5.		Not Harvest-Ready	CNN	Not Harvest-Ready 51.0%
			YOLO	Not Harvest-Ready 96.1%
6.		Not Harvest-Ready	CNN	Not Harvest-Ready 68.9%
			YOLO	Harvest-Ready 85.6%
7.		Not Harvest-Ready	CNN	Not Harvest-Ready 83.5%
			YOLO	Not Harvest-Ready 97.2%
...	...	...	...	...
...	...	...	...	...

No	Mushroom Image	Ground Truth	Model	Prediction Result
25.		Harvest-Ready	CNN	Harvest-Ready 93.9%
			YOLO	Harvest-Ready 93.5%
26.		Harvest-Ready	CNN	Harvest-Ready 67.2%
			YOLO	Harvest-Ready 95.0%
27.		Harvest-Ready	CNN	Harvest-Ready 98.8%
			YOLO	Harvest-Ready 93.9%
28.		Harvest-Ready	CNN	Harvest-Ready 86.9%
			YOLO	Harvest-Ready 94.2%
29.		Harvest-Ready	CNN	Harvest-Ready 89.2%
			YOLO	Harvest-Ready 94.1%
30.		Harvest-Ready	CNN	Harvest-Ready 86.7%
			YOLO	Harvest-Ready 94.0%

Following the detection process using the CNN (ResNet50) and YOLOv11 models, a comparative performance evaluation was conducted based on the accuracy, precision, recall, and F1-score metrics. These metrics were calculated from the confusion matrices generated using the prediction results presented in Table II. The confusion matrices for each model are presented below.

TABLE III.
CONFUSION MATRIX MODEL CNN & YOLO

Model	Confusion Matrix		
CNN (ResNet50)		Positive	Negative
	Harvest-Ready	15	0
	Not Harvest-Ready	3	12
YOLOv11		Positive	Negative
	Harvest-Ready	14	1
	Not Harvest-Ready	1	14

Following the construction of the confusion matrices for both models, accuracy, precision, recall, and F1-score were computed using the evaluation equations described in the previous section. The calculated metrics were subsequently used to compare the classification performance of the CNN (ResNet50) and YOLOv11 models.

TABLE IV.
PERFORMANCE COMPARISON OF THE CNN (RESNET50) AND YOLOV11 MODELS

Model	Accuracy	Precision	Recall	F1-Score	Training Time
CNN Resnet50	0.9	1	0.833	0.909	71.9 Menit
YOLO v11	0.93	0.93	0.93	0.93	89.3 Menit

Based on the results presented in Table IV, the CNN (ResNet50) model achieved an accuracy of 90%, precision of 100%, recall of 83.3%, and an F1-score of 90.9%. In contrast, the YOLOv11s model obtained balanced performance across all evaluation metrics, with accuracy, precision, recall, and F1-score each reaching 93%. These results indicate that YOLOv11s provides more consistent and robust performance in detecting and classifying oyster mushrooms than the CNN (ResNet50) model.

D. Discussion

In terms of accuracy, YOLOv11s outperformed the CNN (ResNet50) model, achieving 93% compared with 90%, indicating a higher overall classification accuracy on the evaluation dataset. This result suggests that YOLOv11s was more effective in correctly classifying oyster mushroom maturity levels.

For the precision metric, the CNN (ResNet50) model achieved the highest value of 100%, indicating that all samples predicted as harvest-ready were correctly classified without any false positive predictions. However, its recall was limited to 83.3%, implying that a number of relevant samples

were not successfully identified, resulting in false negative predictions. In contrast, YOLOv11s achieved balanced precision and recall values of 93%, demonstrating its ability to correctly identify a greater proportion of samples while maintaining a high level of prediction reliability.

The F1-score of the CNN (ResNet50) model was 90.9%, reflecting a reasonable balance between precision and recall. Nevertheless, this value remained lower than that of YOLOv11s, which achieved an F1-score of 93%. Considering all evaluation metrics, YOLOv11s demonstrated superior overall performance in classifying oyster mushrooms into the “harvest-ready” and “not harvest-ready” categories on the dataset used in this study. The comparison of the performance metrics for both models is illustrated in the following figure.

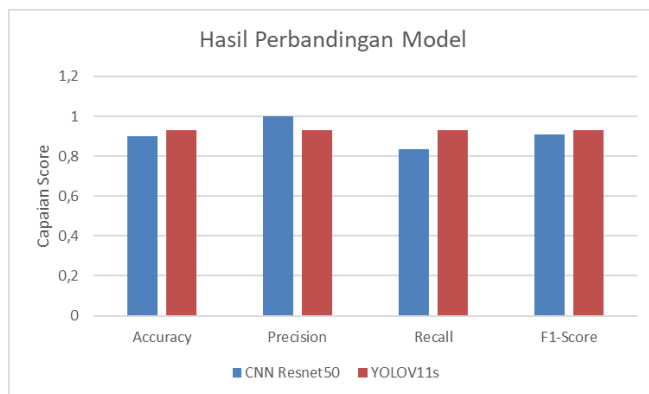


Figure 8. Comparison of Accuracy, Precision, Recall, and F1-Score

Based on the recorded training times, the CNN (ResNet50) model required 71.9 minutes to complete the training process, whereas the YOLOv11s model required a longer training time of 89.3 minutes. This difference in training time can be attributed to the distinct architectural characteristics of the two models.

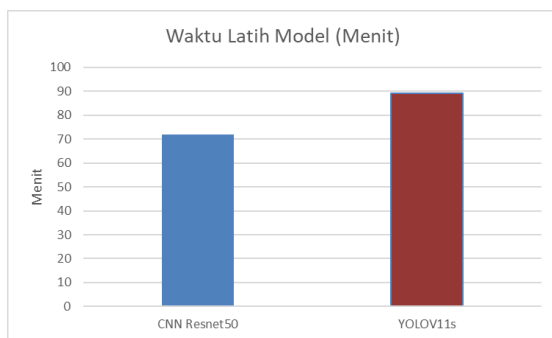


Figure 9. Comparison of Training Time

The shorter training time of the CNN (ResNet50) model can be attributed to its primary task of image classification, where the model only learns discriminative features to assign an input image to a predefined class. In contrast, YOLOv11s performs both object localization and classification simultaneously by predicting bounding boxes, confidence scores, class probabilities, and Intersection over Union (IoU)-based loss during each training iteration. Consequently,

YOLOv11s requires more computational operations, resulting in a longer training time than the CNN (ResNet50) model.

Compared with several previous studies that reported training times of only a few minutes, the training durations of 71.9 minutes for CNN (ResNet50) and 89.3 minutes for YOLOv11s are relatively longer. This can be attributed not only to the differences in model architecture but also to factors such as dataset size, image resolution, data augmentation, training configuration (e.g., the number of epochs and batch size), and the computational resources available. Although both models required relatively long training times, they achieved satisfactory performance in classifying oyster mushrooms into the “harvest-ready” and “not harvest-ready” categories.

IV. CONCLUSION

This study presents a comparative analysis of two deep learning models, namely CNN (ResNet50) and YOLOv11, for classifying oyster mushroom maturity and harvest readiness into two categories: “harvest-ready” and “not harvest-ready.” The evaluation was conducted using 30 newly acquired images that were entirely excluded from the training, validation, and testing datasets to provide an objective and unbiased assessment of each model’s generalization capability under real-world conditions.

The experimental results demonstrate that YOLOv11s achieved superior overall performance, obtaining balanced scores of 93% across all evaluation metrics, including accuracy, precision, recall, and F1-score. In contrast, the CNN (ResNet50) model achieved an accuracy of 90%, a precision of 100%, a recall of 83.3%, and an F1-score of 90.9%. Although the CNN (ResNet50) model attained perfect precision, its lower recall indicates that it failed to correctly identify a portion of the relevant samples, resulting in false negative predictions. YOLOv11s demonstrated more consistent and balanced performance across all metrics, making it the more reliable model for classifying oyster mushroom maturity in this study.

Regarding computational efficiency, the CNN (ResNet50) model completed training in 71.9 minutes, whereas YOLOv11s required 89.3 minutes. The longer training time of YOLOv11s is attributable to its simultaneous execution of object localization and classification tasks, including the prediction of bounding boxes, confidence scores, class probabilities, and IoU-based loss computation during each training iteration. Despite this trade-off, the superior detection accuracy of YOLOv11s makes it the preferred choice for automated oyster mushroom harvest readiness classification.

For future research, several improvements are recommended to further enhance model performance. First, expanding the dataset with a greater number and more diverse images, including variations in lighting, camera angle, and mushroom growth stages, is expected to improve model robustness and generalization. Second, exploring additional architectures such as YOLOv11m or YOLOv11l may yield higher detection accuracy at the cost of increased

computational requirements. Third, integrating the developed model into a real-time monitoring system, potentially combined with Internet of Things (IoT) technologies, could enable the practical deployment of automated oyster mushroom harvest readiness detection in smart farming environments.

REFERENCES

- [1] M. Riski, A. Alawiyah, M. Bakri, N. U. Putri, J. Jupriyadi, and L. Meilisa, "Alat Penjaga Kestabilan Suhu Pada Tumbuhan Jamur Tiram Putih Menggunakan Arduino UNO R3," *Jurnal Teknik dan Sistem Komputer*, vol. 2, no. 1, pp. 67–79, 2021, doi: 10.33365/jtikom.v2i1.42.
- [2] C. Christnatalis, C. D. Sozaro Lase, T. H. Sitompul, and A. P. Hondro, "Klasifikasi Penyakit Pada Baglog Jamur Tiram Menggunakan Metode Convolutional Neural Network," *Jurnal Pendidikan dan Teknologi Indonesia*, vol. 4, no. 11, pp. 359–367, 2024, doi: 10.52436/1.jpti.480.
- [3] U. Sri Rahmadhani and N. Lysbetti Marpaung, "Klasifikasi Jamur Berdasarkan Genus Dengan Menggunakan Metode CNN," *Jurnal Informatika : Jurnal Pengembangan IT*, vol. 8, no. 2, pp. 169–173, 2023.
- [4] C. Chen, F. Wang, Y. Cai, S. Yi, and B. Zhang, "An Improved YOLOv5s-Based Agaricus bisporus Detection Algorithm," *Agronomy*, vol. 13, no. 7, 2023, doi: 10.3390/agronomy13071871.
- [5] L. Shi *et al.*, "OMC-YOLO: A Lightweight Grading Detection Method for Oyster Mushrooms," *Horticulturae*, vol. 10, no. 7, 2024, doi: 10.3390/horticulturae10070742.
- [6] A. Alqhifari, P. Studi, M. Teknologi, I. Universitas, and K. Bukit, "Klasifikasi Citra Mebel Menggunakan," vol. 13, no. 2, 2025.
- [7] C. R. Gunawan, N. Nurdin, and F. Fajriana, "Deteksi Ikan Segar Secara Realtime dengan YOLOv4 menggunakan Metode Convolutional Neural Network," *Jurnal Komtika (Komputasi dan Informatika)*, vol. 7, no. 1, pp. 1–11, 2023, doi: 10.31603/komtika.v7i1.8986.
- [8] J. Ulfah and N. Nurdin, "Implementasi Metode Deteksi Tepi Canny Untuk Menghitung Jumlah Uang Koin Dalam Gambar Menggunakan Opencv," *J. Inform. dan Tek. Elektro Terap.*, vol. 11, no. 3, Aug. 2023, doi: 10.23960/jitet.v11i3.3147.
- [9] P. Aggarwal, N. Kumar, B. Fatimah, and P. Singh, "COVID-19 image classification using deep learning : Advances , challenges and opportunities," *Comput. Biol. Med.*, vol. 144, no. November 2021, p. 105350, 2022, doi: 10.1016/j.compbimed.2022.105350.
- [10] S. Yousef and F. Qudaisat, "The Impact of ResNet50 on Image Recognition Accuracy Compared to Prior Convolutional Neural Networks," *Journal of Information Systems Engineering and Management*, vol. 2025, no. 57s, pp. 2468–4376, 2024, [Online]. Available: <https://www.jisem-journal.com/>
- [11] T. Diwan, G. Anirudh, and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural successors, datasets and applications," *Multimed. Tools Appl.*, vol. 82, no. 6, pp. 9243–9275, 2023, doi: 10.1007/s11042-022-13644-y.
- [12] A. Ilyana, N. Nurdin, and M. Maryana, "Real-Time Detection of Coffee Cherry Ripeness Using YOLOv11," *J. Appl. Informatics Comput.*, vol. 9, no. 4, pp. 1170–1178, 2025.
- [13] L. Deng, Y. Tan, D. Zhao, and S. Liu, "Research on object detection in remote sensing images based on improved horizontal target detection algorithm," *Earth Sci. Inform.*, vol. 18, no. 3, pp. 1–25, 2025, doi: 10.1007/s12145-025-01814-z.
- [14] C. R. Gunawan, N. Nurdin, and F. Fajriana, "Design of a Real-Time Object Detection Prototype System with YOLOv3 (You Only Look Once)," *Int. J. Eng. Sci. Inf. Technol.*, vol. 2, no. 3, pp. 96–99, 2022.
- [15] Y. Luo, A. Wu, and Q. Fu, "MAS-YOLOv11: An Improved Underwater Object Detection Algorithm Based on YOLOv11," *Sensors*, vol. 25, no. 11, pp. 1–29, 2025, doi: 10.3390/s25113433.
- [16] Y. Yanto, F. Aziz, and I. Irmawati, "Yolo-V8 Peningkatan Algoritma Untuk Deteksi Pemakaian Masker Wajah," *JATI (Jurnal Mahasiswa Teknik Informatika)*, vol. 7, no. 3, pp. 1437–1444, 2023, doi: 10.36040/jati.v7i3.7047.
- [17] Z. Hong *et al.*, "Multi-Scale Ship Detection from SAR and Optical Imagery Via A More Accurate YOLOv3," *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.*, vol. 14, pp. 6083–6101, 2021, doi: 10.1109/JSTARS.2021.3087555.
- [18] S. Du, P. Zhang, B. Zhang, and H. Xu, "Weak and Occluded Vehicle Detection in Complex Infrared Environment Based on Improved YOLOv4," *IEEE Access*, vol. 9, pp. 25671–25680, 2021, doi: 10.1109/ACCESS.2021.3057723.
- [19] A. Putra Pranjaya, F. Rizki, R. Kurniawan, and N. K. Daulay, "Penyakit Pada Daun Tanaman Padi Berbasis YoloV5 (You Only Look Once)," *Media Online*, vol. 4, no. 6, pp. 3127–3136, 2024, doi: 10.30865/klik.v4i6.1916.
- [20] Y. Zhou *et al.*, "Adaptive Active Positioning of Camellia oleifera Fruit Picking Points: Classical Image Processing and YOLOv7 Fusion Algorithm," *Applied Sciences (Switzerland)*, vol. 12, no. 24, 2022, doi: 10.3390/app122412959.
- [21] H. Wang, X. Xu, Y. Liu, D. Lu, B. Liang, and Y. Tang, "Real-Time Defect Detection for Metal Components: A Fusion of Enhanced Canny–Deverney and YOLOv6 Algorithms," *Applied Sciences (Switzerland)*, vol. 13, no. 12, 2023, doi: 10.3390/app13126898.
- [22] H. T. Vo, K. C. Mui, N. N. Thien, and P. P. Tien, "Automating Tomato Ripeness Classification and Counting with YOLOv9," *International Journal of Advanced Computer Science and Applications*, vol. 15, no. 4, pp. 1120–1128, 2024, doi: 10.14569/IJACSA.2024.01504113.
- [23] R. Sapkota, Z. Meng, M. Churuvija, X. Du, Z. Ma, and M. Karkee, "Comprehensive Performance Evaluation of YOLO11, YOLOv10, YOLOv9 and YOLOv8 on Detecting and Counting Fruitlet in Complex Orchard Environments," 2024.
- [24] J. Ulfah, M. Ula, F. Fajriana, and N. Nurdin, "Analisis Perbandingan Kinerja Algoritma You Only Look Once (YOLOv8) Dan Single Shot Detector (SSD) dalam Pengenalan Nominal Uang Kertas," *Journal of Artificial Intelligence and Software Engineering*, vol. 4, no. 2, 2024, doi: 10.30811/jaise.v5i4.7471.
- [25] L. M. Al Ikhlās, D. Abdullah, and N. Nurdin, "Environmental Image Analysis for Smoke-Free Area Compliance Mapping Using YOLOv11 and Geographic Information Systems", *JAIC*, vol. 10, no. 3, pp. 3034–3045, Jun. 2026.