LONTAR KOMPUTER VOL. 15, NO. 1 APRIL 2024 **p-ISSN 2088-1541**
DOI : 10.24843/LKJITI.2024.v15.i01.p02 **e-ISSN 2541-5832**
Accredited Sinta 2 by RISTEKDIKTI Decree No. 158/E/KPT/2021

Fine-Tuned RetinaNet for Real-Time Lettuce Detection

Eko Wahyu Prasetyo[a1], Hidetaka Nambo[a2]

[a]Artificial Intelligence, Kanazawa University
Kakumamachi, Kanazawa, Ishikawa, Jepang
[1]prasetyoekowahyu7@gmail.com
[2]nambo@blitz.ec.t.kanazawa-u.ac.jp

Abstract

The agricultural industry plays a vital role in the global demand for food production. Along with population growth, there is an increasing need for efficient farming practices that can maximize crop yields. Conventional methods of harvesting lettuce often rely on manual labor, which can be time-consuming, labor-intensive, and prone to human error. These challenges lead to research into automation technology, such as robotics, to improve harvest efficiency and reduce reliance on human intervention. Deep learning-based object detection models have shown impressive success in various computer vision tasks, such as object recognition. RetinaNet model can be trained to identify and localize lettuce accurately. However, the pre-trained models must be fine-tuned to adapt to the specific characteristics of lettuce, such as shape, size, and occlusion, to deploy object recognition models in real-world agricultural scenarios. Fine-tuning the models using lettuce-specific datasets can improve their accuracy and robustness for detecting and localizing lettuce. The data acquired for RetinaNet has the highest accuracy of 0.782, recall of 0.844, f1-score of 0.875, and mAP of 0,962. Metrics evaluate that the higher the score, the better the model performs.

Keywords: *RetinaNet, computer vision, fine-tuned model, agricultural, deep learning*

1. Introduction

The agricultural industry faces the challenge of meeting the increasing global demand for food production while minimizing resources[1]. Conventional methods of lettuce harvesting rely on manual labor[2], which can be time-consuming and prone to human error. Research has focused on automation technology, such as robotics, to improve harvest efficiency and reduce reliance on human intervention. Deep learning-based object detection models have succeeded in various computer vision tasks, including object recognition and localization[3]. However, these models must be fine-tuned to adapt to the specific characteristics of lettuce plants.

Computer vision has many essential tasks, including object detection. Object recognition is crucial, yet the two differ significantly [4]. The primary difference is that object detection focuses on determining the location of objects using bounding boxes[5]. In contrast, object recognition takes an additional step by categorizing and labeling the identified objects based on predefined classes[6]. Both components are crucial in computer vision applications, providing nuanced insights into the visual content under scrutiny.

In deep learning, convolutional neural networks (CNNs) are often used for image data analysis, including tasks like segmentation, object detection, and image classification. Object detection models can generally be categorized into two-stage and one-stage approaches. Two-stage object detection involves using a selective search algorithm to generate region proposals for the target object, which are then classified using a CNN. While achieving high detection accuracy, this approach requires longer training times and reduces detection speed. Researchers have been exploring using AI-based object detection for lettuce, aiming to minimize costs and labor while increasing efficiency in lettuce harvesting. Previous studies have investigated the detection of abnormal leaves in hydroponic lettuce using machine learning algorithms such as Multinomial Logistic Regression (MLR) and Support Vector Machine (SVM)[7]. Additionally, researchers have developed an automatic iceberg lettuce harvesting robot [8], utilizing a vision system for lettuce localization and classification.

This research aims to develop a fine-tuned model that can recognize lettuce in real-time situations. The researchers built a fine-tuned model using several models like RetinaNet, which is a pre-trained model. The results are then analyzed and evaluated, comparing data such as F1 score recall in deep learning. Convolutional neural networks (CNNs) are often used for image data analysis, including tasks like segmentation, object detection, and image classification. Object detection models can generally be categorized into two-stage and one-stage approaches. Two-stage object detection involves using a selective search algorithm to generate region proposals for the target object, which are then classified using a CNN. While achieving high detection accuracy, this approach requires longer training times and reduces detection speed. Researchers have been exploring using AI-based object detection for lettuce, aiming to minimize costs and labor while increasing efficiency in lettuce harvesting. Previous studies have investigated the detection of abnormal leaves in hydroponic lettuce using machine learning algorithms such as Multinomial Logistic Regression (MLR) and Support Vector Machine (SVM). Additionally, classification, Precision, accuracy, and ROCAUC. By incorporating these metrics, the researchers aim to gain valuable insights into the efficiency of the trained model in detecting lettuce instances accurately.

Research related to the article "A Deep Learning Approach for Weed Detection in Lettuce Crops Using Multispectral Images" uses drones and captures RGB images and uses three models such as HOG-SVM, YOLO, and R-CNN using evaluation metrics such as accuracy, precision, and F1-score, by using evaluation metrics we can evaluate the model, whether the model results are efficient[9]. In this research, we add a fine-tuning method that can take time to train the model, and it is hoped that the model can detect objects (lettuce) in real-time.

2. Research Methods

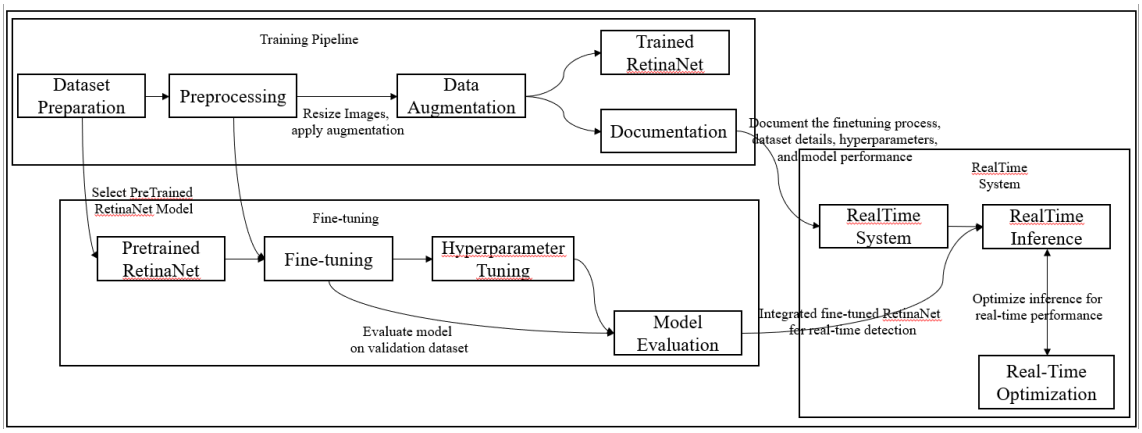

Figure 1. Block Diagram

Figure 1 depicts the workflow for real-time object detection using RetinaNet. The process starts with data preparation (collecting, resizing, and augmenting the dataset). A pre-trained RetinaNet model serves as the foundation. This model is fine-tuned using the prepared data. The tuned model (precision, recall, mAP) is then evaluated on the validation dataset. Next, hyperparameters are adjusted to optimize real-time performance, followed by further optimization for real-time inference. Finally, the tuned and optimized model is integrated into the real-time system, and the entire process is documented. This results in a real-time system with high-accuracy object detection capabilities.

2.1. Fine-tuning model

Fine-tuning in object detection refers to taking a pre-trained model trained on a large dataset and further teaching it on a smaller task-specific dataset[10]. That helps the model adapt to the specific object classes and variations in the target task. The model can learn to detect and localize objects more accurately in a particular job by initializing the model with pre-trained

weights and fine-tuning it on the new dataset. Fine-tune processed by loading a pre-trained model and modifying its output layer to accommodate a custom number of classes. Initializes an optimizer using stochastic gradient descent (SGD) with a learning rate of 0.001, momentum of 0.9, and weight decay of 0.0005. This optimizer is crucial for optimizing the model parameters during the training process on a custom dataset, representing a key element in the fine-tuning procedure. Fine-tuning improves the model's performance compared to training from scratch, especially when the target dataset is small or different from the pre-training dataset.

2.2. Pytorch

In the context of object detection, PyTorch provides the torchvision library, which offers various well-known pre-trained models such as Faster R-CNN[11] and RetinaNet. This library also provides functions for data processing [12], augmentation, and dataset preparation for training and evaluating object detection models [13]. Furthermore, PyTorch facilitates the training and validation process of object detection models by providing classes such as optimizers, loss functions, and schedulers that can be used to optimize and fine-tune models. The primary use case for PyTorch is training machine learning models on GPU.

2.2.1. RetinaNet

RetinaNet is a single, unified network comprising a backbone network and two task-specific subnetworks [14]. It is designed to overcome the limitations of earlier models that need help accurately detecting objects at large and small scales.

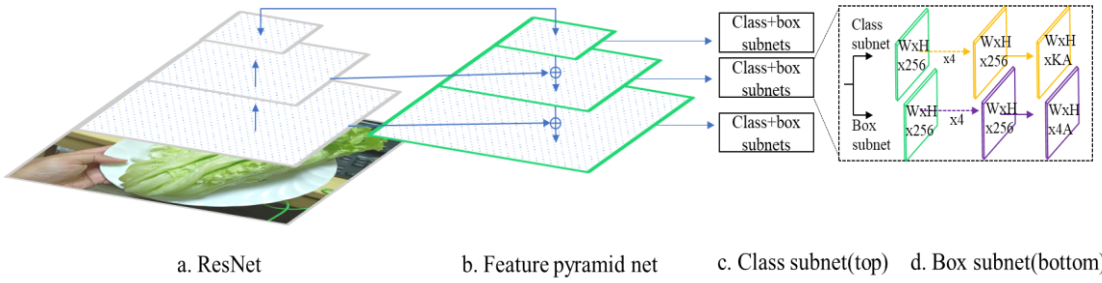

Figure 2. RetinaNet: A Simple One-Stage Object Detector with Focal Loss and FPN Backbone

RetinaNet utilizes a Feature Pyramid Network (FPN) to extract multiple-scale features from a convolutional neural network[15]. Allowing the model to capture fine-grained details of small objects while maintaining contextual information for larger objects [16]. The FPN generates a set of feature maps with different resolutions, forming a pyramid-like structure, as shown in Figure 2.

RetinaNet object detection is performed using a two-branch architecture. The first branch predicts the presence of an object (foreground/background classification) at each spatial location on the feature maps. This branch uses a focal loss to address the issue of class imbalance, where many areas are in the background. The focal loss assigns higher weights to complex examples, which helps to focus the training on challenging samples.

2.3. Evaluation Method

In evaluating object detection methods, many essential ways are used to analyze model performance [17], such as recall accuracy precision IoU and f1 score.

2.3.1. Evaluation Metrics

The evaluation metrics visualize an object detection model's predicted results [9] by comparing them with the actual data labels [18].

Accuracy: Accuracy is a metric that measures the percentage of data the model correctly predicts as positive or negative [19].

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

Recall: (Recall, Sensitivity) measures a model's ability to detect all actual positive instances of an object from the total number of positive samples [20].

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

Precision: Precision measures how accurately the model identifies the actual object instance of all the cases it predicts as an object [21].

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

F1-score: F1-score provides an overview of the balance between detecting as many actual objects as possible (Recall) and providing the correct predictions for these objects (Precision)

$$\text{F1} - \text{score} = 2 * \frac{(\text{Precision} * \text{Recall})}{(\text{Precision}+\text{Recall})} \quad (4)$$

Intersection over Union: Intersection over Union (IoU) is a metric that measures the overlap between predicted and ground truth bounding boxes in object detection tasks[22].

$$\text{IoU} = \frac{\text{TP}}{(\text{TP}+\text{FP}+\text{FN})} \quad (5)$$

2.3.2. Train loss

Train Loss measures how closely the model meets the desired goal during training. This metric calculates the difference between the model's predicted and actual values in the trained data[23]. The main goal is to optimize the training loss so the model can learn from the data and get a smaller value of the training loss the better the model is trained.

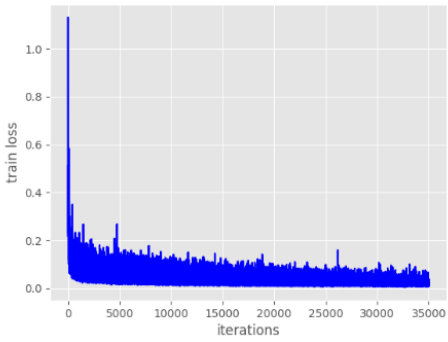

Figure 3. Example of Train Loss

As shown in Figure 3, The decreasing train loss indicates that the model's performance is improving over time during the training process. As the model iteratively learns from the training data, it becomes more effective in making accurate predictions and reducing errors. This decrease in train loss signifies that the model effectively adjusts its parameters to fit the data better, leading to improved performance and increased accuracy. A decreasing train loss is a positive sign, showing that the model is converging towards better results and becoming more proficient in its task.

2.3.3. Validation loss

Validation Loss measures how well a model performs on validation data not used to train it. This helps assess how well the model generalizes and accurately predicts new, unseen data.

Validation loss is also used to monitor overfitting. In this condition, the model 'remembers' too much training data and cannot predict new data well.

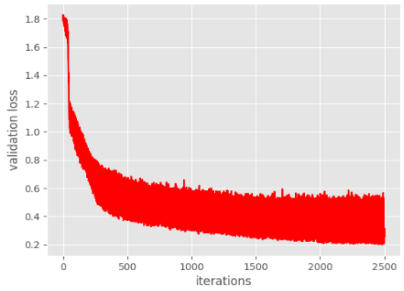

Figure 4. Example of Validation Loss

As shown in Figure 4, The downward trend in validation loss indicates that the model is not overfitting the training data, meaning that it not only "memorizes" the training data but can also find more general patterns and apply them to new data. The slower the validation loss decreases, the better the model can generalize to data that has never been seen before, and this is an indicator of the excellent quality of the model that has been trained.

2.3.4. AUROC

AUROC (Area Under the Receiver Operating Characteristic Curve) is an evaluation metric used to measure the performance of a classification model [24], particularly in object detection problems. The ROC (Receiver Operating Characteristic) curve visualizes the model's performance, distinguishing between positive and negative classes by varying the prediction threshold. A higher AUROC value indicates the model's ability to differentiate between positive and negative categories. AUROC helps compare different models' performance and select the most suitable model for object detection tasks[25].

$$\text{AUROC} = \int \text{TPR}\, d\text{FPR} \tag{6}$$

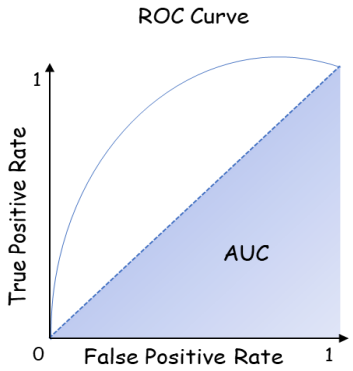

Figure 5. Schematic Diagram of ROC Curve

As shown in Figure 5, The ROC Curve graph is handy for understanding the trade-off between TPR and FPR in various scenarios. It helps choose the optimal threshold for a classification model based on the application's specific needs. The larger the area under the ROC curve, the better the model's performance at discriminating between positive and negative classes.

2.3.5. Inference Object Detection

Inference uses the trained model to detect objects in new images or data by generating predictions about the location and class of things in the image or data [26].

Figure 6. Inference of Lettuce

As shown in Figure 6, the lettuce inference aims to identify lettuce leaves in the image and provide a bounding box for each detected leaf. It can be used in various applications, such as agricultural analysis, vegetable garden surveillance, or any other application that requires identifying and detecting objects in images.

3. Experimental Result and Discussion

This section shows the results of the experiments related to evaluating metrics, train loss, validation loss, and inference performance on images. Precision, recall, and F1 scores are also assessed. Experiments also evaluate train and validation loss to understand the model's learning. Furthermore, the model's performance on the image inference task is evaluated using Intersection over Union (IoU) to measure the extent to which the model predictions match the ground truth.

3.1. Experiments

In this experiment, we used the lettuce datasets obtained by taking photos independently and used them as training data media. In contrast, we used data circulating online, such as the Kaggle and COCO (Common Objects in Context) datasets, for validation.

3.1.1. Dataset

A dataset is a collection of data used to train and test object detection models in object detection. It consists of images or videos containing target objects to be detected and annotations that provide information about the object's position and class in the image. The number of datasets used to train the model is 630 images.

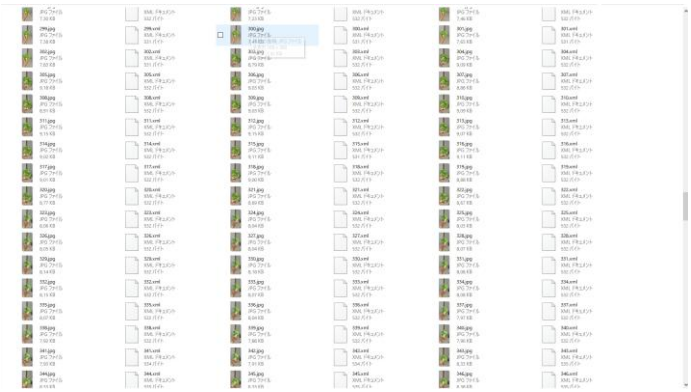

Figure 7. Dataset of Lettuce

As shown in Figure 7, a dataset in the context of object detection refers to a collection of data used for training and testing object detection models. It consists of images containing target objects and annotations that must be detected. In this research, using an average in validation appears to serve the purpose of calculating the average loss at each validation iteration to monitor and evaluate the model's performance in real time. This is noteworthy even though the dataset is evenly split 50:50 between training and validation, emphasizing the importance of continuous performance evaluation during training.

3.1.2. Annotations

Annotations are the information associated with each object in the images or videos. Annotations include bounding box coordinates surrounding the thing, class labels, and other attributes such as pose or object landmarks [27]. Annotations are essential for training object recognition models to recognize and learn the characteristics of target objects.

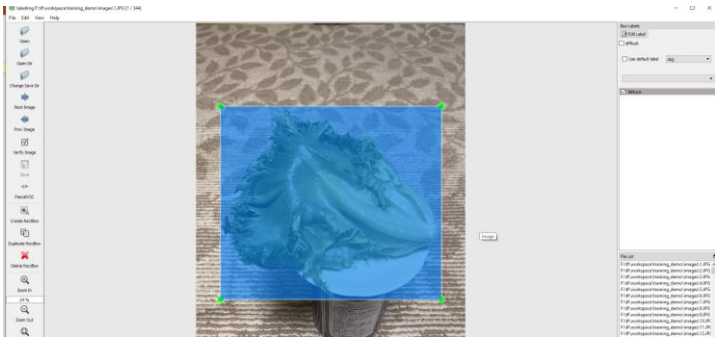

Figure 8. Annotate for lettuce datasets.

As shown in Figure 8. Object detection datasets can be created manually with human annotators who visually mark the objects in the images or videos. There are also popular pre-existing object datasets created and shared by the research community, such as COCO, Pascal VOC (Visual Object Classes) [28], and Open Images. These datasets are often used as benchmarks in the research and development of object detection models. This research uses 630 datasets and using three classes. Such as "background," "lettuce," and "no lettuce." Including the "background" class during the fine-tuning process enables the model to distinguish between the objects it wants to detect and the unrelated background areas [29]. By incorporating the background class, the model becomes proficient in differentiating relevant things from their surrounding context, resulting in more precise and effective object detection.

3.1.3. Develop Environment

The program used in this experiment was written in Python (Version 3.9.16), Matplotlib (Version 3.7.1), and Albumentations (Version 1.3.0), which are commonly used libraries for the development and analysis of machine learning models. PyTorch (Version 2.0.0+cu117) is used for neural network model building and training, while Matplotlib (Version 3.7.1) was used for data visualization and graph generation. We used a Graphics Processing Unit (GeForce RTX 2080 Ti, 16GB VRAM) for rendering. OpenCV-Python (Version 4.7.0.72) is useful in image processing and computer vision applications. Tqdm (Version 4.65.0) is a helpful library for updating the progress bar when iterating in a loop. Pandas (Version 2.0.0), NumPy (Version 1.24.2), and Pillow (Version 9.3.0) are used for data analysis and manipulation in the form of tables or data frames. Scikit-learn (Version 1.2.2) and Scikit-image (Version 0.20.0) are widely used libraries for machine learning and statistical modeling, including training, selecting features, and evaluating models.

3.1.4. Experimental settings

A machine-learning model was constructed using a lettuce image. Acceleration of this training using a graphical processing Unit (GPU). Before training, each print is resized to 512 to make training more accessible; cross-entropy is used as function loss in this research. We used

Stochastic Gradient Descent (SGD)[30] to optimize the model, with a learning rate of 0.001, a momentum of 0.9, and a weight decay 0.0005. We trained the model with 500 epochs and 8 batch sizes for every 100 epochs. We saved the training loss, validation loss, and the model to evaluate whether the model has overfitting, underfitting, or other problems for each epoch.

3.3. Evaluation Result of The RetinaNet Model

In this research, we developed an object detection model using the RetinaNet model to detect lettuce. There are several evaluation matrices that we use in our study this time, such as train loss, validation loss, accuracy, recall, Precision, f1-score, mean average precision (mAP), and IoU. which provides insight into how well our models learn during the training process.

3.3.1. Train Loss

Our observation showed an interesting trend in our train loss during training iterations. If we visualize the training loss against the number of iterations, we see that the train loss graph is blue and shows a specific pattern. The initial train loss is high and unstable at the beginning of the training. However, after a few iterations, the training loss decreases gradually.

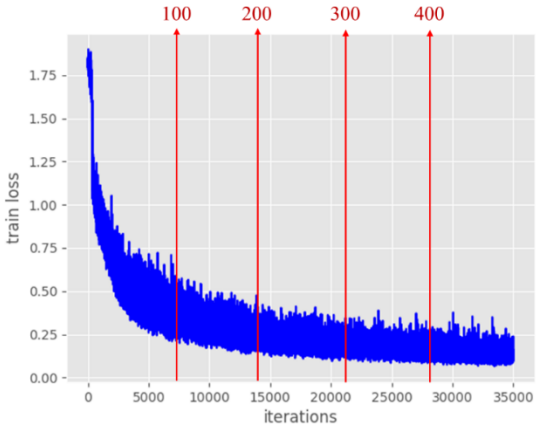

Figure 9. RetinaNet Train Loss for 500 Epochs

As the number of iterations increases, the initially high train loss gradually decreases. This gradual decrease in training loss, as shown in Figure 9, indicates that our model successfully learns essential patterns in the training data. One hundred until 400 epochs in the figure is a marker that train loss divided every 100 epochs until the last of epochs 500. The longer the training process lasts, the better our model can generalize and predict accurately. It went up a bit but then went down regularly. The decreasing train loss graph indicates that our model optimizes the parameters and improves its performance over time.

3.3.2. Validation Loss

Validation Loss is a measure of the extent to which our model can generalize and predict with high accuracy data that has never been seen before.

p-ISSN 2088-1541
e-ISSN 2541-5832

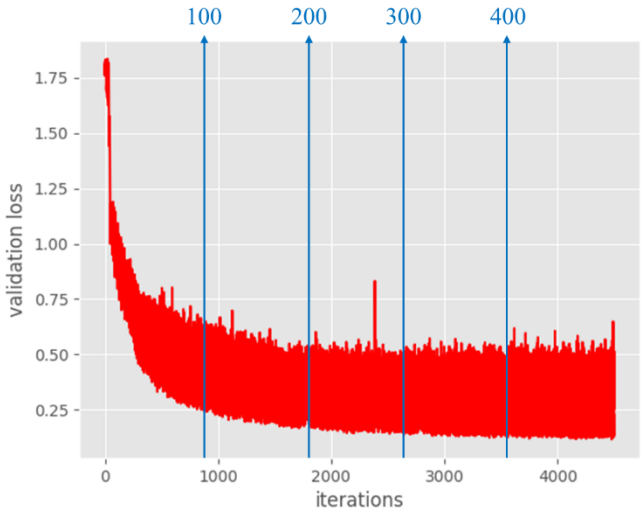

Figure 10. RetinaNet Validation Loss for 500 Epochs

This gradual decrease in validation loss shows that our model can generalize well to validation data. The longer the training process lasts, the better our model can learn the patterns in the validation data. As shown in Figure 10, the trend analysis of the validation loss, we can conclude that our model successfully performs object detection in the lettuce validation data. 100 until 400 epochs in the figure is a marker that validation loss divided every 100 epochs until the last of epochs 500. The initial high validation loss gradually decreases as the number of iterations increases.

3.3.3. Evaluation Metrics

This section provides an in-depth analysis of the RetinaNet model's performance for object lettuce detection and the implications of the evaluation results for the model's success in lettuce object detection over all tasks.

Table 1. Results of Evaluation Metrics

	Test Instances					
Epochs	Accuracy	AUC	Recall	Precision	F1-score	mAP
100	0.645	**0.74**	0.625	**0.946**	0.752	0.95
300	0.683	0.697	**0.917**	0.917	0. 792	0.946
500	**0.782**	0.698	0.844	0.909	**0.875**	**0.962**

Table 2 shows the performance measures of different Epochs models on a single set of 586 instances. In this table evaluation, the epochs of the "500" model were the best performers among the options considered. It achieved an accuracy of 0.782, meaning it could correctly classify 78.2% of the objects in the data set. It also showed a high AUC of 0.698, indicating a solid ability to discriminate between different classes.

In addition, Epochs "500" showed a high recall value of 0.844, indicating that it successfully identified 88.4% of the objects in the data set. The Precision of 0.909 suggests that the model achieved a high percentage of true positives among predicted positive instances. Also, the mAP got a high score of 0.962, which means the model can achieve better accuracy when recognizing the lettuce. Overall, the model achieved an F1 score of 0.875. This indicates a good balance between Precision and recall.

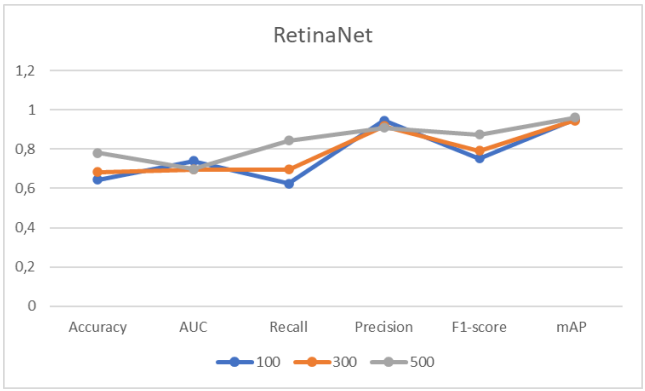

Figure 11. Evaluation Matrix

Figure 11 shows that increasing the number of epochs can improve the performance of the RetinaNet model in the lettuce object detection task. As the number of epochs increases, the model tends to have a better ability to detect lettuce objects with a higher recall rate while maintaining a high precision level. Detection results also improve, as increased F1-score and mAP values indicate. This improvement can lead to better model learning as the number of epochs increases. In this study, the optimal number of epochs is 500, where the model achieves the highest performance with good levels of recall, Precision, and F1-score.

3.3.4. Inference

We perform inference on the object detection model to detect lettuce in images. Inferencing involves using a previously trained model to identify and map the location of lettuce objects in a snap. To test the inference, we use a test data set. This data set consists of images that the model has never seen before. Each photo will be run through an object detection model. The inference results are evaluated using accuracy, Precision, recall, and F1 score metrics. These metrics indicate the model's performance in correctly recognizing and mapping lettuce objects.

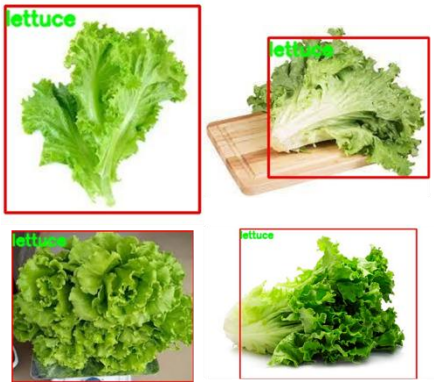

Figure 12. Inference Images

The evaluation process involves testing the model on four different sample images, which are test data the model has never seen before. The evaluation results show in Figure 12. that the object detection model successfully detected lettuce accurately in the four sample images. In each image, the model can identify and map the exact location of the lettuce through the bounding box. In addition to high accuracy, the model can provide detection results with good Precision and recall. Precision measures the model's ability to correctly classify a detection as lettuce, while recall measures the model's ability to identify and detect as much lettuce in an image as possible.

3.3.5. Intersection over Union (IoU)

In this section, we will look at the IoU values of the RetinaNet models that have been trained.

Figure 13. IoU from Model RetinaNet

In Figure 13, the IoU value between the ground truth box and the predictor box is 0.72. The IoU value indicates how much the prediction box overlaps with the ground truth box. Higher IoU values indicate more significant agreement between predicted and ground truth. For example, the IoU value of 0.72 means that the expected box significantly overlaps the ground truth box, indicating that the model successfully recognizes the lettuce object in the image.

4. Conclusion

In evaluating the RetinaNet model for lettuce object detection, various performance metrics were considered across different epochs. Firstly, the model's accuracy demonstrated a clear upward trend, reaching 0.782 at 500 epochs and a high IoU value of 0.72. They indicated improved object detection capabilities with prolonged training. Similarly, the AUC values reflected enhanced discrimination between positive and negative examples as the number of epochs increased. The recall, Precision, F1 score, and mAP consistently improved, underscoring the model's effectiveness in detecting lettuce objects with increasing training iterations.

Furthermore, a detailed analysis of each metric revealed the Precision of the RetinaNet model in classifying lettuce objects with high values across different epochs. The F1 scores, representing the balance between Precision and recall, indicated an improved equilibrium with increased training epochs. The mean Average Precision consistently showed high-quality detection results at every epoch set, emphasizing the robust performance of the model. These findings suggest that extending the training duration positively impacts the RetinaNet model's proficiency in lettuce object detection, providing valuable insights for future refinements and optimizations.

References

[1] D. Mittal, G. Kaur, P. Singh, K. Yadav, and S. A. Ali, “Nanoparticle-Based Sustainable Agriculture and Food Science: Recent Advances and Future Outlook,” *Frontiers in Nanotechnology*, vol. 2. Frontiers Media S.A., Dec. 04, 2020. doi: 10.3389/fnano.2020.579954.

[2] D. I. Pomoni, M. K. Koukou, M. G. Vrachopoulos, and L. Vasiliadis, “A Review of Hydroponics and Conventional Agriculture Based on Energy and Water Consumption, Environmental Impact, and Land Use,” *Energies*, vol. 16, no. 4. MDPI, Feb. 01, 2023. doi: 10.3390/en16041690.

[3] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar, and B. Lee, “A survey of modern deep learning based object detection models,” *Digital Signal Processing: A Review Journal*, vol. 126. Elsevier Inc., Jun. 30, 2022. doi: 10.1016/j.dsp.2022.103514.

[4] B. H. Husain and T. Osawa, “Advancing Fauna Conservation through Machine Learning-Based Spectrogram Recognition: A Study on Object Detection using YOLOv5,” *Jurnal*

Sumberdaya Alam dan Lingkungan, vol. 10, no. 2, pp. 58–68, Aug. 2023, doi: 10.21776/ub.jsal.2023.010.02.2.

[5] W. Syechu, B. B. Nasution, and M. S. Effendi, "CONVOLUTIONAL NEURAL NETWORK OPTIMIZATION FOR DEEP WEEDS," *Sinkron*, vol. 8, no. 1, pp. 268–274, Jan. 2023, doi: 10.33395/sinkron.v8i1.12046.

[6] F. Xiao, H. Wang, Y. Xu, and R. Zhang, "Fruit Detection and Recognition Based on Deep Learning for Automatic Harvesting: An Overview and Review," *Agronomy*, vol. 13, no. 6. MDPI, Jun. 01, 2023. doi: 10.3390/agronomy13061625.

[7] R. Yang *et al.*, "Detection of abnormal hydroponic lettuce leaves based on image processing and machine learning," *Information Processing in Agriculture*, vol. 10, no. 1, pp. 1–10, Mar. 2023, doi: 10.1016/j.inpa.2021.11.001.

[8] S. Birrell, J. Hughes, J. Y. Cai, and F. Iida, "A field-tested robotic harvesting system for iceberg lettuce," *Journal of Field Robotics*, vol. 37, no. 2, pp. 225–245, Mar. 2020, doi: 10.1002/rob.21888.

[9] K. Osorio, A. Puerto, C. Pedraza, D. Jamaica, and L. Rodríguez, "A Deep Learning Approach for Weed Detection in Lettuce Crops Using Multispectral Images," *AgriEngineering*, vol. 2, no. 3, pp. 471–488, Sep. 2020, doi: 10.3390/agriengineering2030032.

[10] M. Li *et al.*, "AlignDet: Aligning Pre-training and Fine-tuning in Object Detection," Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.11077

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," Jun. 2015, [Online]. Available: http://arxiv.org/abs/1506.01497

[12] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," Mar. 2019, [Online]. Available: http://arxiv.org/abs/1903.02428

[13] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.01703

[14] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," Aug. 2017, [Online]. Available: http://arxiv.org/abs/1708.02002

[15] A. G. Hochuli, A. S. Britto, D. A. Saji, J. M. Saavedra, R. Sabourin, and L. S. Oliveira, "A comprehensive comparison of end-to-end approaches for handwritten digit string recognition," *Expert Syst Appl*, vol. 165, Mar. 2021, doi: 10.1016/j.eswa.2020.114196.

[16] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," Dec. 2016, [Online]. Available: http://arxiv.org/abs/1612.03144

[17] A. Badithela, T. Wongpiromsarn, and R. M. Murray, "Evaluation Metrics for Object Detection for Autonomous Systems," Oct. 2022, [Online]. Available: http://arxiv.org/abs/2210.10298

[18] P. Singh, N. Singh, K. K. Singh, and A. Singh, "Chapter 5 - Diagnosing of disease using machine learning," in *Machine Learning and the Internet of Medical Things in Healthcare*, K. K. Singh, M. Elhoseny, A. Singh, and A. A. Elngar, Eds., Academic Press, 2021, pp. 89–111. doi: https://doi.org/10.1016/B978-0-12-821229-5.00003-3.

[19] F. S. Nahm, "Receiver operating characteristic curve: overview and practical use for clinicians," *Korean Journal of Anesthesiology*, vol. 75, no. 1, pp. 25–36, Feb. 2022, doi: 10.4097/kja.21209.

[20] H. R. Sofaer, J. A. Hoeting, and C. S. Jarnevich, "The area under the precision-recall curve as a performance metric for rare binary events," *Methods in Ecology and Evolution*, vol. 10, no. 4, pp. 565–577, Apr. 2019, doi: 10.1111/2041-210X.13140.

[21] N. Tatbul, T. J. Lee, S. Zdonik, M. Alam, and J. Gottschlich, "Precision and Recall for Time Series," Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.03639

[22] D. Zhou *et al.*, "IoU Loss for 2D/3D Object Detection," in *Proceedings - 2019 International Conference on 3D Vision, 3DV 2019*, Institute of Electrical and Electronics Engineers Inc., Sep. 2019, pp. 85–94. doi: 10.1109/3DV.2019.00019.

[23] S. Salman and X. Liu, "Overfitting Mechanism and Avoidance in Deep Neural Networks," Jan. 2019, [Online]. Available: http://arxiv.org/abs/1901.06566

[24] J. Muschelli, "ROC and AUC with a Binary Predictor: a Potentially Misleading Metric," *J Classif*, vol. 37, no. 3, pp. 696–708, Oct. 2020, doi: 10.1007/s00357-019-09345-1.

[25] A. Prosperi, P. A. Korswagen, M. Korff, R. Schipper, and J. G. Rots, "Empirical fragility and ROC curves for masonry buildings subjected to settlements," *Journal of Building Engineering*, vol. 68, Jun. 2023, doi: 10.1016/j.jobe.2023.106094.
[26] E. W. Prasetyo, N. Hidetaka, D. A. Prasetya, W. Dirgantara, and H. F. Windi, "Spatial Based Deep Learning Autonomous Wheel Robot Using CNN," *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi*, vol. 11, no. 3, p. 167, Dec. 2020, doi: 10.24843/lkjiti.2020.v11.i03.p05.
[27] W. Kurdthongmee, K. Suwannarat, and C. Wattanapanich, "A Framework to Estimate the Key Point Within an Object Based on a Deep Learning Object Detection," *HighTech and Innovation Journal*, vol. 4, no. 1, pp. 106–121, Mar. 2023, doi: 10.28991/HIJ-2023-04-01-08.
[28] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *Int J Comput Vis*, vol. 88, no. 2, pp. 303–338, Jun. 2010, doi: 10.1007/s11263-009-0275-4.
[29] T. T. Santos, L. L. de Souza, A. A. dos Santos, and S. Avila, "Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association," *Comput Electron Agric*, vol. 170, Mar. 2020, doi: 10.1016/j.compag.2020.105247.
[30] A. Beznosikov, E. Gorbunov, H. Berard, and M. Diro, "Stochastic Gradient Descent-Ascent: Unified Theory and New Efficient Methods," 2023.