Computers and Electrical Engineering 110 (2023) 108843

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Deep learning and computer vision for leaf miner infestation severity detection on muskmelon (*Cucumis melo*) leaves☆

RajinderKumar M. Math [a,*], Nagaraj V. Dharwadkar [b]

[a] Department of Electronics and Communication Engineering, B.L.D.E. Association's V.P Dr. P.G. Halakatti College of Engineering and Technology, Ashram Road, Vijayapur, Karnataka, India
[b] Department of Computer Science and Engineering, Rajarambapu Institute of Technology, Islampur, Maharashtra, India

ARTICLE INFO

This paper is for CAEE special section VSI-sacs. Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr Zuorro Antonio.

Keywords:
Deep learning
Pest detection
Leaf miner
Computer vision
Convolutional neural networks
RetinaNet
Faster R-CNN
Detectron2

ABSTRACT

Corp protection against pests is known to play a crucial role in developing efficient crop management strategies for Precision Agriculture. A recent estimation by Food and Agriculture Organization (FAO) shows that the perennial loss due to crop pests and diseases amounts to nearly 40% of agricultural crop production at a global level. Identifying pests and diseases and eradicating them without automation is laborious and time-consuming. Automation in detecting and identifying miners at the onset and their eradication is possible using deep learning (DL) and computer vision. This study aims to develop a *Detectron2*-based framework to detect and localize miner infestations on muskmelon leaves by developing a detection model that integrates DL and a computer vision library to enhance detection capabilities. The approach develops, experiments, and compares a region-based detector (Faster Region-based Convolutional Neural networks (R-CNN)) with a region-free (RetinaNet) by training and validating the bounding box annotated custom dataset of leaf miner infected muskmelon leaves imaged using a smartphone camera. The results show that the RetinaNet-based detector outperforms the Faster R-CNN-based detector in recognizing the infestation severity levels, significantly increasing mean average precision and acquiring faster detection speeds.

1. Introduction

Crop pests, insects, and diseases have severely threatened global agricultural fruit and vegetable cultivation. These pests severely impact the crops of poorly managed fields and favour pest growth. Controlling crop pests is possible by using a proper field management strategy, such as proper irrigation management [1], that ensures optimal moisture levels for the crops and avoids pest infestations due to high moisture levels in the soil. The conventional farming practices, the only method to mitigate the effects of unwanted pests/insects was to manually identify the crop pest or diseases and irradicate them using the required quantities of pesticides. The downside of manual detection of pests and diseases is unreliable, time-consuming, and demands highly skilled labour. An automatic system capable of detecting and identifying crop problems can be a feasible solution to overcome the challenges faced by manual detection and identification. The earlier automated systems used image processing techniques to identify and classify crop pests or diseases, but the limited availability of image data was a bottleneck. Technological advancements such as power IoT (PIoT)

☆ This paper was recommended for publication by Associate Editor Dr Zuorro Antonio.

* Corresponding author.
E-mail addresses: ec.math@bldeacet.ac.in (R.M. Math), nagaraj.dharwadkar@ritindia.edu (N.V. Dharwadkar).

https://doi.org/10.1016/j.compeleceng.2023.108843
Received 26 May 2022; Received in revised form 22 June 2023; Accepted 23 June 2023

provide low-latency [2] communication in accessing and processing images from high-resolution digital cameras and affordable low-cost imagining devices such as smartphones, creating an abundance of crop data in the form of publicly available image datasets. These image datasets provide a precise measure of the visual variability in the growth of the crop. Modelling these variabilities using machine learning or deep learning algorithms makes it possible to precisely detect, classify, and localize the pest or diseases of the crops. Identifying crops' pests, insects, and diseases mainly includes classification, object detection, and localization.

Literature shows ample successful implementations of machine learning algorithms in developing reliable systems to identify or classify crop pests or diseases. The earlier performance of image-based classification and identification of crops' diseases, insects, or pests solely relied on skill-based feature extractions. Some of these features are scale-invariant feature transform (SIFT) [3], histogram of oriented gradients (HOG) [4], and speeded-up robust features (SURF) [5]. These image processing techniques for feature extraction usually follow a classifier like Support Vector Machines (SVM) [6], Artificial Neural Networks (ANN) [7], K-Nearest Neighbours (KNN) [8], or Random Forest (RF) [9]. Though these conventional implementations provided the required classification accuracy, they had design complexity and performance saturation limitations with the limited datasets. With the popularity of Convolutional Neural Networks (CNNs), the availability of computational power, and a massive amount of image data, deep learning algorithms soon started to replace conventional machine learning techniques by showing improved performance. The use of CNNs led to performance improvements, better known for their unique capability of automatically extracting the required features to classify the images accurately without requiring any exclusive feature extractor module. A domain expert would otherwise require selecting the tailormade features that accurately define the classes in the input image. Google's online access to low-cost cloud-based computing resources (GPU and TPUs), known as Colaboratory [10], led to researchers' overall development of deep learning algorithms worldwide, especially in agriculture. Some successful implementations of deep learning algorithms in identifying and classifying the diseases, pests, or insects of field crops are [11–14].

With the implementations mentioned above of deep learning models, it is evident that deep learning models are slowly replacing the traditional machine learning models. The transfer learning approach is widely adopted to reduce additional training times and computational resource requirements for building high-performance deep learning models. The transfer learning approach uses the concept where pre-trained deep learning models like (ResNet, Xception, Inception, and VGG) train on a massive dataset of images such as ImageNet (consisting of about 14 million images belonging to 1000 classes). This approach drastically reduces the training time (since the model is pre-trained and does not require training from scratch). The transfer learning technique provides two options for developing a deep learning model. In the first option, the pre-trained model can be used as a feature extractor, while the second option, known as fine-tuning, keeps the initial layers intact. In contrast, fully connected layers replace the number of neurons representing the dataset classes. Some implementations of the transfer learning approach for solving disease, pest, or insect detection problems in agriculture can be found in Refs. [15,16].

To solve more complicated tasks, such as detecting and localizing diseases or pests by an Agribot or robotic arm, merely using a deep learning algorithm might not suffice. These requirements need deep learning models and computer vision libraries capable of identifying the threats (diseases, pests, or insects) and providing the precise localization of threats on the crop parts. These integrated models developed using (deep learning and computer vision libraries) provide improved functionalities that further enhance automation by helping the robots *see* as we humans do and take necessary actions that are more accurate, fast, and reliable. Some research utilizing computer vision libraries and deep learning models that provide solutions to these challenges can be found in Refs. [17,18].

To address these issues of pest detection and localization, a PyTorch-based modular and extensible framework for computer vision algorithms called *Detectron2* (https://github.com/facebookresearch/detectron2) [19] precisely detects and localizes the infestation severity caused by the leaf miner on muskmelon leaves. Two state-of-the-art models (region-based Faster R-CNN and region-free RetinaNet) were tested and evaluated on the commonly used object detection and localization metrics. Modern pest detection models must fulfill two conflicting requirements (higher mAP and higher inference speed) for detection. This study strikes a proper balance between these two requirements by developing a RetinaNet model to provide higher values of both mAP and inference speeds.

The main contributions of this research are:

- Creating a high-quality annotated image dataset with bounding boxes for detecting muskmelon leaf infestation severity in three classes: *low, medium*, and *high*.
- Creating a framework based on *Detectron2* to design, experiment, and evaluate the performance of two state-of-the-art object detection models: Faster R-CNN (a region-based, two-stage detector) and RetinaNet (a region-free, single-stage detector). This framework will be used to precisely detect and localize leaf miner infestation severity on muskmelon leaves using various combinations of COCO baseline models and backbone networks.

The rest of the paper is structured as follows: Section 2 (materials and methods) deals with the methods used in developing the framework for infestation level detection using the *Detectron2* computer vision model library, highlighting dataset preparation, architecture, experimental setup, and some of the model performance evaluation metrics. Section 3 (results and discussion) tabulates the results obtained during the training process, performance evaluation followed by the inference results. Section 4 (Comparison with similar implementations) compares the proposed RetinaNet model's performance with some of the implementations from the literature. Finally, Section 5 (conclusion and future directions) concludes the paper with a conclusion and possible future scope.

2. Material and methods

2.1. Dataset preparation

Creating an image dataset for the detection and identification of leaf miner infestation on muskmelon leaves requires a comprehensive collection of images representing the infestation in various stages. As seen in Fig. 1, our dataset includes three classes of infestation severity and was created through a three-step process. First, we selected muskmelon leaves at various infestation levels. Next, we captured images of the leaves using a smartphone camera in a controlled laboratory setting. Finally, we annotated the images with bounding boxes to indicate the region of interest, or the location of the infestation in the image, before pre-processing. This dataset will be used to develop an integrated deep learning and computer vision model for the detection and localization of pest infestations.

2.2. Data annotation

Muskmelon leaves infected with leaf miner infestations were carefully collected and imaged using a smartphone in a laboratory setting. The laboratory equipment consisted of a Xiaomi Redmi K20 Pro smartphone equipped with a 48 MP Sony IMX586 sensor, a smartphone holder, LED bulbs for uniform lighting, and a glossy black paper background. The images were then downsized from 1846 $\times$ 4000 to 1024 $\times$ 1024 in order to reduce training times without compromising on image details. An open-source image annotation tool called LabelMe was used to provide the necessary annotations after resizing the images. LabelMe was selected for its user-friendly interface, which made it easy for users to annotate their datasets and save the annotations in the desired formats. Given that the problem at hand pertains to detection and classification, bounding box annotation was chosen to categorize the leaf miner infestations into three classes; low, medium, and high, based on the level of infestation. As shown in Fig. 2, the annotation process was carried out on the images using the annotation tool.

In order to use the datasets with *Detectron2*, they must be in the Microsoft Common Objects in Context (MS COCO) format. This includes using the JSON file format for bounding box annotations for each image, which includes important information such as the label (classes), bounding box coordinates, as well as the image's height and width.

As depicted in Fig. 3, the dataset is divided into a train set and a test set using a standard 80–20% split. The train set comprises 400 images, classified into three categories based on infestation severity: *low, medium*, and *high*. Fig. 3 (a) and (b) illustrate the train set and test set instances, respectively. Additionally, the train set is composed of 420 instances that belong to three classes: 165 instances of *low* infestation, 184 instances of *medium* infestation, and 71 instances of *high* infestation. On the other hand, the test set includes 103 instances, with 39 instances of *low* infestation, 40 instances of *medium* infestation, and 24 instances of *high* infestation. It is worth noting that the dataset is unbalanced, making detection and localization more challenging.

2.3. Data augmentation

The *Detectron2* data transformation feature provides a robust set of image augmentation capabilities to expand the dataset and improve model generalizability. The features include options for resizing and randomizing elements such as rotation, brightness, lighting, saturation, contrast, and vertical flipping.

2.4. Proposed Detectron2 framework

The illustration in Fig. 4 shows the utilization of the *Detectron2* framework for the training and evaluation of two cutting-edge detection techniques. These include a region-based, two-stage Faster R-CNN detector [20] and a region-free, single-stage RetinaNet detector [21].

In a laboratory setting, high-resolution images of muskmelon-infested leaves were captured using a smartphone camera. These raw images were then carefully pre-processed, which included steps such as resizing and rescaling, as well as selecting specific channels for

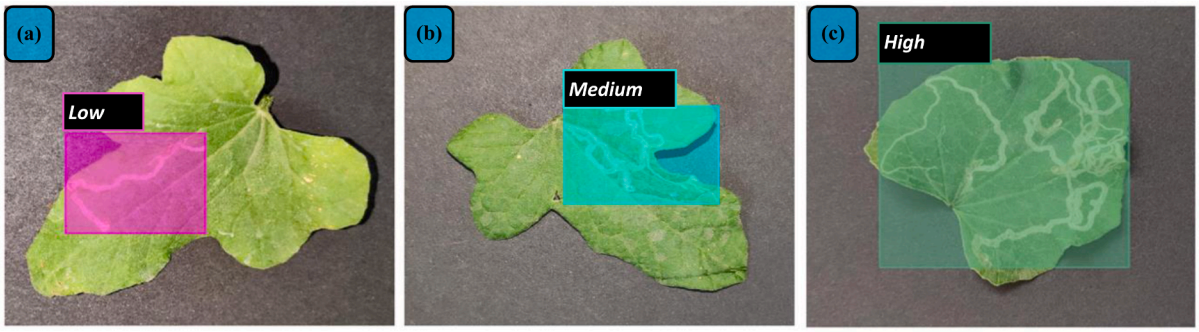

Fig. 1. Bounding box annotations indicating the classes of dataset (a) Low (b) Medium and (c) High.

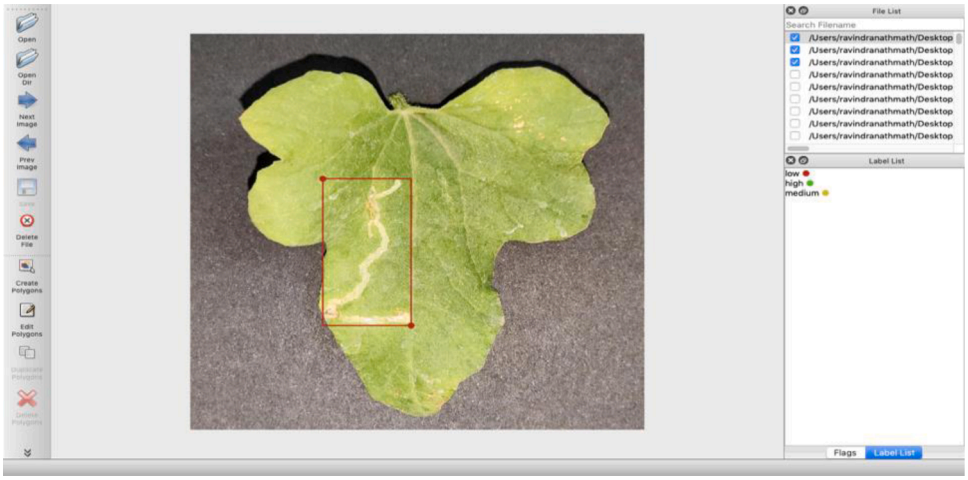

Fig. 2. LabelMe open-source annotation tool showing bounding box annotations.

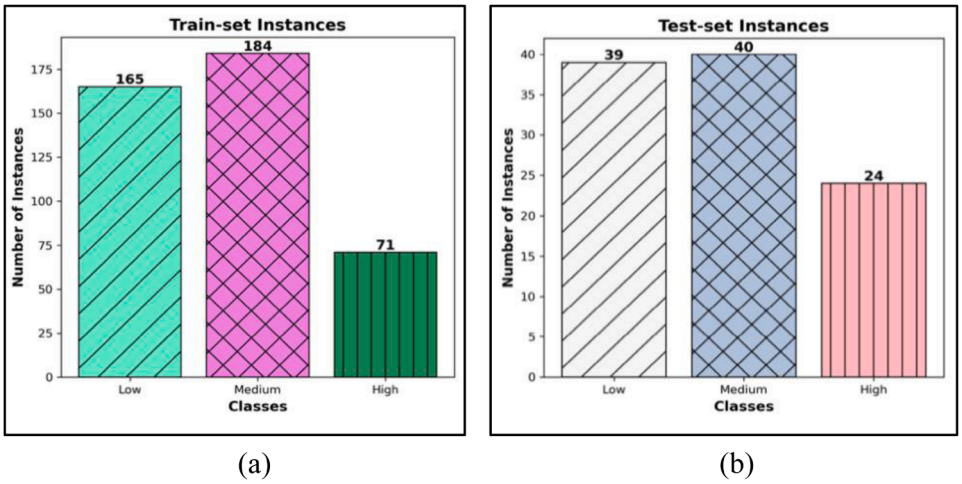

Fig. 3. Dataset division with number of Instances indicating (a) Train and (b)Test sets.

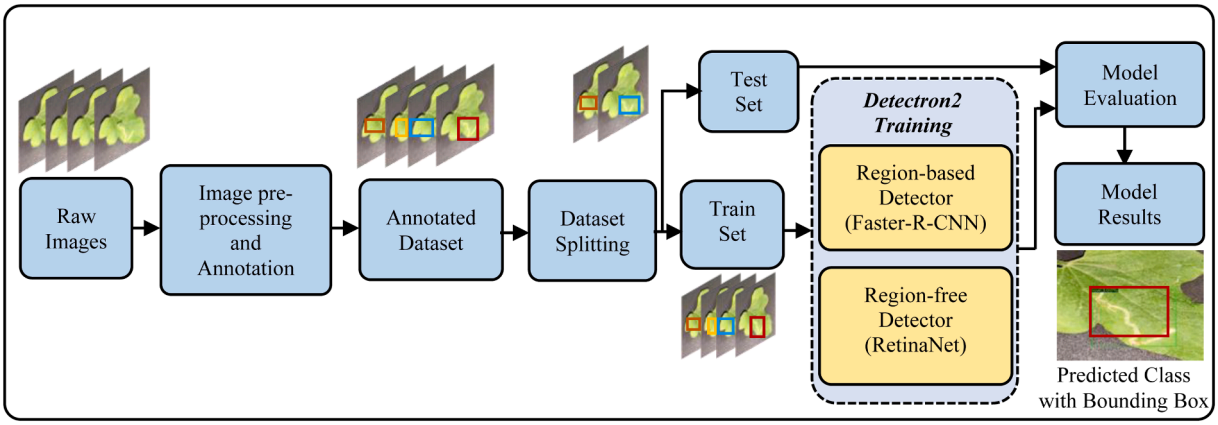

Fig. 4. Architectural framework for training and evaluation of proposed models.

analysis. After pre-processing, the images were manually annotated with bounding boxes in the COCO format and saved as a JSON file for further analysis.

A common practice in machine learning is to divide a dataset into a train set and a test set. In this case, the dataset was split into an 80–20 ratio, with 400 images allocated for training and 100 images reserved for testing. The models were trained for 2000 epochs using the train set, and their performance was evaluated by applying them to the test set. The models output predictions of infestation severity, along with a confidence level and a bounding box indicating the location of the infestation in the image.

The internal design of Faster R-CNN involves two stages. The first stage utilizes a CNN backbone to generate feature maps, while the region proposal network (RPN) generates region proposals by operating on these feature maps. These proposals are then passed through a region of interest (RoI) layer, resulting in fixed-size regions. In the second stage, a pair of Fully Connected Layers (FCL) are applied after the RoI pooling layer, providing both the classification and location of leaf miner infestations. The model produces two outputs: the first indicates the probability of one of the classes (*low, medium*, and *high*) and the second provides the predicted bounding box of the infestation magnitude. The *Detectron2* offers various backbone networks such as ResNet (50, 101, 152), ResNeXt (50, 101, 152), VGG-16, and Feature Pyramid Network (FPN) with ResNet (50, 101, 152), and ResNeXt (50, 101, 152). In this particular implementation, Faster R-CNN utilized three ResNet backbones with FPN, C4, and DC5, and one ResNeXt with FPN.

The RetinaNet utilizes a powerful combination of a ResNet backbone network and a Feature Pyramid Network (FPN) to extract features from the input image at multiple scales. This is then followed by two subnetworks that work together to provide bounding box regression and output class prediction. The classification subnetwork uses a Fully Convolutional Network (FCN) to predict the probability of the presence of an object at all possible positions in the image. Meanwhile, the box regression subnetwork also utilizes an FCN to provide the regression of each anchor box and reduce the offset against the ground truth object. The RetinaNet model is trained using two essential baselines, which are the ResNet-based (R50-FPN and R101-FPN).

2.5. Experimental environment setup

Google's online Python-based platform, called Colaboratory or Colab, provided the platform for training and evaluating the developed models. Table 1 represents the detailed environment setup using *Detectron2*.

The *Colab* provides easy training and testing of machine learning algorithms on the go. Apart from giving free computing resources like GPU and TPUs, it offers some of the widely used pre-installed libraries necessary for developing machine learning (ML) or DL models.

2.6. Evaluation metrics

The proposed model's performance was evaluated using standard object detection metrics, including average precision (AP), mean average precision (mAP), average recall (AR), and F1-score. These metrics were calculated using intersection over union (IoU) threshold values ranging from 0.5 to 0.95. This approach is commonly used in object detection problems as it provides a clear measure of the model's accuracy. The *Detectron2* COCO evaluator also utilizes IoU threshold values of (0.5), (0.75), and (0.5:0.95) to evaluate parameters such as average precision and recall. The calculation of IoU values is performed using Eq. (1) for a specific threshold value.

$$IoU = \frac{A_O}{A_U} \tag{1}$$

where, A_O represents the overlapping area between the predicted bounding boxes and ground truth while A_U indicates area representing the union between them.

In order to accurately determine precision and recall, we must gather the necessary data points such as true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). These values are determined by measuring the Intersection over Union (IoU) between the predicted bounding box and the ground truth bounding box. The evaluation process typically uses a threshold of 0.5. When the IoU is greater than 0.5, the detection is considered correct and referred to as a true positive (TP). Conversely, when the IoU is less than 0.5, the detection is considered incorrect and referred to as a false positive (FP). Using these values, precision (P) and recall (R) can be calculated using the formulas depicted in Eqs. (2) and (3), respectively.

Table 1
Details of *Detectron2*'s experimental setup.

Parameters/libraries	Details/Version
Python	3.7.13
NumPy	1.21.5
Detectron2	0.6
Compiler	GCC 7.3
CUDA compiler	CUDA 11.1
PyTorch	1.10.0+cu111
GPU 0	Tesla P100-PCIE-16GB
Pillow	7.1.2

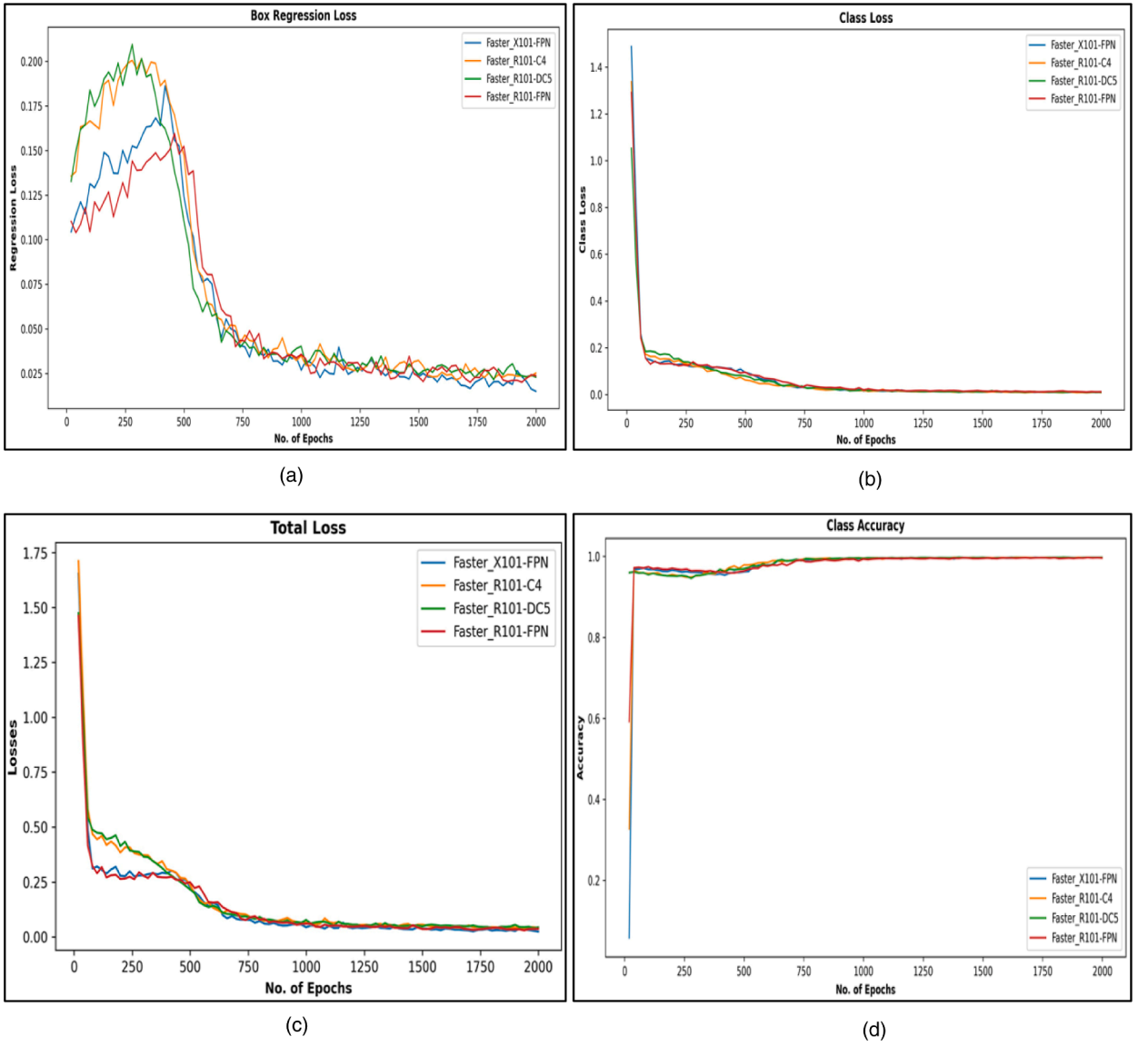

Fig. 5. Training results for Faster R-CNN indicating (a) Box regression loss (b) Classification loss (c) Total loss and (d) Class accuracy.

$$Precision\ (P) = \frac{TP}{(TP + FP)} \tag{2}$$

$$Recall\ (R) = \frac{TP}{(TP + FN)} \tag{3}$$

By using P and R values, the F1-score is calculated by Eq. (4)

$$F1 - score\ = 2\ \times \frac{P * R}{(P + R)} \tag{4}$$

The F1-score is a metric that strikes a balance between precision and recall, making it an effective measure of a model's detection capabilities. A higher F1-score indicates a higher level of precision and recall, which in turn implies that the model is performing well. Additionally, the area under the precision-recall curve, as represented by Eq. (5), can also provide valuable insight into a model's performance through the calculation of Average Precision values.

$$AP\ = \int_0^1 P(R)\mathrm{d}R \tag{5}$$

The mean average precision (MAP) is a commonly used metric for evaluating the performance of object detection models. It is calculated by averaging the Average Precision (AP) scores for each class in the dataset, as outlined in Eq. (6).

$$mAP = \frac{1}{3}\sum_{i=1}^{3} AP_i \tag{6}$$

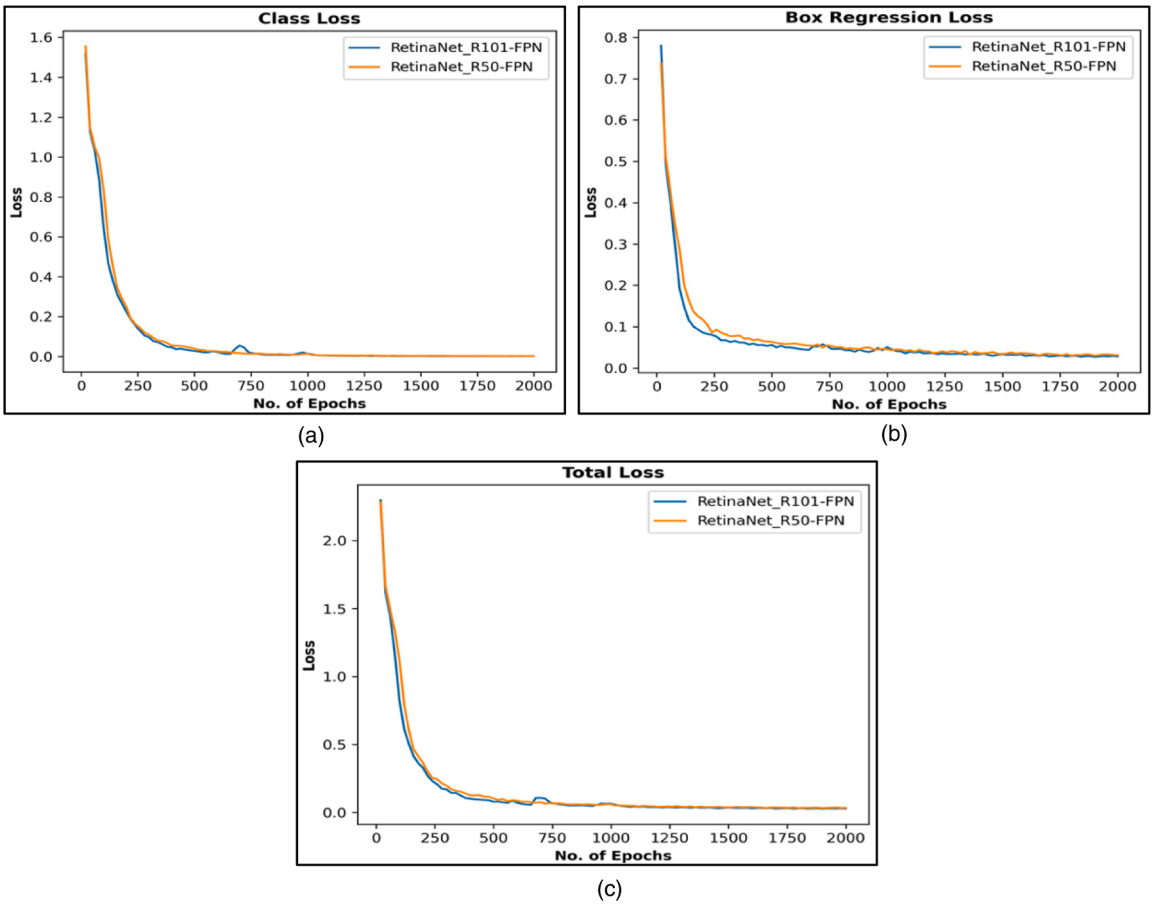

Fig. 6. Training results for RetinaNet model indicating (a) Class loss (b) Box regression loss and (c) Total loss.

3. Results and discussion

The two models' successful training and testing yielded the results using the metrics described in Section 2.6. The following subsections deals with the visualization of the results obtained during the training process. Further, the models evaluated with the standard evaluation metrics consist of mAP, AR, and the F1-score.

3.1. Training results

After undergoing training and evaluation, two object detection models, Faster R-CNN and RetinaNet, displayed exceptional performance. These models were trained on a train set for 2000 epochs and evaluated on a test-set using *Detectron2's* COCO evaluator. The Faster R-CNN model utilized four COCO baselines with different backbone combinations, including X101-FPN (ResNeXt + FPN backbone), R101-FPN (ResNet + FPN), R101-C4 (ResNet conv4 backbone with conv5 head), and R101-DC5 (Dilated-C5). Meanwhile, the RetinaNet model employed the COCO baseline as RetinaNet R50-FPN and RetinaNet R101-FPN. It is worth noting that the Faster R-CNN and RetinaNet models underwent separate training and evaluation processes. The results obtained after training *Detectron2* with Faster R-CNN for 2000 epochs are illustrated in Fig. 5. The bounding box regression loss for 2000 epochs is represented by Fig. 5(a).

As shown in the graph, the Faster R-CNN model utilizing the X101-FPN baseline achieved the highest level of performance with a low box regression loss of 0.0125. The model's class loss evaluation is affected by the presence of a background class, as demonstrated in Fig. 5(b). There is minimal variation in the results obtained from different baselines. Fig. 5(c) illustrates the total loss for the Faster R-CNN model, which combines both box regression and class loss. Furthermore, Fig. 5(d) presents the accuracy achieved by the various baseline models. The comparison of the Faster R-CNN model with different baseline models suggests that the X101-FPN outperforms the others when considering both model losses and accuracy. Additionally, we trained the ResNet model for 2000 epochs using two COCO baseline models, RetinaNet R50-FPN and R101-FPN. Fig. 6 presents plots of the various losses experienced during the training process.

As illustrated in Fig. 6(a), the class loss fluctuates as the number of epochs increases, while Fig. 6(b) displays the variation in box regression loss for the two baselines. Additionally, Fig. 6(c) illustrates the total loss over the course of the epochs. Despite the minor variations in loss performance between the models, it becomes apparent when assessing other evaluation metrics such as mAP and R. A key feature of the RetinaNet-based model is its ability to address class imbalance through the use of focal loss instead of the traditional cross-entropy loss commonly used in object detection.

3.2. Performance evaluation

3.2.1. Mean average precision (mAP)

Following a successful training phase, the performance of the Faster R-CNN and RetinaNet models was evaluated using various baselines. The evaluation process utilized average precision (AP) as the primary metric and Table 2 illustrates the results for different threshold values of IoU. For example, the AP values obtained with an IoU threshold of 0.5 are represented as $AP_{IoU=0.5}$, and similarly, $AP_{IoU=0.75}$ indicates a threshold of 0.75.

The evaluation of average precision (AP) is conducted using a range of IoU threshold values, starting at 0.5 and increasing by 0.05 until 0.95. The AP_M represents the AP values for medium-sized instances, defined as those with an area between 32^2 and 96^2, while AP_L represents the AP values for larger instances, with an area greater than 96^2.

As shown in Table 2, the RetinaNet model utilizing the R101-FPN baseline demonstrated the highest level of performance with an average precision (AP) of 92.36%. In the *Detectron2* framework, the AP metric is referred to as the mean average precision (mAP) when evaluating on the COCO dataset. Therefore, the results obtained for AP can be considered equivalent to mAP.

The mAP values plotted for various baselines in Fig. 7 indicate that the RetinaNet R101-FPN baseline model attained the highest mAP value of 92.36%. Apart from the comprehensive model-wise evaluation, the category-wise assessment provided insights into the model's capability to understand the distinction between the severity classes of muskmelon.

The three infestation severity levels correspond to *low, medium,* and *high*. The class-wise results are tabulated for mAP, as shown in Table 3. The RetinaNet R101-FPN obtained the highest value of mAP, equal to 100 for the class representing *high* infestation levels by leaf miners. In contrast, the *low* and *medium* classes yielded an mAP value of 92.06% and 85.02%, respectively. These results are better than the recent research [22] that uses a multi-scale feature approach for pest detection.

Table 2
Evaluation of average precision with various baselines.

Sl. No.	COCO Baselines	Average Precision (AP) in (%)				
		AP	$AP_{IoU=0.5}$	$AP_{IoU=0.75}$	AP_M	AP_L
1.	Faster R-CNN X101-FPN	87.60	98.90	96.02	65.15	88.42
2.	Faster R-CNN R101-C4	89.03	98.60	95.89	60.10	89.67
3.	Faster R-CNN R101-DC5	87.08	98.57	97.23	67.57	87.70
4.	Faster R-CNN R101-FPN	88.27	98.55	97.55	75.05	88.55
5.	**RetinaNet R101-FPN**	**92.36**	**96.97**	**95.42**	**67.57**	**93.02**
6.	RetinaNet R50-FPN	92.17	97.16	95.78	67.57	92.68

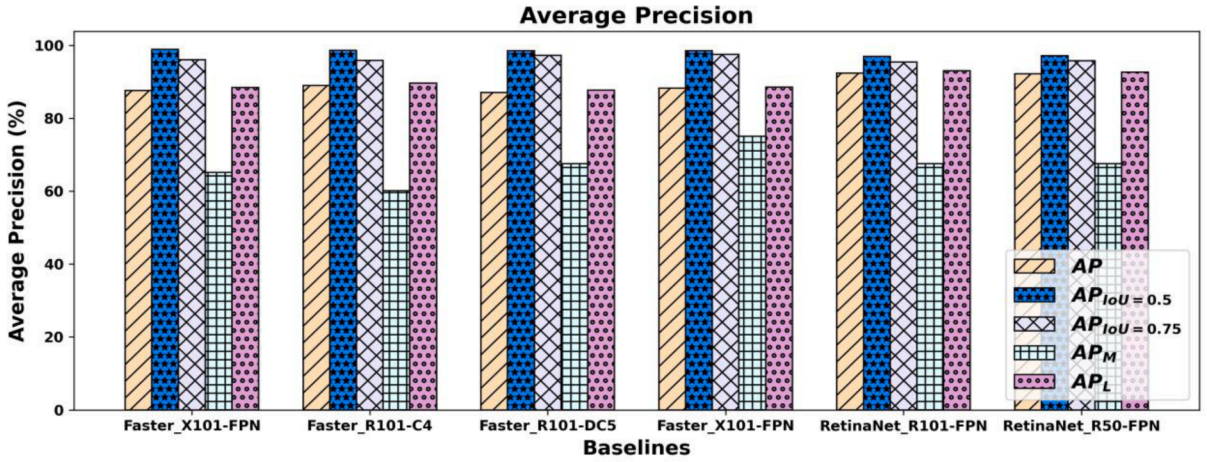

Fig. 7. Mean average precision values obtained for Faster R-CNN and RetinaNet models.

Table 3
Category-wise evaluation of various baselines for Faster R-CNN and RetinaNet models.

SL. No.	COCO Baselines	Class-wise Mean Average Precision in (%)		
		Low	Medium	High
1.	Faster R-CNN X101-FPN	85.70	84.92	92.17
2.	Faster R-CNN R101-C4	89.30	85.51	92.28
3.	Faster R-CNN R101-DC5	87.46	84.62	89.15
4.	Faster R-CNN R101-FPN	90.06	83.00	91.74
5.	**RetinaNet R101-FPN**	**92.06**	**85.02**	**100**
6.	RetinaNet R50-FPN	91.98	86.18	98.35

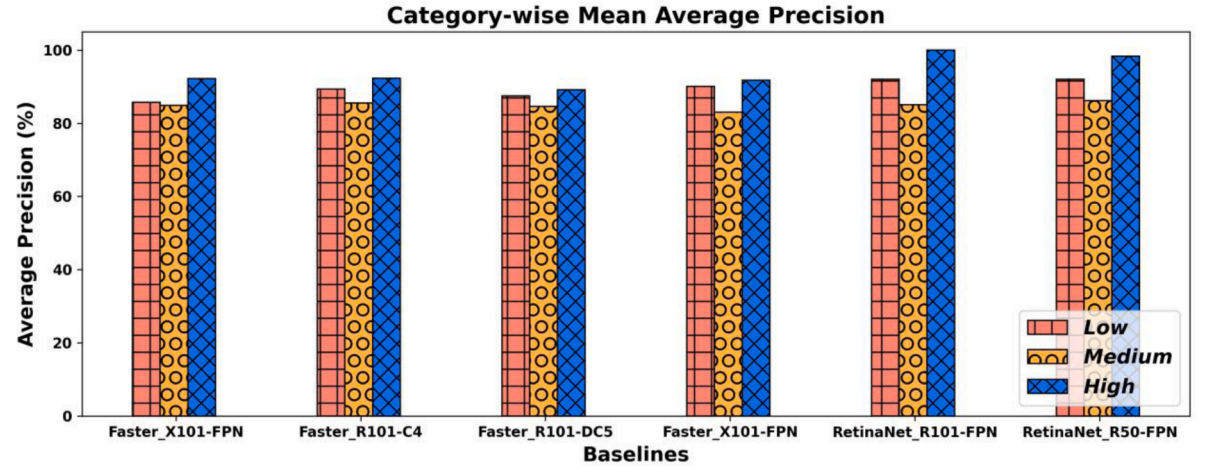

Fig. 8. Category-wise mean average precision values for various baselines for Faster R-CNN and RetinaNet models.

As a result of analysing the overall model and category-wise evaluation of mAP for both Faster R-CNN and RetinaNet models, it can be concluded that the RetinaNet model, which is region-free and has an R101-FPN baseline, performed better than the region-based Faster R-CNN model. This can also be visualized in the category-wise variation of mAP illustrated in Fig. 8.

3.2.2. Average recall (AR)

The second performance metric evaluated for the models used in the framework is the average recall (AR). The values of AR for the COCO baselines are presented in Table 4. As shown in bold, the RetinaNet R-50-FPN achieved the highest results.

In Table 4, the AR values are presented for different maximum detection per image scenarios. $AR_{max=1}$ represents the AR values

Table 4
Evaluation of average recall with various baselines for Faster R-CNN and RetinaNet models.

Sl. No.	COCO Baselines	Average Recall (AR) in (%)				
		$AR_{max=1}$	$AR_{max=10}$	$AR_{max=100}$	AR_M	AR_L
1.	Faster R-CNN X101-FPN	89.00	91.20	91.20	65.00	91.60
2.	Faster R-CNN R101-C4	90.40	92.40	92.40	70.00	92.80
3.	Faster R-CNN R101-DC5	88.40	90.80	90.80	75.00	91.10
4.	Faster R-CNN R101-FPN	89.30	91.60	91.60	80.00	91.80
5.	RetinaNet R101-FPN	92.10	94.40	94.40	75.00	94.70
6.	**RetinaNet R50-FPN**	**92.70**	**94.80**	**94.80**	**75.00**	**95.20**

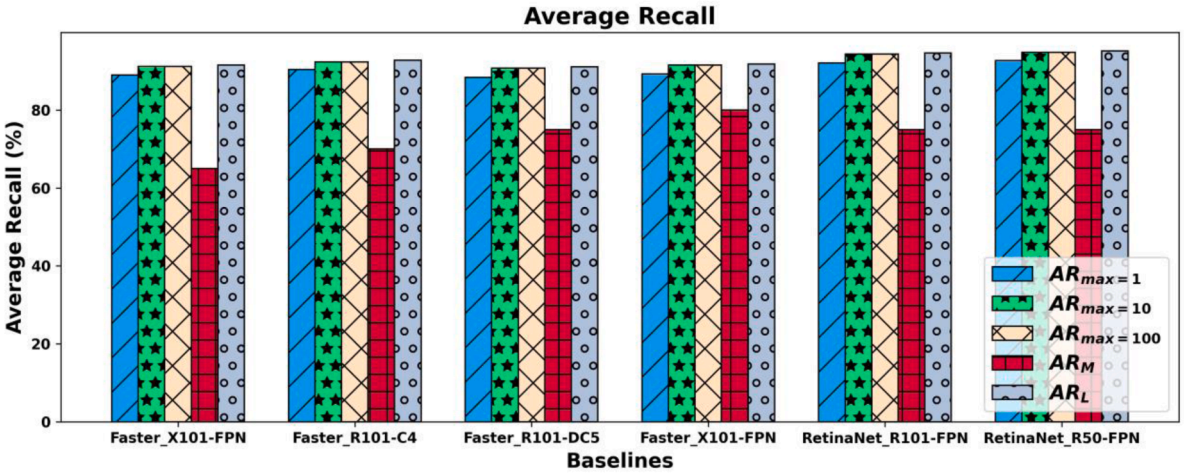

Fig. 9. Average recall values for various baselines for Faster R-CNN and RetinaNet models.

obtained when there is a maximum of 1 detection per image, while $AR_{max=10}$ and $AR_{max=100}$ indicate the maximum detections of 10 and 100 per image, respectively. Additionally, the AR_M represents the AR across scales with the area ($32^2 <$ area $< 96^2$) and AR_L with the area (area $> 96^2$). The RetinaNet model yielded the best performance for AR using the R50-FPN baseline, achieving an $AR_{max=1}$ value of 92.70%. As shown in Fig. 9, the AR values plotted for various baselines demonstrate that the RetinaNet R50-FPN baseline model achieves the highest value among all models.

Based on the evaluation of the mean average precision (mAP) metric, it can be concluded that the region-free RetinaNet model (utilizing the R50-FPN baseline) outperformed the region-based Faster R-CNN model.

3.2.3. F1-score

Upon evaluating the mean average precision and average recall for the models, we calculated the F1-score for various baselines within the framework. As shown in Table 5, the values for AP, AR, and F1-score are presented in a tabular format. Notably, the F1-score values are derived from the AP and AR values as represented in the table.

Thus, based on the overall model's evaluation for the F1-score, it can be observed that the region-free RetinaNet model (with R50-FPN and R101-FPN baselines) outperformed the region-based Faster R-CNN model. The higher values of AP and AR ensured higher values of the F1-score, as indicated in Fig. 10.

3.3. Inference results

The trained RetinaNet model (with R101-FPN baseline) was used to obtain the inference on the test set with a test score threshold of 0.7. The inference results in Fig. 11 show the predicted bounding box with class and corresponding confidence levels. The inference results indicate the two aspects of an object detection model.

The mAP indicates the model's capability of accurately detecting and classifying the objects (leaf miner infestations in this case), while the inference time measures the model's quickness in providing the result. Table 6 depicts the inference times obtained for various COCO baseline models with different image dimensions. As highlighted in bold, the RetinaNet model (R50-FPN) provides the best performance with the lowest inference time across all the input image dimensions. The fastest inference time recorded for the lowest input dimension of 512 × 512 was 0.20 s, while for the highest dimension of 1024 × 1024, the model attained an inference time of 0.31seconds.

The graphical representation of the inference time for various baseline models is as shown in Fig. 12.

Table 5
. F1-score with average precision and recall for various baselines in (%).

SL. No.	COCO Baselines	Average Precision	Average Recall	F1-score
1.	Faster R-CNN X101-FPN	87.60	89.00	88.29
2.	Faster R-CNN R101-C4	89.03	92.40	90.67
3.	Faster R-CNN R101-DC5	87.08	90.80	88.90
4.	Faster R-CNN R101-FPN	88.27	91.60	89.90
5.	**RetinaNet R101-FPN**	**92.36**	**94.40**	**93.37**
6.	**RetinaNet R50-FPN**	**92.17**	**94.80**	**93.47**

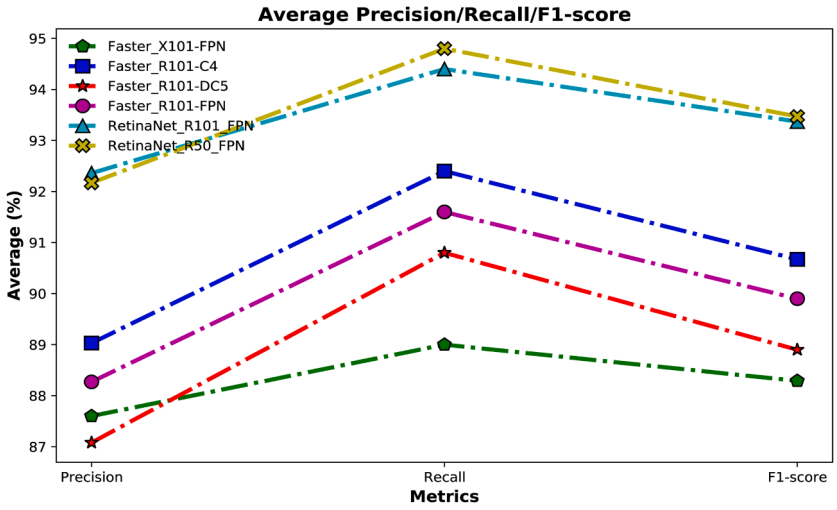

Fig. 10. Metric values attained by various baselines of Faster R-CNN and RetinaNet models.

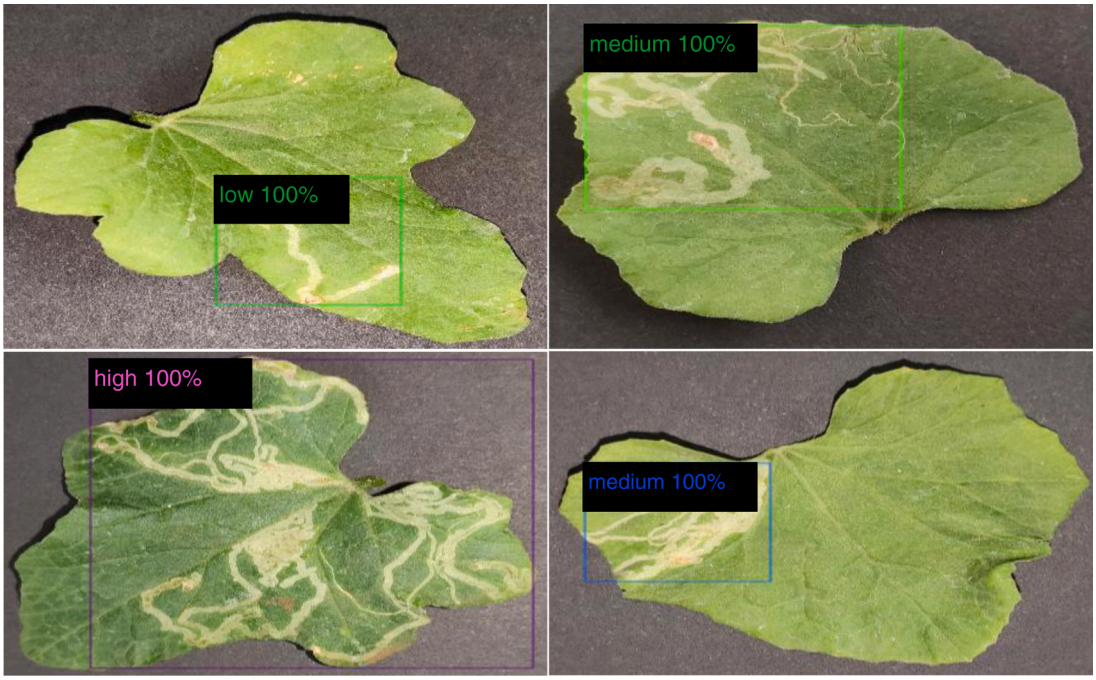

Fig. 11. Sample of predicted class and bounding box images with accompanying confidence levels for each inference.

Table 6
Comparison of inference times for various baselines with different input image dimensions.

SL. No.	COCO Baselines	Inference time (sec)			
		512 × 512	640 × 640	800 × 800	1024 × 1024
1.	Faster R-CNN X101-FPN	0.43	0.46	0.47	0.54
2.	Faster R-CNN R101-C4	0.37	0.40	0.43	0.51
3.	Faster R-CNN R101-DC5	0.28	0.29	0.32	0.42
4.	Faster R-CNN R101-FPN	0.22	0.26	0.27	0.34
5.	RetinaNet R101-FPN	0.21	0.25	0.26	0.33
6.	**RetinaNet R50-FPN**	**0.20**	**0.23**	**0.24**	**0.31**

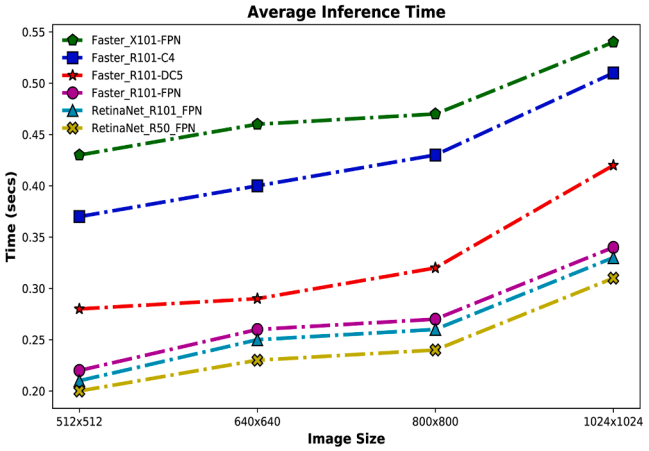

Fig. 12. Average inference times attained by various baseline models for different image dimensions.

4. Comparison with similar implementations

This research presents a framework for detecting and locating leaf miner infestations on muskmelon leaves using popular object detection models in the *Detectron2* framework. The Faster R-CNN and RetinaNet detection models were trained and evaluated, with the results indicating that the RetinaNet model with baselines of R50-FPN and R101-FPN performed better in terms of accuracy (mAP) and inference speeds, making it suitable for real-time implementations. To the best of our knowledge, this is the first study to detect leaf miner infestations on muskmelon leaves using object detection models. A comparison with similar implementations involving different crop pests and diseases from the literature was also conducted. The comparison parameters included the identified problem, crop involved (vegetables, pulses, or fruits), infestation type (disease, pest, or insect), method (detection algorithm), and the mAP/Accuracy

Table 7
Comparison of the proposed model with some of the similar implementations in the literature.

SL. No.	Ref.	Identified Problem	Pest	Crops/ vegetables/fruits	Method	mAP/Accuracy (%)
1	[23]	Real-time Identification of Diseases and Pests	Various tomato pests	Tomato	R-FCN†	88.20
2	[24]	Multi-category pest detection in agriculture	Variety of pests that harm tomato plants	Various	AF-R-CNN‡	56.40
3	[25]	Diseases and pest detection	Banana diseases and pests	Banana	(Faster R-CNN and SSD)	Faster R-CNN InceptionV2 (88.92) Faster R-CNN ResNet50 (88.83) SSD MobileNetV1 (81.78)
4	[26]	Detection of aphids	Aphids	Wheat	HOG+SVM	86.81
5	[27]	Identification of *Tuta absoluta*	*Tuta absoluta*	Tomato	DCNN	91.90
6	Proposed method	Leaf miner infestation Identification	Leaf miner	Muskmelon	Faster R-CNN and RetinaNet	Faster R-CNN (89.03) RetinaNet (92.36)

† - Region-based Fully Convolutional Network.
‡ - Anchor-Free Region-based CNN.

attained by the models. As shown in Table 7, the proposed method outperforms other state-of-the-art methods from literature, as highlighted in bold.

5. Conclusion and future directions

Our research resulted in the creation of an integrated framework that combines deep learning (DL) and computer vision algorithms to develop and evaluate state-of-the-art models for detecting and pinpointing leaf miner infestations on muskmelon leaves. We trained and tested these models using a custom dataset, which was compiled by capturing images of a self-maintained terrace garden using a smartphone camera in a laboratory setting. The images were manually annotated using the LabelMe open-source tool and saved in JSON format, with bounding boxes indicating the severity of infestation in three classes (*low, medium*, and *high*). The results showed that the region-free RetinaNet model outperformed the region-based Faster R-CNN model. Specifically, the RetinaNet model with a ResNet101 backbone and FPN achieved the highest mean average precision of 92.36%, while the ResNet50 backbone with FPN obtained an average recall value of 92.70%. These results demonstrate the RetinaNet model's ability to accurately classify and locate leaf miner infestations with speed. Our proposed method strikes the ideal balance between mean average precision and inference times, which is crucial for detecting small objects such as pests and diseases. Integrating DL and computer vision algorithms leads to efficient and precise detection and localization of crop pests. Early detection of pest infestation minimizes the spread to surrounding crops and reduces financial losses for farmers. The future improvement involves the expansion of the dataset (possibly to detect pests on multiple crops) and testing the developed model under deployment scenarios for real-time validation and further model performance improvements.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Math RKM, Dharwadkar NV. An intelligent irrigation scheduling and monitoring system for precision agriculture application. Int J Agric Environ Inf Syst 2020; 11(4):1–24. https://doi.org/10.4018/IJAEIS.2020100101.
[2] Zhang S, Wang Z, Zhou Z, Wang Y, Zhang H, Zhang G, Ding H, Mumtaz S, Guizani Mohsen. Blockchain and federated deep reinforcement learning based secure cloud-edge-end collaboration in power IoT. IEEE Wirel Commun 2022;29(2):84–91. https://doi.org/10.1109/MWC.010.2100491.
[3] Lowe DG. Distinctive image features from scale-invariant keypoints. Int J Comput Vis 2004;60(2):91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94.
[4] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: Proceedings of the 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), San Diego, CA, USA. 1; 2005. p. 886–93. https://doi.org/10.1109/CVPR.2005.177.
[5] Bay H, Ess A, Tuytelaars T, Van Gool L. Speeded-up robust features (surf. Comput Vis Image Understand 2008;110(3):346–59. https://doi.org/10.1016/j.cviu.2007.09.014.
[6] Mokhtar U, El Bendary N, Hassenian AE, Emary E, Mahmoud MA, Hefny H, Tolba MF. SVM-based detection of tomato leaves diseases. Adv Intell Syst Comput 2015:641–52. https://doi.org/10.1007/978-3-319-11310-4_55.
[7] Espinoza K, Valera DL, Torres JA, López A, Molina-Aiz FD. Combination of image processing and artificial neural networks as a novel approach for the identification of Bemisia tabaci and Frankliniella occidentalis on sticky traps in greenhouse agriculture. Comput Electron Agric 2016;127:495–505. https://doi.org/10.1016/j.compag.2016.07.008.
[8] Singh J, Kaur H. Plant disease detection based on region-based segmentation and KNN classifier. In: Proceedings of the international conference on ISMAC in computational vision and bioengineering 2018 (ISMAC-CVB); 2019. p. 1667–75. https://doi.org/10.1007/978-3-030-00665-5_154.
[9] Wójtowicz A, Piekarczyk J, Czernecki B, Ratajkiewicz H. A random forest model for the classification of wheat and rye leaf rust symptoms based on pure spectra at Leaf Scale. J Photochem Photobiol B Biol 2021;223:112278. https://doi.org/10.1016/j.jphotobiol.2021.112278.
[10] Bisong E. Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Berkeley, CA: Apress; 2019. https://doi.org/10.1007/978-1-4842-4470-8_7.
[11] Wang J, Li Y, Feng H, Ren L, Du X, Wu J. Common pests image recognition based on deep convolutional neural network. Comput Electron Agric 2020;179: 105834. https://doi.org/10.1016/j.compag.2020.105834.
[12] Li Y, Wang H, Dang LM, Sadeghi-Niaraki A, Moon H. Crop pest recognition in natural scenes using convolutional neural networks. Comput Electron Agric 2020; 169:105174. https://doi.org/10.1016/j.compag.2019.105174.
[13] Chen P, Li WL, Yao SJ, Ma C, Zhang J, Wang B, Zheng CH, Xie CJ, Liang D. Recognition and counting of wheat mites in wheat fields by a three-step deep learning method. Neurocomputing 2021;437:21–30. https://doi.org/10.1016/j.neucom.2020.07.140.
[14] Math RKM, Dharwadkar NV. Early detection and identification of grape diseases using convolutional neural networks. J Plant Dis Prot 2022;129(3):521–32. https://doi.org/10.1007/s41348-022-00589-5.
[15] Grünig M, Razavi E, Calanca P, Mazzi D, Wegner JD, Pellissier L. Applying deep neural networks to predict incidence and phenology of plant pests and diseases. Ecosphere 2021;12(10). https://doi.org/10.1002/ecs2.3791.
[16] Thenmozhi K, Srinivasulu Reddy U. Crop pest classification based on deep convolutional neural network and transfer learning. Comput Electron Agric 2019;164: 104906. https://doi.org/10.1016/j.compag.2019.104906.
[17] Lins EA, Rodriguez JP, Scoloski SI, Pivato J, Lima MB, Fernandes JM, da Silva Pereira PR, Lau D, Rieder R. A method for counting and classifying aphids using computer vision. Comput Electron Agric 2020;169:105200. https://doi.org/10.1016/j.compag.2019.105200.
[18] Nanehkaran YA, Zhang D, Chen J, Tian Y, Al-Nabhan N. Recognition of plant leaf diseases based on computer vision. J Ambient Intell Humaniz Comput 2020. https://doi.org/10.1007/s12652-020-02505-x.
[19] Y. Wu, A. Kirillov, F. Massa, W.Y. Lo, and R. Girshick, "Detectron2" https://github.com/facebookresearch/detectron2 (2019).

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," arXiv.org, 06-Jan- 2016. [Online]. Available: https://arxiv.org/abs/1506.01497. [Accessed: 29-Aug-2021].
[21] T.Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," arXiv.org, 07-Feb- 2018. [Online]. Available: https://arxiv.org/abs/1708.02002. [Accessed: 29-Aug-2021].
[22] Dong S, Du J, Jiao L, Wang F, Liu K, Teng Y, Wang R. Automatic crop pest detection oriented multiscale feature fusion approach. Insects 2022;13(6):554. https://doi.org/10.3390/insects13060554.
[23] Fuentes A, Yoon S, Kim S, Park D. A robust deep-learning-based detector for real-time tomato plant diseases and pests' recognition. Sensors 2017;17(9):2022. https://doi.org/10.3390/s17092022.
[24] Jiao L, Dong S, Zhang S, Xie C, Wang H. Af-RCNN: an anchor-free convolutional neural network for multi-categories agricultural pest detection. Comput Electron Agric 2020;174:105522. https://doi.org/10.1016/j.compag.2020.105522.
[25] Selvaraj MG, Vergara A, Ruiz H, Safari N, Elayabalan S, Ocimati W, Blomme G. AI-powered banana diseases and pest detection. Plant Methods 2019;15(1). https://doi.org/10.1186/s13007-019-0475-z.
[26] Liu T, Chen W, Wu W, Sun C, Guo W, Zhu X. Detection of aphids in wheat fields using a computer vision technique. Biosyst Eng 2016;141:82–93. https://doi.org/10.1016/j.biosystemseng.2015.11.005.
[27] Mkonyi L, Rubanga D, Richard M, Zekeya N, Sawahiko S, Maiseli B, Machuve D. Early identification of Tuta absoluta in tomato plants using deep learning. Sci Afr 2020;10. https://doi.org/10.1016/j.sciaf.2020.e00590.

Rajinder Kumar M. Math received his Ph.D. degree from Visvesvaraya Technological University in 2023. He is currently serving as an Assistant Professor in the Department of Electronics and Communication Engineering of B.L.D.E. A's CET, Vijayapur, Karnataka. His research interests include the design of crop management systems for precision agriculture, emphasising on harnessing the capabilities of machine learning, deep learning, and computer vision in providing agricultural automation.

Nagaraj V. Dharwadkar obtained his Ph. D in Computer Science and Engineering in 2014 from National Institute of Technology, Warangal. He is a Professor and the Head of the Computer Science and Engineering Department at the Rajarambapu Institute of Technology, Islampur. He has 23 years of teaching experience and published 100 papers in various international journals. His area of research interest includes multimedia security, image processing, data mining and machine learning.