Comparative Analysis of Deep Learning Models for Olive Detection on the Branch

ERHAN KAHYA[1], YASIN ASLAN[2]
[1]Department of Electronic and Automation
Tekirdağ Namık Kemal University
Tekirdağ,
TURKEY

[2]Freelance Senior Software Developer
Tekirdağ,
TURKEY

Abstract: - The future of deep learning integration in agriculture holds great potential for advancing sustainable agricultural practices, precision agriculture and improved decision-making. With the rapid development of image processing and artificial intelligence technologies in recent years, deep learning has begun to play a major role in identifying agricultural pests and optimizing agricultural product marketing. However, there are challenges related to data quality, model scalability, and geographical limitations for widespread adoption of deep learning in agriculture. This study on Olive was conducted to improve the quality of the data set and to ensure more reliable training of object detection models. According to the result of the training process of YOLOv7 used in the study, it was concluded that it was characterized by decreasing loss values and showed an increase in the model's ability to detect objects correctly. It was observed that the other model, YOLOv8l, had a more effective learning capacity and a tendency to learn faster. The performance of both models was evaluated with various metrics, and it was determined that YOLOv8l had higher Precision, Recall, and mAP values. It was emphasized that YOLOv8l showed high performance even in low epoch numbers and can be preferred especially in cases where time and computational resources were limited. It was determined that YOLOv7 made detections in a wide confidence range, but had difficulty in detections with low confidence scores. It was observed that YOLOv8l made more stable and reliable detections with higher confidence scores. The metric data of the "YOLOv8l" model was found to be higher compared to other models. The F1 score of the YOLOv5l model was 92.337%, precision 96.568%, recall %88,462,mAP@0.5:0.65 value gave the highest score with 94.608%. This research on deep learning-based object detection models indicated that YOLOv8l showed superior performance compared to YOLOv7 and was a more reliable option for agricultural applications.

Key-Words: - Deep learning, YOLOv8, YOLOv7, identification, analysis, classification.

Received: August 21, 2023. Revised: November 15, 2023. Accepted: December 11, 2023. Published: December 31, 2023.

1 Introduction

Deep learning has had a major impact on the agricultural sector and has started to be used in many areas. In agricultural applications, deep learning is used in crop management, water management, soil management, livestock management, and classification. Deep learning is an artificial intelligence method that uses multi-layered artificial neural networks and can be used in areas such as soil analysis, disease, and pest detection in agriculture. Deep learning is used to increase agricultural productivity and improve production. In particular, the ability of deep learning to deal with large data sets shows that it is highly effective in agricultural applications, [1]. Deep learning is used in a wide variety of areas in agricultural applications. The successes achieved with deep learning methods in the agricultural image recognition competition have revealed the potential in this field, [2]. Deep learning is also used in processing agricultural sensor data and forecasting agricultural production, [3]. Deep learning plays an important role in issues such as sustainability, productivity, reduction of input costs, food safety, and environmental sustainability in agricultural applications, [4]. The use of these technologies in agricultural applications enables the development of smarter and sustainable agricultural practices, [5]. Deep learning is used in areas such as agricultural

data analytics, disease diagnosis, and production forecasting to increase the automation and efficiency of agricultural processes, [6]. Deep learning models are also used for decision support systems in agricultural production, [7]. In this context, it is seen that deep learning techniques are used in a wide range of agricultural applications and provide a significant transformation in the agricultural sector. The use of these techniques contributes to making agricultural processes more efficient, sustainable, and smart.

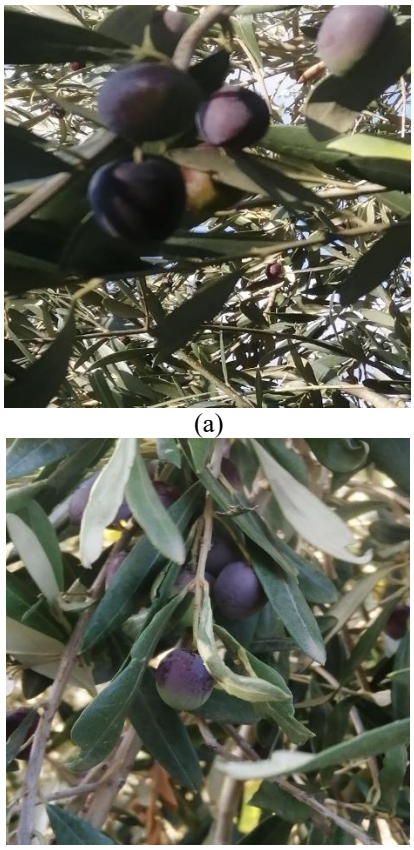
2 Material and Method

2.1 Material

Olive is the fruit of the olive tree and is usually used for olive oil production or direct consumption. There are two main types. The first is table olives, which are used for direct consumption and are usually pickled in water or stored in oil. The other is small, bitter-pressed olives that are generally used for olive oil production. Olive trees generally grow in regions specific to the Mediterranean climate, and the harvest time of this fruit varies depending on the geographical location and olive type. Olives generally ripen in autumn and are harvested during this period. The olive harvesting process is a very important stage in olive oil production as it significantly affects the quality of the final product. Olive harvesting can be performed using manual or mechanical methods, and each approach has specific effects. Manual harvesting involves manually picking olives from trees or collecting olives that have fallen to the ground and is usually done by farm workers, [8]. On the other hand, mechanical harvesting uses technological methods such as shaking trees to dislodge olives. This method may be more efficient, but can also cause potential damage to the fruit, [9]. The labor aspect of olive harvesting is also an important issue. This is because hand picking of olives involves the participation of farm laborers, [10]. This emphasizes the social and occupational health dimensions associated with olive harvesting and emphasizes the need for fair labor practices and worker welfare in agricultural settings.

2.2 Method

While preparing the data set of the olive fruit, which was targeted for object detection and analysis within the scope of the study, 140 images taken from the producer olive field in Tekirdağ Naip Village and various internet sources were used. The parts containing each olive image within the images were marked with a bounding box area. Sample images taken from the producer's field are shown in Figure 1 (a-b).



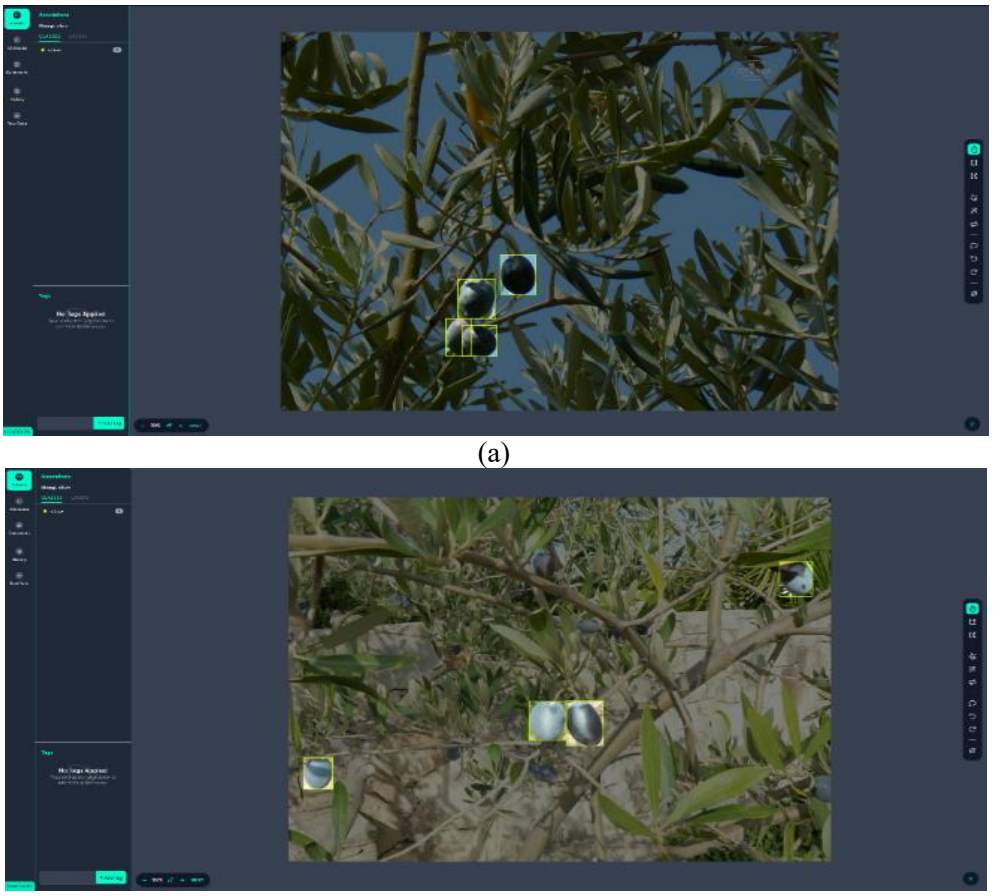
(a) (b)

Fig. 1: a-b: Examples of educational images

2.2.1 Labeling

For an object detection model to be able to train on a dataset, the objects to be detected must be labeled/signed in the dataset to be trained. For this reason, the parts containing the olive image in each of the 140 images should be marked with the bounding box area and assigned to the "olive" class, which is the object class it belongs to. There are many programs, websites, and tools available in the open-source communities for image labeling. RoboFlow is a versatile platform that offers a range of functions for robotics and artificial intelligence applications. It enables the development and training of machine learning models by providing users with access to public datasets and the ability to send private data, [1]. The platform also supports pre-processing of images and their labels, as well as the creation of bounding boxes for tasks such as object detection and classification, [10], [11]. RoboFlow enables end users to control robots by developing flowchart-like programs consisting of visual markers and making them accessible to

people without extensive programming expertise, [12]. Roboflow offers a flow-based visual programming language specifically designed for engineering robot motion and manipulation tasks by meeting the needs of both inexperienced and experienced programmers, [13]. This marking and labeling process is easily done through the graphical user interface of the site. The Label screen is shown in Figure 2. a-b.



(a)

(b)

Fig. 2: a-b: Label display

2.2.2 Training Model Selection

In the study we carried out, the YOLOv7 and YOLOv8 families, developed as an open source of the YOLO model family developed by the CNN method, were preferred. The YOLOv7 architecture, an extension of YOLOv4, has been developed with several architectural reforms, including the Extended Efficient Layer Aggregation Network (E-ELAN) and free trainable tools, [14]. YOLOv7 is the newest variant of the YOLO object detector family, known for its high accuracy and fast implementation, [15]. The YOLOv7-Pose model, based on YOLO-Pose and YOLOv7, uses E-LAN and E-ELAN architecture for high accuracy and fast, [16]. The YOLOv7 architecture includes a computation block called E-ELAN, which allows better learning within the framework, [17].

YOLOv8, the latest version of the YOLO series, has attracted attention with its advances in real-time object detection. It is designed to provide high accuracy while maintaining real-time performance by combining the strengths of various real-time object detectors, [18]. The YOLOv8 architecture offers five different size options that meet various application needs, from nano-size to extra-large size, [19]. Building on the strengths of its previous versions, YOLOv8 has emerged as a faster and more capable option and positions itself as a superior alternative to existing models, [20]. The inclusion of object detection skills in the YOLOv8 framework increases its potential for robotic

cognitive architectures by contributing to perceptual fixation capabilities, [21]. YOLOv7 and YOLOv8 versions used in the study contain models within themselves. YOLOv7, YOLOv8n/s/m, and l (nano-small-medium and large) models were preferred for deep learning training.

2.2.3 Initiation of Training

To start the training of the model that will perform olive detection, the location of the YOLOv7/v8 model on the computer was visited. In the next step, a Python editor was used to control and run the train.py program located in the main directory that manages the YOLOv7/v8 training. This Python program can be customized with various parameters and made suitable for specific training for Olive detection.

Within the project, the parameters and regulations in the code written below were preferred for olive fruit.

YOLOv7

!python train.py --device 0 --batch 20 --epochs 100 --data {dataset.location}/data.yaml --weights 'yolov7_training.pt'

--device: This parameter indicates the device on which the training will be performed. The value "0" is generally used for the first graph processing unit (GPU).

--batch: The number of data point packets to be used by the display card at a time while training the model.

--epochs: The number of times all training data is shown to the trained network and the weights are updated while training the model

--data: It is a configuration file that defines the training data of the model. This file contains the data path, class labels, and other data characteristics.

--weights: Indicates the initial weights to be used during training of the model. This is important in transfer learning situations. The file 'yolov7_training.pt' contains the pre-trained weights. If this parameter is not indicated, the model usually starts training with random weights.

YOLOV8

!yolo task=detect mode=train model=yolov8n.pt data={dataset.location}/data.yaml epochs=50 imgsz=640 plots=True

!yolo task=detect mode=train model=yolov8s.pt data={dataset.location}/data.yaml epochs=50 imgsz=640 plots=True

!yolo task=detect mode=train model=yolov8m.pt data={dataset.location}/data.yaml epochs=50 imgsz=640 plots=True

!yolo task=detect mode=train model=yolov8l.pt data={dataset.location}/data.yaml epochs=50 imgsz=640 plots=True

--task: Determines which task the training will perform. “detect” refers to the object detection task.

--mode: Indicates that the model will be run in training mode. YOLO can be run in different modes, for example, “train” can be used for training, and “test” can be used for testing or validation.

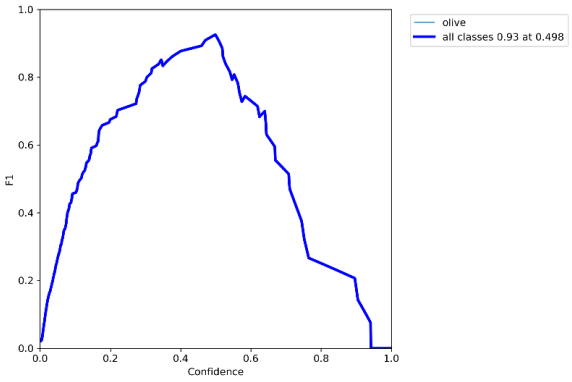
--model: Indicates the path and name of the model to be used during training.

--imgsz: The pixel size at which the images to be trained will be reduced by the YOLOv8 model.

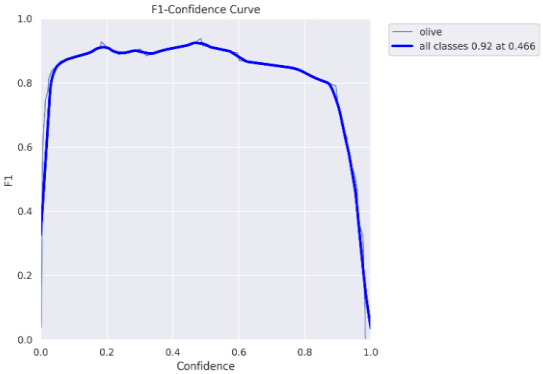
--plots: Refers to graph drawings created during training. These graphs are used to monitor the performance of the model.

As a result of running these code lines correctly, the training process of the model started. The program first checks the YOLOv7/v8 files and checks for any update status. Then, the training process is carried out during the determined number of cycles (epoch).

Examination of the results of YOLOv7 and YOLOv8l algorithms according to error matrix metrics is shown in Figure 3, Figure 4, Figure 5, Figure 6, Figure 7, Figure 8 respectively.



(Image 1, Size: 640x640, Batch: 20 Epoch: 100, Algorithm: YOLOv7)



(Image 2, Size: 640x640, Epoch: 50, Algorithm: YOLOv8l)

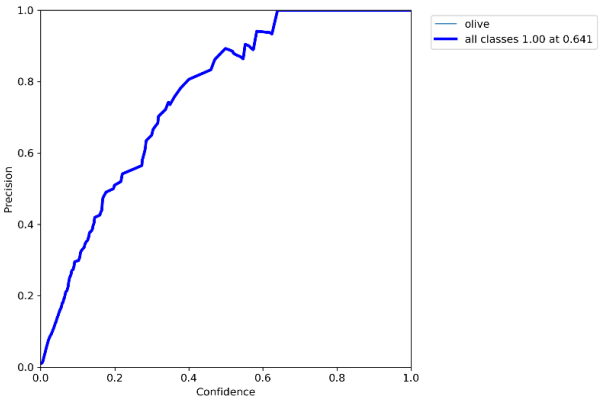
Fig. 3: F1 Score display

According to the analysis of the graphs in Figure 3, the F1 Score result analysis is as follows.The F1-Confidence curve of the YOLOv7 algorithm at 640x640 resolution and after 100 epochs of training are shown in Image 1. From the graph, it was seen that the model achieved a high F1 score for the 'olive' class. The maximum F1 score (about 0.93) was obtained at a confidence threshold of 0.498. The curve rises at lower confidence thresholds. After reaching the optimum point, it decreased with increasing confidence threshold. This indicated that the model provided a good balance between the accuracy and scope of the detections within a given confidence interval. However, it was understood that after the confidence threshold of 0.6, the F1 score decreased rapidly. It was understood that high threshold values can increase false negatives and the sensitivity of the model decreased.

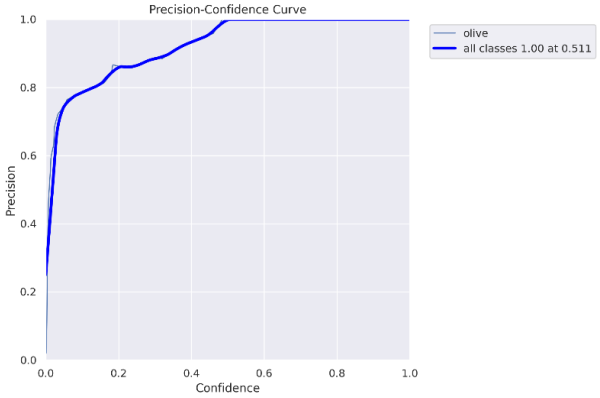
The F1-Confidence curve of the YOLOv8l algorithm obtained at 640x640 resolution and after 50 epochs of training is shown in Image 2. The model achieved an almost constant high F1 score (approximately 0.92) for the 'olive' class and this score reached to maximum at a confidence threshold of 0.466. The curve is almost a straight line over a wide confidence interval. This indicated that the model provided high accuracy and sensitivity even over a wide confidence interval. The confidence threshold of the curve decreased sharply at around 0.8, and from this point onwards, it was understood that the model significantly increased the false-negative rate and the sensitivity decreased.

In both graphs, the F1 scores of all classes were also indicated to evaluate the overall performance of the model. For YOLOv7, the F1 score of all classes was 0.93, while it was shown as 0.92 for YOLOv8l. These values indicated that both models showed high performance and gave good results over a wide threshold of confidence. For both models, it was understood that the best performance was achieved at medium confidence thresholds. It was observed that higher or lower threshold values negatively affected performance.

According to the analysis of the graphs in Figure 4, the Precision result analysis is as follows.The accuracy-confidence curve of the YOLOv7 algorithm for the 'olive' class is shown in Image 1. The curve observed in the image started with low accuracy at low confidence thresholds and increased as the confidence threshold increased. This indicated that the model showed improvement in the accuracy of detections within a certain confidence interval. The accuracy reached a maximum (1.00) at a confidence threshold value of 0.641, indicating that all of the model's detections at this threshold were correct. However, accuracy decreased rapidly as the confidence threshold increased further. This suggests that at very high confidence thresholds the model increased false negatives and therefore made fewer true positive detections.



(Image 1, Size: 640x640, Batch: 20 Epoch: 100,Algorithm: YOLOv7)

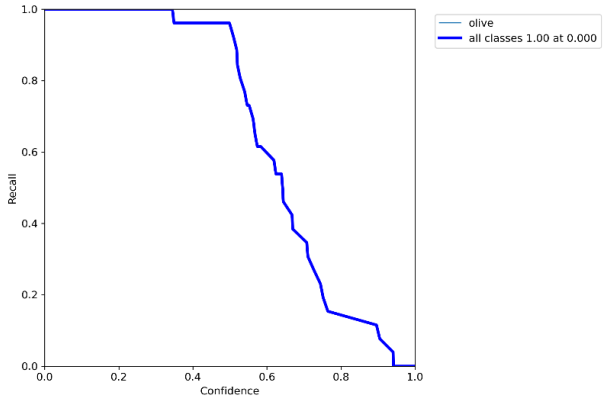


(Image 2, Size: 640x640, Epoch: 50, Algorithm: YOLOv8l)

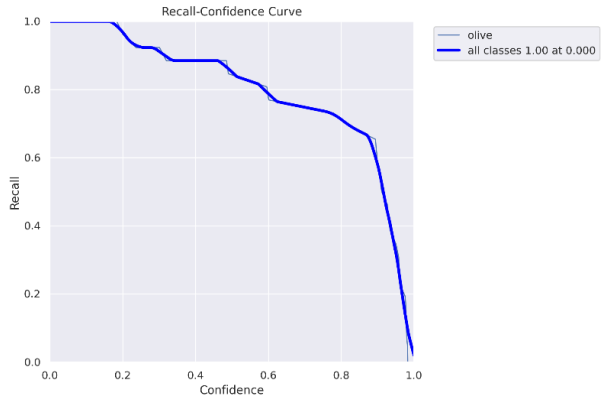
Fig. 4:Precision display

The truth-confidence curve for the 'olive' class of the YOLOv8l algorithm is shown in Image 2. The curve rose rapidly at low confidence thresholds and reached maximum accuracy (1.00) at a confidence threshold of 0.511. It was observed that the curve almost flattened towards the high confidence thresholds and remained at this level. This showed that the YOLOv8l algorithm achieved high accuracy, especially at intermediate confidence thresholds, and that the model made true positive detections even above this threshold, keeping the number of false positives minimum.

The maximum accuracy values for all classes for both models are indicated in the graph and are shown as 1.00 in both cases. This showed that the model reached the maximum accuracy value when tested on all classes.



(Image 1, Size: 640x640, Batch: 20 Epoch: 100, Algorithm: YOLOv7)



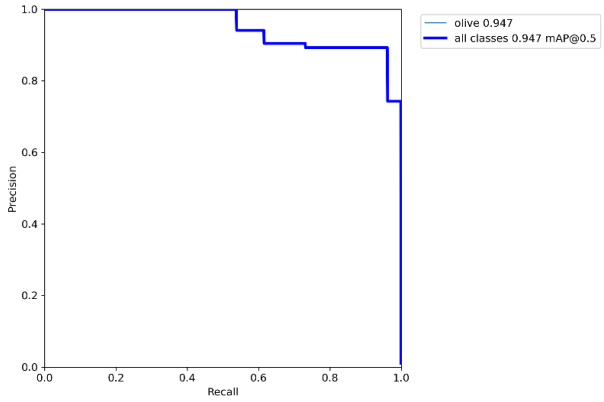
(Image 2, Size: 640x640, Epoch: 50, Algorithm: YOLOv8l)

Fig. 5:Recall display

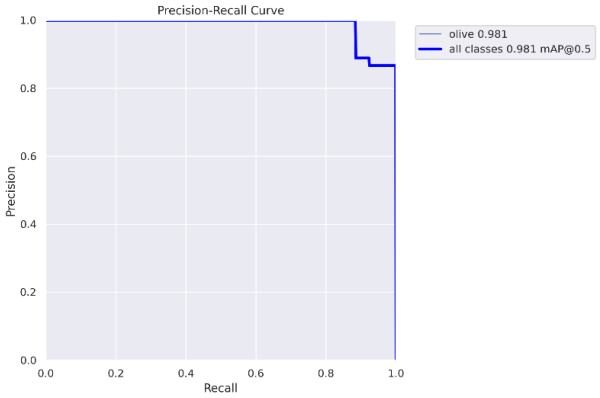
According to the analysis of the graphs in Figure 5, the Recall result analysis is as follows. The sensitivity-confidence curve of the YOLOv7 algorithm for the 'olive' class is shown in Image 1. The sensitivity started at 1.00 (perfect sensitivity) at low confidence thresholds and decreased as the confidence threshold increased. This indicated that the model was able to detect almost all true positives at low confidence thresholds. However, it was seen that as the confidence threshold increased, the model missed some true positive detections and the sensitivity decreased. In the graph, the confidence threshold decreased sharply to around 0.8. It was understood that higher threshold values significantly reduced the sensitivity by increasing the false negative rate.

The sensitivity-confidence curve of the YOLOv8l algorithm for the 'olive' class is shown in Image 2. Similarly, the model showed high sensitivity at low confidence thresholds. The sensitivity started to decrease rapidly at a relatively low confidence threshold (around 0.2) and continued to decrease as the confidence threshold increased. In this model, sensitivity decreased significantly at high confidence thresholds. This indicated that the model had a high false negative rate at high confidence thresholds.

In both graphs, the sensitivity scores of all classes were indicated to evaluate the overall performance of the model, and in both cases, excellent sensitivity (1.00) was achieved at the lowest confidence threshold. This indicated that both models were able to detect all true positives at low confidence thresholds.



(Image 1, Size: 640x640, Batch: 20 Epoch: 100, Algorithm: YOLOv7)



(Image 2, Size: 640x640, Epoch: 50, Algorithm: YOLOv8l)

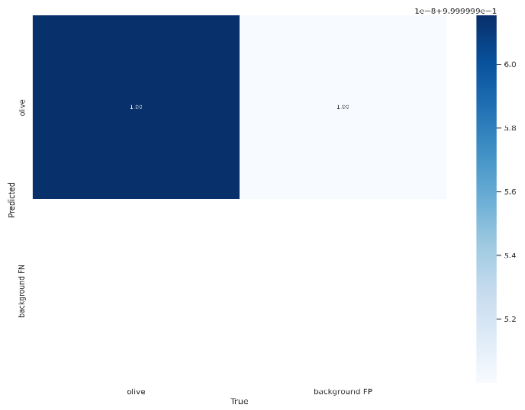
Fig. 6:Precision-Recall display

According to the analysis of the graphs in Figure 6, the Precision-Recall result analysis is as follows.In the Precision-Recall curve showing the performance of the YOLOv7 algorithm in Image 1, it was seen that a very high precision value was achieved for the 'olive' class. The graph shows that the model achieved high accuracy over a wide range of precision (recall). The mAP@0.5 value of the model for the 'olive' class was found as 0.947, indicating that the model detected objects with high precision.

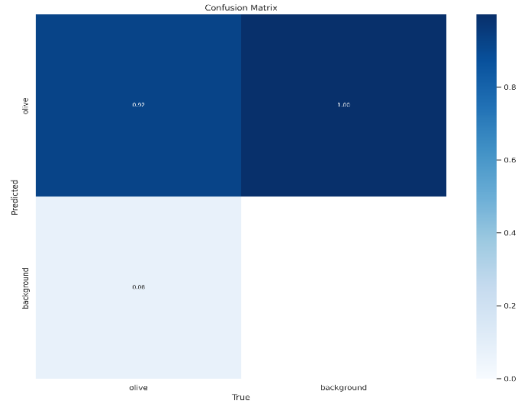
In the Image 2 graph, it was seen that in the Precision-Recall curve of the YOLOv8l algorithm, almost perfect accuracy was obtained for the 'olive' class. The curve in the graph showed that the model

detected true positives with very high accuracy and the value of mAP@0.5 was 0.981. It was understood that the classification performance of the model was outstanding and that it detected very few false positives.

In both graphs, it was seen that there was a sharp decrease in accuracy after a certain sensitivity value. This meant that the model detected all positive examples, but this caused some false positives. The graphs showed the overall performance of all classes while maintaining high accuracy and sensitivity values for the 'olive' class. The high mAP@0.5 values of both models indicated that they performed excellently in object detection. The graphs showed the overall performance of the model, as well as how balanced its performance was for different classes. While both algorithms offered high accuracy and sensitivity, it was remarkable that YOLOv8l achieved a very high value of mAP@0.5 even at a lower epoch number. This indicated that YOLOv8l may be more effective than YOLOv7 in certain situations.



(Image 1, Size: 640x640, Batch: 20 Epoch: 100, Algorithm: YOLOv7)



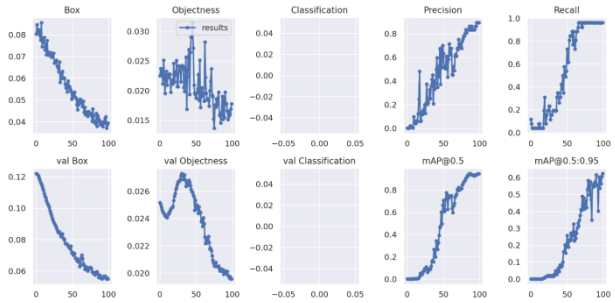
(Image 2, Size: 640x640, Epoch: 50, Algorithm: YOLOv8l)

Fig. 7: Confusion Matrix display

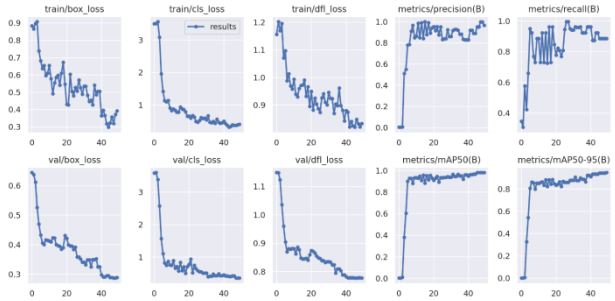
According to the analysis of the graphs in Figure 7, the Confusion Matrix result analysis is as follows.Image 1 showed that the YOLOv7 model performed almost perfectly for the "olive" class. The matrix showed a high TP ratio (1.0) for the "olive" class and a high TN ratio for the "background" class. This meant that the model detected the "olive" class almost without error and made almost no false positive predictions.

In Image 2, the confusion matrix of the YOLOv8l model again showed a high TP rate for the "olive" class (0.89), but there was an FP ratio (0.08) indicating that this model also misclassified some of the "background" class as "olive". This meant that although the YOLOv8l model run with a slightly lower accuracy rate, it still exhibited a quite high accuracy.

Both matrices showed the high sensitivity of the models in detecting the "olive" class and the overall low false positive rate. However, while the YOLOv7 model almost completely avoided false positives in the "background" class, the YOLOv8l model was somewhat weak in this regard. This indicated that YOLOv7 was able to discriminate better than YOLOv8l in this particular case.



(Image 1, Size: 640x640, Batch: 20, Epoch: 100, Algorithm: YOLOv7)



(Image 2, Size: 640x640, Epoch: 50, Algorithm: YOLOv8l)

Fig. 8: Loss Function display

According to the analysis of the graphs in Figure 8, the Loss Function result analysis is as follows.The graphs showed how various loss functions and performance metrics changed during the training process of the YOLOv7 and YOLOv8l

algorithms. Such graphs are used to understand the learning process and performance of the model on training and validation sets.

Image 1 Analysis (YOLOv7)

This series of graphs shows the training process of the YOLOv7 model trained at 640x640 resolution, with 20 batch sizes and 100 epochs.

- **Box Loss:** It measures the availability of the bounding boxes predicted by the model to the real boxes. A downward trend indicates that the model starts to predict bounding boxes more accurately over time.
- **Objectness Loss:** It measures how accurately the model predicts the presence of an object. A decrease indicates that the model better discriminates object presence over time.
- **Classification Loss:** It measures the classification accuracy of the model. A decrease indicates that the model classifies classes more accurately over time.
- **Precision ve Recall:** It shows the accuracy (how accurate positives are detected) and sensitivity (how many of all positives are detected) values of the model. An increase in both values indicates an improvement in the model's performance.
- **mAP@0.5 ve mAP@0.5:0.95:** It measures the average accuracy rates of the model. In general, an increasing trend indicates that the model can generalize well over all classes.

Image 2 Analysis (YOLOv8l)

The second series of graphs shows the training process of the YOLOv8l model trained at 640x640 resolution and 50 epochs.

- **Train Box/Class/DFL Loss:** Bounding box, classification, and directional focus loss on the training set of the model. Downward trends indicate that the model shows improvement during the training process.
- **Val Box/Class/DFL Loss:** The losses of the model on the validation set. Decreases indicate that the model also shows improvement in the validation set and does not make overfitting.
- **Precision/Recall:** Accuracy and precision of the model on the training set. Stable and high values indicate the reliability of the model.
- **Metrics/mAP50/B:** Average accuracy rates of the model. High mAP values indicate that the model makes consistent detections over a wide range.

In both images, it was seen that the performance and loss functions of the model in the training process tended to decrease. It was understood that YOLOv7 had a more stable training process with a longer training time (100 epochs), while YOLOv8l showed a high performance in a shorter time (50 epochs). It was observed that both models had high mAP values and generally showed high accuracy and sensitivity.

3 Research Results

The training process of YOLOv7 was characterized by generally decreasing loss values. The decrease in Box, Objectness, and Classification Loss values indicated an increase in the model's ability to correctly detect objects within bounding boxes, and correctly identify and classify object presence. The increase in the Precision and Recall metrics indicated that the accuracy and sensitivity of the model's detections increased. The metrics mAP@0.5 and mAP@0.5:0.95 indicated the reliability and interclass consistency of the model with high values. The loss graphs for YOLOv8l showed a similar trend of improvement, but higher mAP values were obtained at a shorter number of epochs. This indicated that the learning capacity and speed of YOLOv8l was more efficient than YOLOv7. Precision and Recall values showed that the model was highly capable of accurately detecting and classifying objects.

The confusion matrices showed that YOLOv7 showed almost a perfect performance in classification, while YOLOv8l ran with a certain margin of error. The Precision-Recall curves showed that both models had high Precision and Recall values, but the YOLOv8l performed nearly perfectly even at a low epoch count.

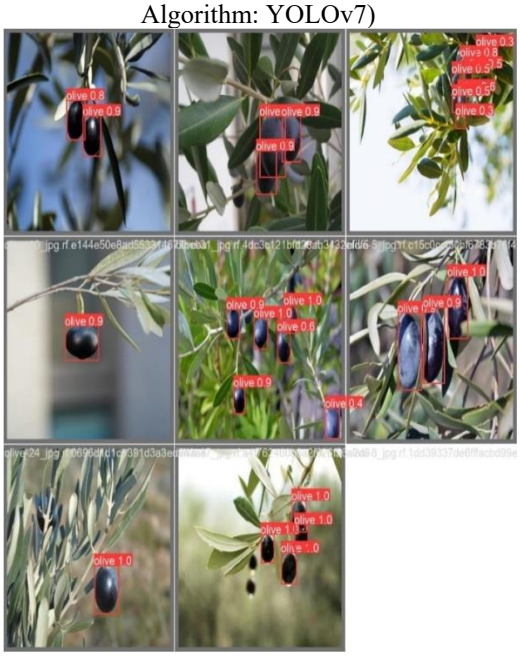
Both models exhibited high performance, but it was remarkable that YOLOv8l had a tendency to learn faster and achieved high mAP values with fewer epochs. This indicated that YOLOv8l may be preferable, especially when time and computational resources are limited. Despite the longer training time, YOLOv7 showed a more stable performance during the learning process. Both models provided high accuracy and precision, but YOLOv8l provided a slight advantage in classification.

Extensive analysis of training processes and performance metrics showed that both models were suitable for object detection. However, the performance of YOLOv8l showed that it was more

efficient, especially in terms of training time and computational resources.

Figure 9 shows the test result screen, Figure 10 shows the "Validation Batch" prediction markings resulting from the training of YOLOv7 and Figure 11 shows the YOLOv8l models.

(Image 1, Size: 640x640, Batch: 20, Epoch: 100, Algorithm: YOLOv7)



(Image 2, Size: 640x640, Epoch: 50, Algorithm: YOLOv8l)

Fig. 9: Test result screen

Fig. 10: Size: 640x640, Batch: 20, Epoch: 100, Algorithm: "Validation Batch" prediction markings resulting from the training of the YOLOv7 model

Fig. 11:Size: 640x640, Epoch: 50, Algorithm: "Validation Batch" prediction markings resulting from the training of the YOLOv8l model

The training of YOLOv7 and YOLOv8l models with different epoch numbers affects the learning dynamics. YOLOv7 was trained for 100 epochs and YOLOv8l was trained for 50 epochs. During the training process, it was observed that the loss values in the YOLOv7 model showed a slower decrease, whereas the YOLOv8l model showed a faster and more stable decrease trend. This indicates that the

learning capacity and speed of YOLOv8l is more effective than YOLOv7. Furthermore, when the performances of both models on the validation data are analyzed, it is concluded that YOLOv8l shows high accuracy and sensitivity despite the lower number of epochs, while YOLOv7 is not as effective despite the longer training time.

The analysis of the hyperparameters (e.g. learning rate, momentum, weight reduction) specified in the model's configuration files significantly affects the performance of each model. The different sets of hyperparameters used in the YOLOv7 and YOLOv8l models directly affected the training processes and results of the models. For example, the higher learning rate applied in the YOLOv8l model enabled the model to learn faster and generalize more effectively. In contrast, the lower learning rate in the YOLOv7 model caused the model to learn more slowly but more carefully. These differences were reflected in the object detection performances as well as the training processes of the models.

Analyzing the performance of both models under different lighting conditions and background scenarios is an important factor in evaluating the applicability of the models in real-world applications. In the tests, the YOLOv8l model gave better results in low-light conditions and images with complex backgrounds. The YOLOv7 model, on the other hand, performed better under standard lighting and less complex backgrounds. This shows that YOLOv8l is capable of better generalization in challenging conditions.

The data augmentation techniques used affected the performance of both models. Traditional data augmentation techniques (e.g. rotation, translation) applied in the YOLOv7 model increased the generalization capability of the model. In the YOLOv8l model, more advanced data augmentation techniques (e.g. color space transformations, geometric transformations) were used, which made the model more robust to more diverse and challenging data sets.

The comparison between the different sizes of the YOLOv8 series (nano, small, medium, large) shows that each model size offers different advantages in terms of performance and computational costs. In particular, the YOLOv8l model, while offering high accuracy and mAP values, requires more computational resources. On the other hand, smaller models such as YOLOv8n provide adequate performance with fewer computational resources. This emphasizes the importance of choosing the model size according to different application requirements.

When the performance of both models is evaluated in real-time applications, it is observed that YOLOv8l is superior in terms of processing speed and detection accuracy. In particular, the YOLOv8l model offers faster detection times and high accuracy rates, while the YOLOv7 model can make stable detections despite its slower processing time. These results show that model selection in real-time applications depends not only on accuracy rates but also on processing speed. YOLOv8l stands out as a suitable option for applications requiring high speed and accuracy, while YOLOv7 can be preferred in less urgent situations and in scenarios where stability is more important. The YOLOv7 model detected olives after 640x640 resolution, 20 batch sizes, and 100 epochs of training. The images included situations with different densities of olives in various environments. The model detected olives at a wide confidence interval. In some cases, correct detections were made with very high confidence scores (0.9 and above), while lower confidence scores (e.g. 0.3) were used in some detections. Detections with low confidence scores can often indicate false positives or detections made under challenging conditions. This is important in evaluating the model's balance of precision and sensitivity, as well as its ability to generalize different image conditions.

The YOLOv8l model was tested on similar images at 640x640 resolution and after 50 epochs of training. High confidence scores (0.9 and above) were seen in most of the detections which indicated that the model was able to make fairly accurate and precise detections. However, the small number of detected objects may mean that the model misses true positives in some cases. This tests the sensitivity of the model, especially in compelling scenarios such as difficult lighting conditions or overlapping objects.

When the outputs of both models were compared, it was seen that YOLOv8l tended to make detections with higher confidence scores. This indicated that the model followed a more stable and reliable learning path during training and showed a better generalization performance. YOLOv7, on the other hand, made detections at a wider confidence interval, indicating that the model may make more false positive detections in certain scenarios.

4 Discussion and Conclusion

The integration of deep learning techniques into agriculture has attracted great attention in recent years. Deep learning has been successfully applied in various areas of agriculture, including smart

agriculture, precision agriculture, and agricultural data analysis, [22]. In their study on image description and deep learning in agriculture, [23] indicated that deep learning made significant progress in the detection of crop diseases. Furthermore, [24] highlighted the challenges and opportunities of applying machine and deep learning technologies in agriculture by providing insights regarding the specific challenges encountered in Nigerian agriculture. The future of deep learning in agriculture has great potential for advances in sustainable agricultural practices, precision agriculture, and improved decision-making. [25] reflected the rapid development of image processing and artificial intelligence technologies in recent years and emphasized the critical role of deep learning in identifying corn pests. Furthermore, the potential of deep learning to optimize e-commerce marketing of agricultural products was highlighted by [26], pointing out different applications of deep learning in various agricultural fields. [27], identified diseases and pests in tomatoes by applying deep learning methods to identify and manage crop diseases in agricultural harvesting. [28], used image segmentation of fruit characteristics for harvest in their study using deep learning methods in fruit recognition and evaluation for agricultural harvesting. [29], used the YOLOv4 deep learning model to locate apples for robotic apple harvesting. They found that the robotic harvest detection success rate of the study was 92.9%. [30], detected bunches of grapes on the vine using the YOLOv4 deep learning model. [31], performed detection and classification of bell pepper leaf diseases using artificial neural networks. With the model they created, they found that the pepper plant leaf was healthy or bacterial with 99.99% predictive accuracy. [32], used deep learning methods for robotic tomato harvesting in their study. In addition, they determined the cut-off point for the robotic harvesting of tomatoes by image analysis method.

[33], conducted a prediction study by planting seeds of 692 native genotypes in their study entitled Fruit Yield Prediction in Pepper Using an Artificial Neural Network. The integration of deep learning into agriculture offers numerous opportunities for the development of agricultural practices, improving crop management, and advancing sustainable agriculture. However, addressing challenges related to data quality, model scalability, and geographic limitations is crucial for the widespread adoption of deep learning in agriculture. The information provided by the reviewed academic papers provides valuable perspectives on the current state and future potential of deep learning in agriculture. The results of our study are in parallel with similar research we have stated. YOLOv7 and YOLOv8 algorithms were used in the training of the network. A comparison of the test performance evaluation results of these algorithms is given in Table 1.

In Table 1, the object detection performances of different variations of the YOLOv7 and YOLOv8l models were compared. While the performance of YOLOv7 after 99 epochs was notable for a significantly lower F1 score (28.842%), each version of the YOLOv8 series, especially YOLOv8l (92.337% F1 score), performed much higher. These performance measurements are critical in evaluating the efficiency and reliability of each model in the object detection task. The low F1 score of the YOLOv7 model indicated that the model was not optimal in terms of accuracy and scope of its detections. In particular, a low Recall rate (19.56%) indicated that the model missed a large proportion of the objects that were actually positive. On the other hand, YOLOv8l's high Precision (96.568%) and very high Recall (88.462%) rates indicated that the model showed superior performance in terms of both accuracy and scope. YOLOv8l's mAP@0.5:0.95 metric (94.608%) showed that the model successfully detected even objects with high levels of difficulty and correctly classified a wide range of objects.

Table 1. Comparison of Model Performance Values

Model	Epoch	F1 Score	Precision	Recall	mAP@0.5:0.95
Yolov7	99	28.842%	54.89%	19.56%	62.37%
Yolov8n	49	95.293%	96.088%	94.512%	87.223%
Yolov8s	49	94.469%	89.517%	100.000%	90.491%
Yolov8m	49	91.772%	91.243%	92.308%	90.222%
Yolov8l	**49**	**92.337%**	**96.568%**	**88.462%**	**94.608%**

Note: The F1 Score Value in the table was calculated according to the formula [F1=2×(Precision.Recall/Precision+Recall].

As a result of this comparative analysis, it was concluded that YOLOv8l had a significant advantage in object detection performance than YOLOv7. The high F1 score and mAP@0.5:0.95 values of YOLOv8l model showed that it was a superior option as an advanced algorithm for object detection, especially in various and challenging detection scenarios. The fact that YOLOv7 showed low performance despite a higher number of epochs, perhaps reflected the difficulties or configuration deficiencies that the model encountered during the training process. The performance of YOLOv8l

indicated that this model should be preferred in real-world applications as well as object detection tasks. In our study, a comprehensive methodology and analysis approach was made in the development and evaluation of deep learning-based object detection models. According to the research results, it is understood that the advantages provided by the YOLOv8l model will provide a basis for further optimization and extension of the models in future research and applications.

References:

[1] Kamaruzaman, A. S. F., Ani, A. I. C., Farid, M., Bakar, S. J. A., Maruzuki, M. I. F., Setumin, S., & Hadi, M. S. (2023). Systematic literature review: application of deep learning processing technique for fig fruit detection and counting. *Bulletin of Electrical Engineering and Informatics*, 12(2), 1078-1091.

[2] Yang, J., Guo, X., Li, Y., Marinello, F., Ercisli, S., & Zhang, Z. (2022). A survey of few-shot learning in smart agriculture: developments, applications, and challenges. *Plant Methods*, 18(1)

[3] Wang, D., Cao, W., Zhang, F., Li, Z., Xu, S., & Wang, X. (2022). A review of deep learning in multiscale agricultural sensing. *Remote Sensing*, 14(3), 559.

[4] Ryo, M., Schiller, J., Stiller, S., Palacio, J. C. R., Mengsuwan, K., Safonova, A., & Wei, Y. (2022). Deep learning for sustainable agriculture needs ecology and human involvement. *Journal of Sustainable Agriculture and Environment*, 2(1), 40-44.

[5] Jin, X., Zhu, X., Ji, J., Ma, Y., Xie, X., & Zhao, B. (2023). Design and research an online diagnosis platform for tomato seedling facilities production diseases, *Research Square*, p.1-23, https://doi.org/10.21203/rs.3.rs-3121099/v1.

[6] Derisma, D., Rokhman, N., & Usuman, I. (2022). Systematic review of the early detection and classification of plant diseases using deep learning. *IOP Conference Series: Earth and Environmental Science*, 1097(1), 012042. IOP Publishing.

[7] Khairi, D. u., Ahsan, K., Badshah, G., Ali, S. Z., Raza, S. A., Alqahtani, O., & Shiraz, M. (2023). Comparison analysis of machine learning classification on apple fruit quality, *Research Square*, p.1-24. https://doi.org/10.21203/rs.3.rs-3025343/v1.

[8] Habib, R. R. and Fathallah, F. A. (2012). Migrant women farm workers in the occupational health literature. *Work, 41*, 4356-4362. IOS Press.

[9] Berg, H., Maneas, G., & Engstrom, A. (2018). A comparison between organic and conventional olive farming in Messenia, Greece. *Horticulturae*, 4(3), 15.

[10] Tariq, S., Hakim, A., Siddiqi, A. A., & Owais, M. (2022). An image dataset of fruitfly species (bactrocera zonata and bactrocera dorsalis) and automated species classification through object detection. *Data in Brief*, 43, 108366.

[11] Aman Jain, Jatin Gupta, Somya Khandelwal, Surinder Kaur. (2021). Vehicle License Plate Recognition. Fusion: Practice and Applications, *Fusion:Practice and Applications (FPA)*,4 (1), 15-21.

[12] Akiki, P. A., Akiki, P. A., Bandara, A. K., & Yu, Y. (2020). Eud-mars: end-user development of model-driven adaptive robotics software systems. *Science of Computer Programming*, 200, 102534. https://doi.org 10.1016/j.scico.2020.102534

[13] Dave, K. (2018). Comparison of flow-based versus block-based programming for naive programmers. Toronto Metropolitan University,Canada,Thesis. https://doi.org/10.32920/ryerson.14652246.v2

[14] Delgado, G., Cortés, A., García, S., Loyo, E., Berasategi, M., & Aranjuelo, N. (2023). Methodology for generating synthetic labeled datasets for visual container inspection. *Transportation Research Part E: Logistics and Transportation Review*, 175, 103174.

[15] Holla, B. A., Pai, M. M. M., Verma, U., & Pai, R. M. (2023). Enhanced vehicle re-identification for smart city applications using zone specific surveillance. *IEEE Access*, 11, 29234-29249.

[16] Dao, N. D., Le, T. V., Tran, H. T. M., Nguyen, Y. T. H., & Duy, T. D. (2022). The combination of face identification and action recognition for fall detection. *Journal of Science and Technology Issue on Information and Communications Technology*, 37-44.

[17] Ghourabi, M., Mourad-Chehade, F., & Chkeir, A. (2023). Eye recognition by yolo for inner canthus temperature detection in the elderly using a transfer learning approach. *Sensors*, 23(4), 1851.

[18] Lou, H., Duan, X., Guo, J., Liu, H., Bi, L., & Chen, H. (2023). Dc-yolov8: small size object detection algorithm based on camera sensor.

Electronics,12(10):2323. https://doi.org/10.3390/electronics12102323.

[19] Dehaerne, E., Dey, B., Esfandiar, H., Verstraete, L., Suh, H. S., Halder, S., … & De Gendt, S. (2023). Yolov8 for defect inspection of hexagonal directed self-assembly patterns: a data-centric approach. *38th European Mask and Lithography Conference (EMLC 2023)*,Dresden,Dermany.

[20] Asante, I., Tsun, M. T. K., Jo, H. S., & McCarthy, C. (2023). Segmentation-based angular position estimation algorithm for dynamic path planning by a person-following robot. *IEEE Access*, 11, 41034-41053.

[21] González-Santamarta, M. Á., Rodríguez-Lera, F. J., & Olivera, V. M. (2023). SAILOR: perceptual anchoring for robotic cognitive architectures, arXiv:2303.08204, *arxiv.org*. https://doi.org/10.48550/arXiv.2303.08204.

[22] Kumar, R., Kumar, P., Aljuhani, A., Islam, A. K. M. N., Jolfaei, A., & Garg, S. (2023). Deep learning and smart contract-assisted secure data sharing for iot-based intelligent agriculture. *IEEE Intelligent Systems*, 38(4), 42-51.

[23] Mamat, N., Othman, M. F., Abdulghafor, R., Belhaouari, S. B., Normahira, M., & Hussein, S. F. M. (2022). Advanced technology in agriculture industry by implementing image annotation technique and deep learning approach: a review. *Agriculture*, 12(7), 1033.

[24] Umar, M.A., Sani, B.M., & Suleiman, U. (2022). An Overview of Machine and Deep Learning Technologies Application in Agriculture: Opportunities and Challenges in Nigeria. *SLU Journal of Science and Technology,4(12).*

[25] Hua, L., Zhang, Y., Zhang, Y., & Zhang, D. (2023). Maize pests identification based on improved yolov4-tiny. *Fifth International Conference on Computer Information Science and Artificial Intelligence (CISAI 2022)*,Chongqing, China.

[26] Hui, Y., Zheng, Z., & Sun, C. (2022). E-commerce marketing optimization of agricultural products based on deep learning and data mining. *Computational Intelligence and Neuroscience*, 2022, 1-11.

[27] Li, J. and Wang, X. (2020). Tomato diseases and pests detection based on improved yolo v3 convolutional neural network. *Frontiers in Plant Science*, 11.

[28] Ni, X., Li, C., Jiang, H., & Takeda, F. (2020). Deep learning image segmentation and extraction of blueberry fruit traits associated with harvestability and yield. *Horticulture Research*, 7(1).

[29] Zhang, L., Jia, J., Gui, G., Hao, X.,Gao, W., Wang, M.(2018). Deep learning based improved classification system for designing tomato harvesting robot. *IEEE Access*, 6, 67940,67950.

[30] Guo, C., Zheng, S., Cheng, G., Zhang, Y., Ding, J. (2023). An improved yolo v4 used for grape detection in unstructured environment. *Frontiers in Plant Science*, 14.

[31] Mustafa, H., Umer, M., Hafeez, U., Hameed, A., Sohaib, A., Ullah, S., Madni, H. A. (2022). "Pepper Bell Leaf Disease Detection and Classification Using Optimized Convolutional Neural Network", *Multimedia Tools and Applications*, 82(8), 12065–12080.

[32] Bai, Y., Mao, S., Zhou, J., Zhang, B. (2023)," Clustered Tomato Detection and Picking Point Location Using Machine Learning-Aided İmage Analysis for Automatic Robotic Harvesting", *Precision Agriculture*, 24(2), 727–743.

[33] Gholipoor, M., & Nadali, F. (2019). Fruit yield prediction of pepper using artificial neural network. *Scientia Horticulturae*, 250, 249–253.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

The authors received no financial support for their search, authorship, and/or publication of this article.

Conflicts of Interest

On behalf of all authors, the corresponding author states that there is no conflict of interest.