# An Improved YOLOv5 for Accurate Detection and Localization of Tomato and Pepper Leaf Diseases

**Balkis Tej**
University of Monastir: Universite de Monastir
**Soulef Bouaafia**
soulef.bouaafia@fsm.rnu.tn
University of Monastir: Universite de Monastir https://orcid.org/0000-0003-0657-6900
**Mohamed Ali Hajjaji**
University of Sousse: Universite de Sousse
**Abdellatif Mtibaa**
University of Sfax: Universite de Sfax

**Research Article**

**Keywords:** Agriculture 4.0, Plant Leaf Disease, Object Detection and Localization, YOLOv5 Network

**Posted Date:** February 26th, 2024

**DOI:** https://doi.org/10.21203/rs.3.rs-3358463/v1

# An Improved YOLOv5 for Accurate Detection and Localization of Tomato and Pepper Leaf Diseases

Balkis Tej[1,2], Soulef Bouaafia[3,4*], Mohamed Ali Hajjaji[3,5] and Abdellatif Mtibaa[1]

[1]Systems Integration & Emerging Energies Laboratory (LR21ES14), National Engineering School of Sfax, University of Sfax, Sfax, Tunisia.
[2] National Engineering School of Monastir, University of Monastir, Monastir, Tunisia.
[3] Electronics and Microelectronics Laboratory, FSM, University of Monastir, Monastir, Tunisia.
[4] Higher Institute of Applied Sciences and Technology of Kairouan, University of Kairouan, Kairouan, Tunisia.
[5] Higher Institute of Applied Sciences and Technology of Sousse, University of Sousse, Sousse, Tunisia.

*Corresponding author(s). E-mail(s): soulef.bouaafia@fsm.rnu.tn;
Contributing authors: balkis.tej@enim.u-monastir.tn; mohamedali.hajjaji@issatso.rnu.tn; abdellatif.mtibaa@enim.rnu.tn;

**Abstract**

Agriculture serves as a vital sector in Tunisia, supporting the nation's economy and ensuring food production. However, the detrimental impact of plant diseases on crop yield and quality presents a significant challenge for farmers. In this context, computer vision techniques have emerged as promising tools for automating disease detection processes. This paper focuses on the application of the YOLOv5 algorithm for the simultaneous detection and localization of multiple plant diseases on leaves. By using a self-generated dataset and employing techniques such as augmentation, anchor clustering, and segmentation, the study aims to enhance detection accuracy. An ablation study comparing YOLOv5s and YOLOv5x models demonstrates the superior performance of YOLOv5x, achieving a mean average precision (mAP) of 96.5%.

**Keywords:** Agriculture 4.0, Plant Leaf Disease, Object Detection and Localization, YOLOv5 Network

## 1 Introduction

With a long-standing tradition of cultivation in Tunisia, tomato (Solanum lycopersicum L.) and pepper (Capsicum annum L.) have emerged as two of the primary vegetable crops, playing a vital role in the country's agricultural landscape and economy. These crops hold significant economic importance due to their widespread cultivation and consumption. In fact, tomato cultivation covers an extensive area of approximately 24,500 hectares, yielding an annual production of 1,416,000 tons [29]. Similarly, pepper crops encompass an area of around 19,000 hectares, yielding an annual

production of 304,000 tons [9]. However, the sustainable production of tomatoes and peppers faces a significant challenge in the presence of various leaf diseases. These diseases, caused by a diverse range of fungal, bacterial, and viral pathogens, pose significant obstacles to crop yield and quality. They result in substantial economic losses for farmers and pose a considerable threat to both food security and livelihoods.

Traditional techniques for plant disease detection have relied heavily on manual visual inspection methods. Trained experts visually examine the plants for symptoms such as discoloration, lesions, or deformities, which can indicate the presence of diseases. However, this technique has many limitations. Firstly, visual inspection is subjective [4], as different experts may interpret symptoms differently, leading to inconsistencies in disease diagnosis. Secondly, it is a time-consuming process [8], particularly in large agricultural fields, and may not be feasible for timely disease management. Additionally, visual symptoms may not always be apparent, especially during the early stages of disease development, resulting in delayed detection and intervention. To address these limitations, improved strategies that can automate and enhance the accuracy of disease detection and localization in tomato and pepper plants are required. Computer vision and learning algorithms offer promising solutions in this regard. With the advancement of machine learning (ML), researchers have deployed various techniques in the field of agricultural science for plant disease detection. Notably, approaches such as K-nearest neighbors (KNN) [35], support vector machines(SVM) [39], Random forest [23], and decision tree (DT) [26] have been investigated to inspect and diagnose plant diseases. Despite their simplicity and minimal data requirements for training, ML-based approaches are time-consuming and heavily rely on trained human resources. In recent years, Deep Learning (DL) has emerged as a key technology in the field of agriculture to overcome ML-based approaches limitations. DL techniques such as convolutional neural networks (CNNs) [30], long-short-term memory (LSTM) [7], and recurrent neural networks (RNNs) [40] have been successfully applied to address complex challenges in agriculture, including plant disease detection. Among the various deep learning models, convolutional neural networks (CNNs) have emerged as the most widely used and effective models for plant disease detection. CNN models, such as Resnet [1], VGG [22], and AlexNet [34], have demonstrated superior performance in accurately classifying plant diseases. These models have been studied and fine-tuned to achieve high accuracy in disease classification tasks. These traditional classification approaches are effective in determining the category or type of objects present in an image. However, they do not provide information about where exactly these objects are located within the image. In other words, while they can tell us what objects are in the picture, they cannot tell us their specific positions. Object detection gets more advanced and challenging. Existing methods for detecting objects can be classified into two main categories: two-stage detection algorithms and one-stage detection algorithms.

The traditional object detection method initially introduced a two-stage detection algorithm. The R-CNN model, proposed by Girshick et al. [6]in 2014, marked the inception of this approach. It typically involves a two-step process. In the first stage, a region proposal network (RPN) generates potential regions of interest (RoIs) in the image that may contain objects. These RoIs are then further refined and classified in the second stage using techniques like Fast R-CNN or Faster R-CNN. Two-stage detectors are known for their superior accuracy, but may have slower inference speeds due to the additional processing steps involved. On the other hand, the one-stage detection algorithm eliminates the need for a Region Proposal Network (RPN) used in the two-stage approach. This allows for direct detection of the object category and regression of the bounding box, resulting in reduced detection time. One of the most notable algorithms representing this approach is You Only Look Once (YOLO). In this context, this paper introduces an improved YOLOv5 approach for the detection and localization of tomato and pepper leaf diseases using a self-generated dataset collected in the region of Monastir, Tunisia. The primary paper contributions are as follows:

- A YOLOv5-based regional detection model is developed to detect and localize multiple tomato and pepper leaf diseases from a single leaf.

- A dataset of tomato and pepper leaf diseases is constructed, containing images of the most relevant tomato and pepper leaf diseases in Tunisia.
- Data augmentation techniques are employed to extend training data in order to overcome the issue of insufficient data and improve the detection and identification performance of the network.
- Automatic segmentation of leaf area from leaf images and anchor clustering algorithms are used to improve the model's robustness.

The rest of the paper is structured as follows: Section 2 provides a review of related works in the field. Section 3 presents the proposed methodology. The dataset description is provided in Section 4. Experimental Results and Discussion are presented in Section 5. Finally, Section 6 concludes the paper and outlines future directions for research.

## 2 Related Works

Several studies have explored the application of object detection algorithms for the accurate and efficient detection of plant leaf diseases.
Nawaz et al. [20] proposed an approach named Faster RCNN based on ResNet-34 and Convolutional Block Attention to localize and classify tomato plant leaf disease. The proposed method achieved an accuracy of 99.97% and a mAP of 0.981 using the public dataset PlantVillage. Patil et al. [24] introduced a novel method to estimate the severity of rice diseases using a private dataset of 1200 images. The proposed model is based on an optimized faster RCNN model using the EfficientNet-B0 architecture as the backbone, for the calculation of leaf area and infected region. It attained an accuracy of 96.43%. Saleem et al. [31] used multiple deep learning algorithms, including Faster-RCNN, SSD, and Region-based Fully Convolutional Networks (RFCN), to detect and classify diseased regions in plants. The highest mAP achieved is 73.07% using PlantVillage dataset. Peng et al. [25] proposed a detection algorithm based on an optimized YOLOv5s model and deep metric learning to identify and localize diseases affecting plant leaves using different public datasets. The best results are achieved when using Resnet50 as the backbone network and the Plantvillage dataset, with an accuracy of 97.88%. Jain et al. [10] developed a smartphone application for automated detection of rice diseases using Yolov3 tiny and yolov4 tiny. Among the two, YOLOv4 tiny yielded the highest performance, achieving an mAP of 97.36% on a dataset of 762 web-collected images. Dai et al. [3] introduced an improved version of YOLOv5 called DA-ActNN-YOLOv5 for regional potato disease detection across multiple cycles. The YOLOv5 network's component modules were replaced using model compression technology (ActNN). The proposed approach reached an mAP of 99.81% when evaluated on a self-generated dataset, leveraging different data augmentation techniques. Li et al. [15] present an improved YOLOv5s model for apple leaf disease detection called BTC-YOLOv5s model. It incorporates the bidirectional feature pyramid network (BiFPN) to effectively fuse features at different scales. Additionally, attention mechanisms such as the transformer and convolutional block attention module (CBAM) are integrated to minimize interference from irrelevant background information and enhance the expression of disease characteristics. The experimental results demonstrated that the proposed approach achieved an mAP of 84.3% when evaluated on a dataset consisting of 2099 images collected from three publicly available datasets. Soeb et al.[36] present an AI-based solution for detecting tea-leaf diseases by training the YOLOv7 model on a dataset collected from tea gardens in Bangladesh. The obtained results showed that the proposed method achieved an mAP of 98.2%. Lin et al. [17] proposed a real-time detection model called TSBA-YOLO for tea disease detection using a self-generated dataset. The model incorporates the self-attention mechanisms of convolutional and transformer layers to enhance global perception and obtain more contextual information. The YOLOv5's PAFPN is replaced with a BiFPN structure for effective multi-scale target fusion. Additionally, the integration of the Shuffle Attention (SA) mechanism enhances the semantic expression of characteristics related to tea diseases. The TSBA-YOLO model achieves an average accuracy mAP of 85.35%. Table 1 presents a comprehensive overview of the existing studies found in the literature.

**Table 1**: Studies summary of plant leaf disease detection

| Ref | Model | Dataset | Number of images | mAP (%) |
|---|---|---|---|---|
| [20] | Faster-RCNN (Resnet34) | PlantVillage | 18 160 | 98.1% |
| [24] | Faster-RCNN (EfficientNet-B0) | Self-generated | 1 200 | 96.43% |
| [31] | Faster-RCNN, SSD and RFCN | PlantVillage | 54 306 | 73.07% |
| [25] | Improved YOLOv5s (Resnet50) | Plantvillage | 16 270 | 97.88% |
| [10] | Yolov4 tiny | Web images | 762 | 97.63% |
| [3] | DA-ActNN-YOLOv5 | Self-generated | 4 546 | 99.81% |
| [15] | BTC-YOLOv5s | Public dataset | 2 099 | 84.3% |
| [36] | YOLOv7 | Self-generated | 4 000 | 98.2% |
| [17] | TSBA-YOLO (YOLOv5) | Self-generated | 1 000 | 85.35% |

# 3 Proposed Methodology

## 3.1 Proposed Approach

An overview of the proposed workflow for plant leaf disease detection is shown in Fig. 1. Firstly, a collection of images is captured using a phone camera. The collected images undergo a data cleaning process to remove any irrelevant or noisy data points.

Next, leaf segmentation from collected images is performed to isolate the plant leaves and eliminate the background. This stage guarantees that the next analysis and detection algorithms are entirely focused on the leaves. Following segmentation, the data is annotated with LabelImg, where bounding boxes are manually created around the plant leaf areas of interest and labeled with class information. This phase is critical for training the detection model. An anchor algorithm is applied to the annotated data to increase the detection model's accuracy. This method assists in the identification and assignment of appropriate anchor boxes that determine the spatial attributes of objects inside pictures. After that, the dataset is divided into training and validation sets, with the training set being used to train the YOLOv5 model. To improve the model's capacity to generalize and handle different circumstances, data augmentation techniques are applied to the training data. The trained model's performance and detection accuracy are assessed using the validation set. To measure the model's performance in detecting plant leaf diseases, metrics such as mAP, accuracy, recall, and F1-score are calculated. Finally, the model's output findings are examined and contrasted. The approach's strengths and weaknesses are examined, and prospective enhancements or changes to the process are identified.

## 3.2 YOLOv5 Network

YOLOv5 [12], one of the most advanced techniques for object detection, is used for detecting plant leaf disease. The YOLOv5 employs a single neural network to analyze the complete image in its entirety. It subsequently partitions the image into regions and assigns probabilities to the bounding boxes of each individual component. It takes an image as input and processes it through a neural network. The output consists of bounding boxes, which indicate where the objects are

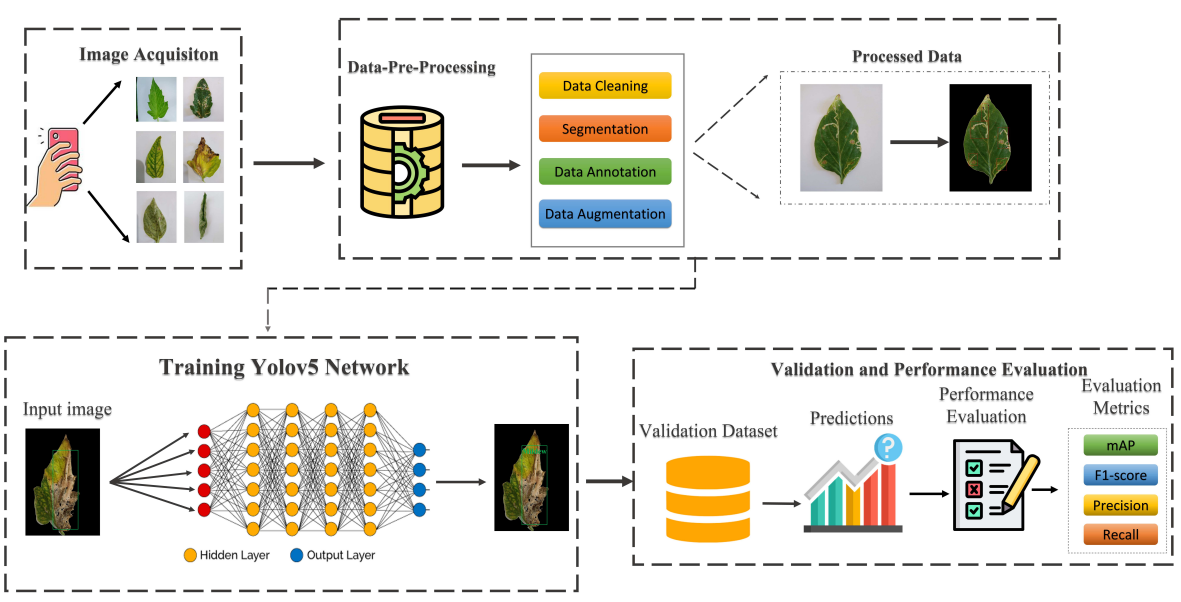



**Fig. 1**: Overview of the proposed approach

located in the image, and class probabilities, which represent the likelihood of each object belonging to a specific category. The YOLOv5 network has evolved through different versions, including YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x [38]. These versions maintain a similar network structure while adjusting the depth and width of feature maps.

The YOLOv5 object detection algorithm is a derivative of YOLOv4, incorporating several improvements and optimizations. While sharing similarities with YOLOv4, YOLOv5 offers distinct advantages that make it more suitable for deployment on embedded devices and enable faster object detection. Compared to YOLOv4, YOLOv5 has a significantly reduced model size, with 90% fewer weight files [11]. This reduction in complexity allows for easier deployment on embedded devices, making real-time object detection more achievable. The YOLOv5 architecture, as shown in Fig. 3, consists of a backbone network, neck layers, and a detection head [13]. The backbone network, based on CSPDarknet53, serves as the feature extractor and captures rich contextual information from the input image. It consists of convolutional layers with different receptive fields, enabling the detection of objects at various scales.

The neck component encompasses a series of layers designed to mix and combine image features before feeding them into the prediction stage. The primary purpose of the neck is to enhance the representation of features extracted from the input image. Specifically, YOLOv5 incorporates PAnet (Path Aggregation Network), a modified version of the Feature Pyramid Network (FPN), serving as the neck module. The detection head operates on the features provided by the Neck (PAnet) module. It is responsible for generating the final predictions. It consists of convolutional layers that process the fused features and predict the coordinates of bounding boxes and the probabilities associated with class labels. YOLOv5s employs anchor boxes, predefined bounding box priors, to handle objects of different sizes and aspect ratios. The standout feature of YOLOv5 lies in its utilization of the Focus and CSP (cross-stage partial connections) layers. The introduction of the Focus layer addresses the objective of reducing the number of layers, parameters, FLOPS (floating-point operations per second), and CUDA memory usage. Additionally, it aims to enhance the speed of both forward and backward and minimize the impact on the mAP metric. In the previous YOLOv5 version, the Focus layer was optimized to operate with just one layer, as opposed to the three layers employed in YOLOv3 [28]. Furthermore, the CSP layer Cross Stage Partial (CSP) expands the incorporation of basic information from the focus layer

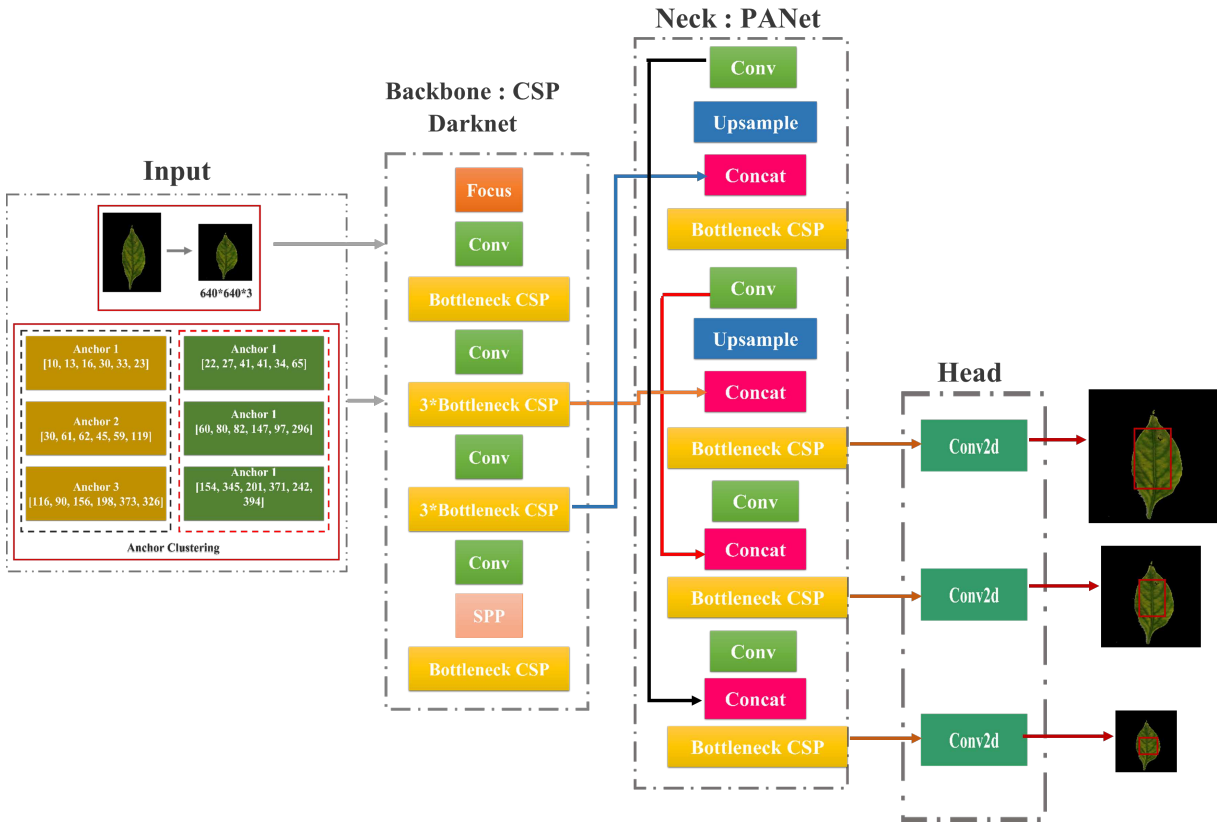



**Fig. 2**: Improved YOLOv5 Architecture

to enhance overall functionality and allow efficient information flow between different network layers, aiding in better feature representation and object localization [16].

In the context of object detection using the YOLOv5 network, there are two important choices to consider during the training process. Firstly, the selection of activation and optimization functions plays a crucial role. Commonly, the Leaky rectified linear unit (ReLU) and Sigmoid functions are used as activation functions. The Leaky ReLU function is often applied in the middle or hidden layers, while the Sigmoid function is typically employed in the final detection layer. Regarding optimization, various techniques can be utilized, such as Stochastic gradient descent (SGD) and adaptive moment estimation (ADAM), to optimize the network parameters and improve its performance. Second, selecting the loss function is critical for accurate YOLOv5 network training. The loss function in YOLOv5 is often a compound loss that considers various components such as the objectness score, class probability score, and bounding box regression score. Depending on the issue, the loss function can take several forms, such as binary cross-entropy loss for binary classification tasks or categorical cross-entropy loss for multi-class classification tasks. Equation (1) is the overall loss function of YOLOv5, where the loss is the sum of three components: $l_{\text{box}}$, $l_{\text{cls}}$, and $l_{\text{obj}}$.

$$Loss = l_{obj} + l_{box} + l_{cls} \tag{1}$$

where $l_{\text{box}}$ is the bounding box regression function. This function is responsible for predicting accurate bounding box coordinates for the detected objects. The $l_{\text{cls}}$ component is the loss function for classification. It focuses on predicting the correct class labels for the detected objects. The $l_{\text{obj}}$ component is the loss function for confidence. It assesses the confidence or certainty of the model's predictions.

### 3.3 Anchor Clustering Algorithm

Due to the specific dataset used in this study, the initial values of Anchors had to be adjusted in advance to fit the dataset and the desired Anchors size. This adjustment was achieved by applying a clustering algorithm. The chosen approach was the K-means++ algorithm, which addresses the issue of initializing cluster centers in the standard K-means algorithm.

The K-means algorithm is a distance-based clustering approach that evaluates similarity based on the distance between data points. It assigns objects that are closer in space a higher degree of similarity, leading to the formation of clusters with high internal similarity and low similarity between different clusters. However, determining suitable initial values for the K-means algorithm can be difficult since they need to be artificially assigned without a well-defined criterion, potentially impacting the final clustering results. To overcome these limitations, the K-means++ algorithm [21] was employed in this study. This algorithm follows a similar methodology to the K-means algorithm but improves the initialization process. It selects cluster centers iteratively, giving preference to sample points that are farther away from existing cluster centers. This approach increases the likelihood of selecting diverse and representative cluster centers. The following algorithm 1 provides an explanation of the anchor clustering algorithm.

**Algorithm 1** K-means++ Initialization

**Require:** Dataset $X$ containing $n$ samples, Number of cluster centers $K$

**Ensure:** Selected cluster centers representing the initial centroids

1: Randomly select a sample point $x_1$ from the dataset $X$ as the first initial cluster center.
2: Initialize an empty list to store the selected cluster centers: $C \leftarrow [x_1]$.
3: **for** $i \leftarrow 2$ to $K$ **do**
4: Calculate the shortest distance $D(x)$ between each sample $x$ and the currently selected cluster centers:
5: **for** each sample $x$ in the dataset $X$ **do**
6: Calculate the distance $d(x, C)$ to the currently selected cluster centers $C$.
7: **end for**
8: Calculate the probability $P(x)$ for each sample $x$ to be selected as the next cluster center:
9: $P(x) \leftarrow \frac{D(x)^2}{\sum_{x'} D(x')^2}$, where $x'$ iterates over all samples in the dataset $X$.
10: Select the sample $x$ with the maximum probability $P(x)$ as the next cluster center:
11: $x_i \leftarrow \arg\max_x P(x)$
12: Add the selected cluster center $x_i$ to the list of cluster centers: $C \leftarrow C \cup \{x_i\}$.
13: **end for**
14: Return the selected cluster centers $C$ as the initial centroids.

### 3.4 Segmentation Algorithm

Upon examining the images in our dataset, we observed the presence of phone and leaf shadows, which can significantly affect the accuracy of disease detection algorithms. The presence of shadows and backgrounds introduces noise and distractions, making it challenging to isolate and analyze the leaf regions accurately. Additionally, it can obscure important features, further complicating the detection process. To address these challenges, we propose a segmentation algorithm 2 to remove background, which effectively eliminates the unwanted elements, allowing for focused analysis of the leaf regions. Our algorithm starts by converting the images to the LAB color space, which provides better separation between the leaf and background. We then apply adaptive thresholding techniques using Otsu methods [33] to specific color channels to create binary masks, highlighting the leaf regions while suppressing the background. Morphological operations are applied to further refine the masks and remove noise. Finally, the inverted masks are employed for leaf region extraction, resulting in clean and isolated representations of the leaves for disease detection. By employing this segmentation algorithm, we can overcome the limitations posed by backgrounds and leaf shadows, enabling more accurate and reliable disease detection on plant leaves. The removal of backgrounds enhances the visibility of leaf features, allowing for better analysis and monitoring of plant health. Fig. 3 provides a visual representation of the output obtained at various steps of the segmentation algorithm.

**Algorithm 2** Leaf Segmentation algorithm with Background Removal

**Require:** RGB image of a plant leaf
**Ensure:** Region of interest (leaf area) from the input leaf image

1: Convert the RGB image to LAB color space.
2: Split the LAB channels into L, A, and B channels.
3: Apply adaptive thresholding on the L channel using Otsu's thresholding method to obtain a binary image (*l_binary*).
4: Apply adaptive thresholding on the B channel (blue-yellow channel) using the inverse Otsu's thresholding method to obtain a binary image (*b_binary*).
5: Combine the binary images (*l_binary* and *b_binary*) using the bitwise AND operation to create a mask representing the leaf area.
6: Perform morphological operations on the mask to remove noise and refine the segmented area. This involves applying the opening operation followed by the closing operation with a predefined kernel.
7: Invert the refined mask to make the background black and the leaf area white.
8: Apply the inverted mask to the original image using the bitwise AND operation to extract the segmented leaf area.
9: Display the segmented image showing the leaf area.

# 4 Dataset Collection and Preparation

Deep learning techniques heavily rely on the availability of suitable and effective datasets to achieve optimal performance. When it comes to studying tomato and pepper plants in Tunisia, the choice of dataset becomes crucial. Existing studies on tomato and pepper leaf diseases commonly use an open dataset sourced from PlantVillage. However, this dataset is tailored to a specific region, considering distinct geographical and environmental factors. It's important to note that tomato and pepper diseases can differ across various regions worldwide due to factors such as shape, variety, and environmental conditions. Also, this open dataset doesn't encompass the common diseases that affect tomato and pepper plants in Tunisia.

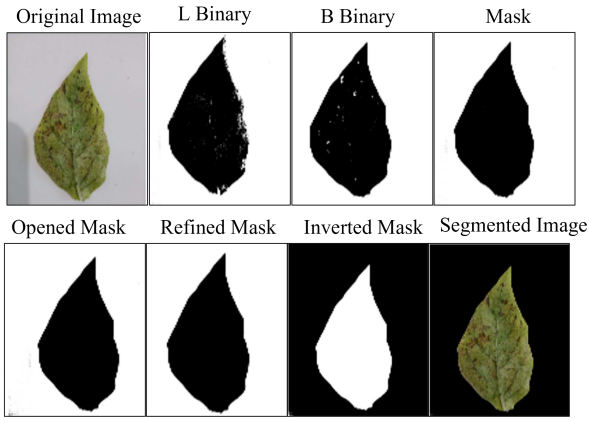



**Fig. 3**: Progression of Segmentation Algorithm: Output at Different Steps

Therefor, the development of a dedicated dataset specifically tailored to the local context of Tunisia is essential, as it will greatly benefit Tunisian farmers. This study specifically aims to create such a dataset, designed to capture disease profiles that are relevant to tomato and pepper crops in Tunisia.

## 4.1 Image Acquisition

The images for this dataset were captured from different tomato and pepper greenhouses located in the Bkalta region of Tunisia. Bkalta is well-known for its favorable weather, featuring ample sunshine and moderate temperatures, making it an ideal location for tomato and pepper cultivation. As a major producer of tomatoes and peppers in Tunisia, the region boasts thriving agricultural activities and a strong farming community. The greenhouses, generously provided by local farmers, had an area of 500 $m^2$ and consisted of four rows, with each row containing two lines of plants. The dataset images were taken in a controlled environment using an OPPO A30 phone camera with a resolution of $4096 \times 3072$ pixels and a capture distance of the cell phone camera of 30 cm. The data collection process was conducted on two specific dates, July 7, 2021, and January 2, 2022, to capture the seasonal variations in tomato and pepper crops and their associated diseases.

The acquisition process was conducted under the supervision of an agricultural engineer from the Ministry of Agriculture in Tunisia, who provided guidance and expertise specifically related

**Table 2**: Dataset Composition

| Class | NI | Symptoms | Image Example |
|---|---|---|---|
| Healthy | 90 | No visible signs of disease or abnormalities |  |
| Leaf miner | 95 | Yellow, squiggly lines on the leaves |  |
| Oidium | 121 | Blackening, chlorosis, and white powdery lesions on the lower surface of the leaf |  |
| Nutrient Deficiency | 67 | Chlorosis in the interveinal areas of the leaf, while main veins retain green coloration |  |
| Mildew | 57 | Small brown spots with a yellow halo |  |
| Tomato yellow curve virus | 70 | Yellowing of leaves, reduction of leaf size, curling of leaf margins |  |

to the description and identification of diseases. Their valuable supervision ensured the accuracy and consistency in capturing and documenting disease-related characteristics in the dataset. Table 2 provides an illustration of the tomato and pepper leaf diseases included in the dataset. It showcases the dataset composition, consisting of a total of 500 leaf images, classified into six distinct disease classes: healthy, leaf miner, oidium, nutrient deficiency, mildew, and tomato yellow curve virus. The collection of the dataset presented a major hurdle during our research. Capturing images for our exclusive dataset using a mobile camera proved to be a time-intensive process, requiring substantial effort and dedication.

## 4.2 Data Preprocessing

In the data preprocessing stage, two crucial steps were performed: data annotation and data augmentation.These steps played a crucial role in preparing the data for subsequent analysis and model training.

### 4.2.1 Data Annotation

Data annotation plays a crucial role in training deep learning models for object detection tasks. In our research, we employed the 'labelimg' tool, a popular annotation tool that offers a user-friendly interface for drawing bounding boxes around objects and assigning corresponding class labels. By executing the command 'pip install

labelImg' in our Python environment, we seamlessly installed the labelimg tool. With 'labelimg', we were able to accurately annotate our dataset by marking the affected regions of interest and assigning the appropriate disease class labels, as shown in Fig. 4. The tool generated output text files containing the annotated data, including the provision of bounding box coordinates along with their corresponding class labels. The text annotation file includes YOLO annotations that indicate the bounding boxes and class labels of objects in the image. The file's lines match annotated objects. The text file includes five decimal numbers, each of which serves a distinct function. The first value shows the class or category of the item within the bounding box that is being annotated. It aids in determining the sort of object in the image. The second value represents the center x-coordinate of the bounding box. It denotes the horizontal position of the box's center relative to the width of the image. The third value denotes the center y-coordinate of the bounding box, representing the vertical position of the box's center relative to the height of the image.The fourth value signifies the width of the bounding box, indicating the extent of the object in the horizontal direction, again relative to the width of the image. The fifth value represents the height of the bounding box, indicating the extent of the object in the vertical direction, relative to the height of the image. To ensure consistency and facilitate model training, these values are normalized within the range of 0 and 1. This normalization is achieved by dividing the values by the width and height of the image. Normalizing the values to this range helps the neural network predict and learn from them more effectively during the training process. The number of lines in the text file corresponds to the number of bounding boxes drawn and annotated in the image. Each line in the file represents a separate bounding box annotation, allowing for multiple annotations within a single image.

### 4.2.2 Data Augmentation

Due to the limited size of our dataset and the imbalance in class distribution, data augmentation techniques were employed.Data augmentation refers to the process of artificially expanding a dataset by applying various transformations or modifications to the existing data samples. It is a common technique used in machine learning and computer vision tasks to increase the diversity and quantity of training data without adding new data[37]. By applying transformations, data augmentation aims to simulate variations that may occur in real-world scenarios. This approach helps improve the generalization and robustness of deep learning models by exposing them to a broader range of data instances. The data augmentation and pre-processing techniques used in this study were performed using a set of specific parameters, which are summarized in Table 3. To provide a visual representation of the effects of these parameters, Fig. 5 showcases sample images after undergoing data augmentation and pre-processing using the aforementioned parameter settings from Table 3. The initial set of 373 training images was augmented, resulting in a total of 3640 images.

- Auto Orientation: The auto-orientation pre-processing phase was implemented to ensure that each image in the dataset was aligned in a consistent vertical orientation.
- Resizing: Resizing was performed to standardize the dimensions of all images for consistent YOLO model training.
- Rotation: rotation was applied to help the model be insensitive to camera orientation and improve its robustness and ability to handle images captured from different angles.
- Flip: By flipping the images horizontally and vertically, the model will be insensitive to subject orientation.
- Blur: Random Gaussian blur was employed to replicate the effect of lower-quality or pixelated images, emulating real-life situations where farmers may use cameras with limited capabilities for capturing images.
- Noise: Noise transformation was used to make the model more resilient to various camera artifacts and disturbances that can occur in real-world scenarios
- Saturation, Exposure, and Brightness: Are applied to help the model to be more resilient to lighting and camera setting changes and to account for the diverse lighting conditions that can occur in real-life situations (sunny or cloudy weather...)

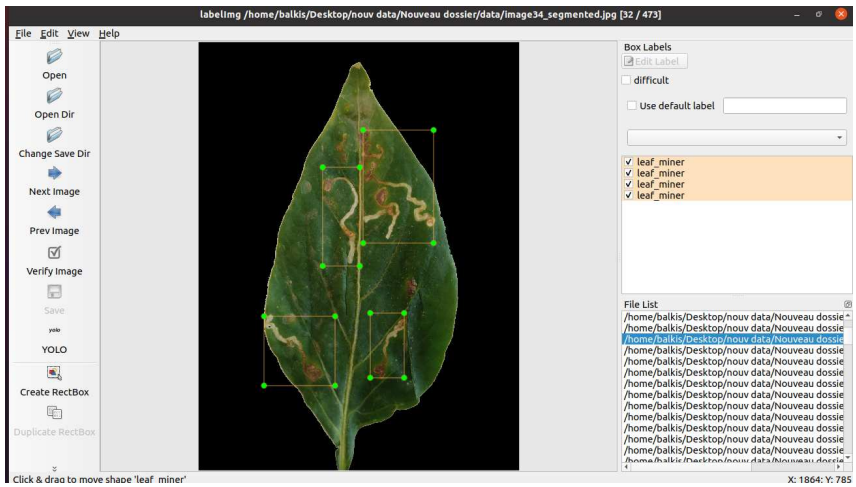



**Fig. 4**: Annotation image example

**Table 3**: Data Augmentation Techniques and Parameters

| Data Augmentation Techniques | Parameters |
|---|---|
| Resizing | 416 X 416 |
| Auto-orientation | Applied |
| Rotation | 90° |
| Flip | Horizontal and vertical flip |
| Random Gaussian Blur | Up to 2.5px |
| Noise | Applied to 5% of pixels |
| Saturation | Between -50% and 50% |
| Exposure | Between -25% and 25% |
| Brightness | Between -30% and 30% |

# 5 Experimental Results and Discussion

## 5.1 Experimental Setup and Training Parameters

In our experimental setup, we conducted the training phase using the Google Colab platform. Google Colab provided us with a convenient and powerful environment for training our deep learning models. It allowed us to leverage the capabilities of high-performance GPUs without the need for expensive hardware. The cloud-based nature of Google Colab also ensured that we had access to ample computational resources and a stable runtime environment throughout our training experiments. This allowed us to efficiently iterate and experiment with different model configurations, hyperparameters, and training strategies.

To perform the experiment, we divided the data into two portions:, allocating 80% for training purposes and reserving 20% for validation. The training phase involved feeding the model with the training images and optimizing its parameters through the process of backpropagation. Once the training was completed, we evaluated the performance of the model using the validation images. This evaluation allowed us to assess how well the trained model generalized to unseen data and provided insights into its overall accuracy and effectiveness. The YOLO object detection models, including YOLOv5, undergo a specific configuration and training process. After labeling the dataset, the model is configured to accommodate the dataset and hardware environment, such as GPU, CUDNN, and OPENCV. To initiate the training of a YOLOv5 model, two YAML files are required. The first YAML file serves to

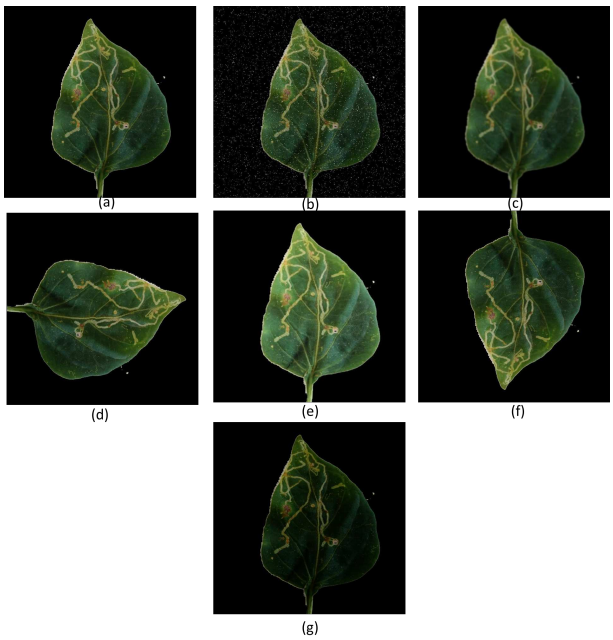



**Fig. 5**: Image examples after applying data augmentation transformations: (a): original image, (b): Noise, (c):Blur, (d):Rotation, (e):Brightness, (f):Flip, (g): exposure (negative)

define essential information such as the training and testing data file paths, the class number to be detected, and the corresponding names of the objects belonging to those classes. The second YAML file encompasses various parameters and configurations, including anchor boxes, the YOLOv5 backbone architecture, and the YOLOv5 head architecture. These YAML files provide the necessary specifications and settings for training the YOLOv5 model, enabling it to learn and detect objects accurately. Hyperparameters and training parameters employed for training YOLOv5 are presented in Table 4.

**Table 4**: Training Hyperparameters

| Parameter | Value |
|---|---|
| Input Image Size | 416x416 |
| Learning Rate | 0.01 |
| Batch Size | 16 |
| Epochs | 50 |
| Optimizer | SGD |
| Weight Decay | 0.0005 |
| momentum | 0.937 |

## 5.2 Evaluation Metrics

Evaluation metrics are crucial for assessing the performance and effectiveness of object detection models. Several assessment criteria were used in the context of YOLOv5 to test the accuracy and robustness of the trained model. Precision, recall, F1-score, average precision (AP), and mAP are the most often used assessment measures. Precision measures the proportion of correctly detected objects out of all the predicted objects. Recall, on the other hand, measures the proportion of correctly detected objects out of all the ground-truth objects. Average precision calculates the average precision values across different levels of confidence thresholds, while mean average precision computes the average precision across all object classes. Among the various evaluation metrics used in target detection algorithms, the mAP holds significant importance in assessing the accuracy of the detection algorithm. Specifically, mAP0.5 refers to the AP obtained by evaluating the object detection model with a threshold of 0.5 for Intersection over Union (IoU). This metric calculates the average value of AP across all categories. Additionally, AP0.5–0.95 measures the mean AP of the model by considering different IoU thresholds from 0.5 to 0.95, with increments of 0.05. This metric provides a more stringent evaluation of model accuracy. Finally, mAP0.5–0.95 represents the average value of AP across all categories when evaluated under different IoU thresholds from 0.5 to 0.95, with a step size of 0.05. In addition to the previously mentioned evaluation metrics, another important factor examined in this study is the number of parameters, which corresponds to the amount of storage space required by the model, typically measured in megabytes (MB). The equations for calculating Recall (R), Precision (P), F1-score [14], AP, and mAP [32] are presented in Equations 2- 6:

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

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

$$AP = \int_0^1 P \cdot R \, dr \tag{5}$$

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

## 5.3 Ablation Study

While YOLOv5 has different versions, such as YOLOv5s and YOLOv5x, it is important to understand how these variations in architecture affect the performance of the models. Conducting an ablation study allows us to systematically analyze and compare the different versions, considering factors such as detection accuracy and computational efficiency. This study helps us gain valuable insights into the specific advantages and trade-offs associated with each YOLOv5 configuration. The presented Table 5 provides a comparison of performance metrics between the YOLOv5s and YOLOv5x models. These metrics include Recall, Precision, F1-Score, mAP (Mean Average Precision), Size (in MB), GFLOPS (Giga Floating-Point Operations Per Second), and FPS (Frames Per Second).

The YOLOv5s model demonstrates a Recall of 91.2%, Precision of 76.4%, F1-Score of 83.14%, and an mAP of 94.8%. It has a smaller model size of 15 MB, requiring less storage space. The GFLOPS for YOLOv5s is 15.8, indicating the computational performance of the model. It achieves an FPS of 137, meaning it can process approximately 137 frames per second.

The YOLOv5x model exhibits improved performance metrics, as well as higher Recall of 95.9%, Precision of 90.6%, F1-Score of 93.1%, and an mAP of 96.5%. However, it has a larger model size of 170 MB, occupying more storage space. The GFLOPS for YOLOv5x is significantly higher at 203.9, indicating its increased computational requirements. The FPS for YOLOv5x is 42, indicating that it can process approximately 42 frames per second. This higher processing speed is beneficial in scenarios that require real-time or near real-time object detection. This comparison highlights the trade-offs between accuracy, model size, computational power, and processing speed. The YOLOv5x model exhibits superior performance in terms of detection accuracy, while the YOLOv5s network offers a more compact size, lower computational requirements, and higher FPS.

### 5.3.1 Accuracy and Loss Analysis

In order to compare the performance of YOLOv5x and YOLOv5s in detecting leaf diseases, we analyzed the loss and accuracy metrics for both models, as shown in Figs. 6 and 7. For YOLOv5s, the classification loss was determined to be 0.00713, indicating the error rate in classifying objects. The objectness loss measured 0.0123, reflecting the confidence of object presence. The bounding box regression loss was recorded as 0.03010, representing the accuracy of predicting object bounding boxes. When considering accuracy, YOLOv5s achieved a precision of 0.74, indicating that 76.4% of the predicted bounding boxes were accurate. The recall value, quantifying the proportion of true bounding boxes correctly predicted, was 0.912, indicating a high level of detection capability. The mAP at an IoU threshold of 0.5 was determined to be 0.948, which represents the overall performance in object detection. The average mAP across varying IoU thresholds from 0.5 to 0.948 (mAP 0.5:0.95) was computed as 0.648, indicating the model's ability to maintain consistent performance across different IoU thresholds.

During the validation or testing phase, we recorded the specific errors associated with the components: box-loss of 0.0233, obj-loss of 0.00567, and cls-loss of 0.0113. These values provide insights into the performance of the model and areas for potential improvement.

In contrast, YOLOv5x demonstrated superior performance compared to YOLOv5s. The classification loss for YOLOv5x was 0.00362, indicating a lower error rate in classifying objects. The objectness loss measured 0.0104, reflecting higher confidence in detecting objects. The bounding box regression loss achieved a value of 0.0247, indicating improved accuracy in predicting object bounding boxes.

Regarding accuracy measures, YOLOv5x achieved a precision of 0.906, representing an improvement over YOLOv5s. The higher precision indicates a higher percentage of accurate bounding box forecasts. The recall value for YOLOv5x was 0.959, indicating a higher proportion of true bounding boxes correctly predicted. The mAP at an IoU threshold of 0.5 was determined to be

**Table 5**: Comparison of Performance Metrics: YOLOv5s vs. YOLOv5x

| Model | Recall (%) | Precision (%) | F1-Score (%) | mAP (%) | Size (MB) | GFLOPs | FPS |
|---|---|---|---|---|---|---|---|
| YOLOv5s | 91.2 | 76.4 | 83.14 | 94.8 | 15 | 15.8 | 137 |
| YOLOv5x | 95.9 | 90.6 | 93.17 | 96.5 | 170 | 203.9 | 42 |

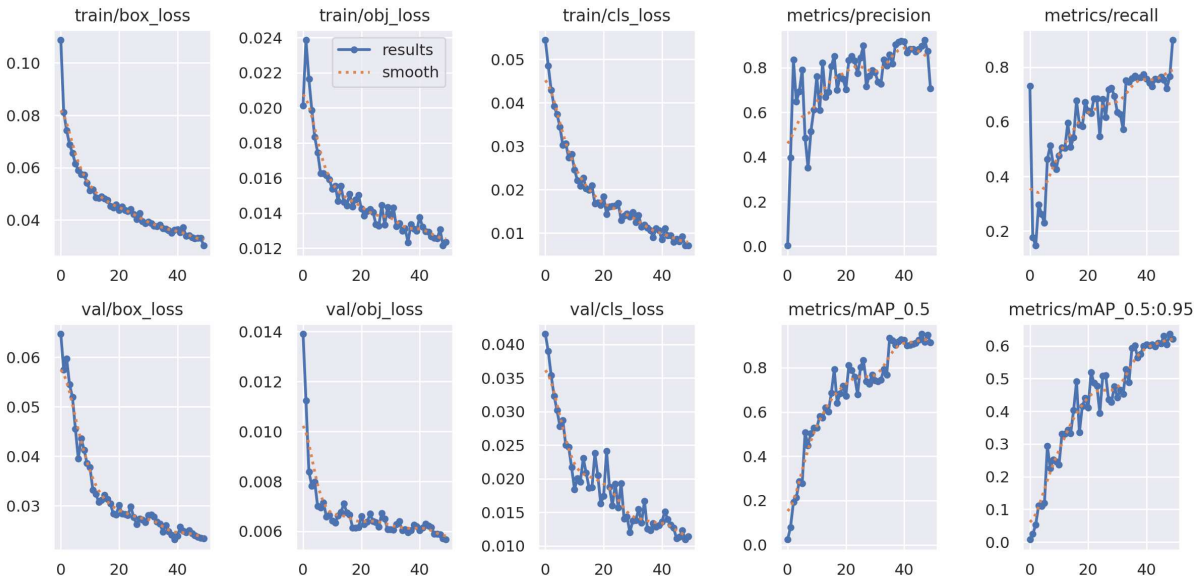



**Fig. 6**: Accuracy and loss graph of leaf disease detection training using YOLOv5s model

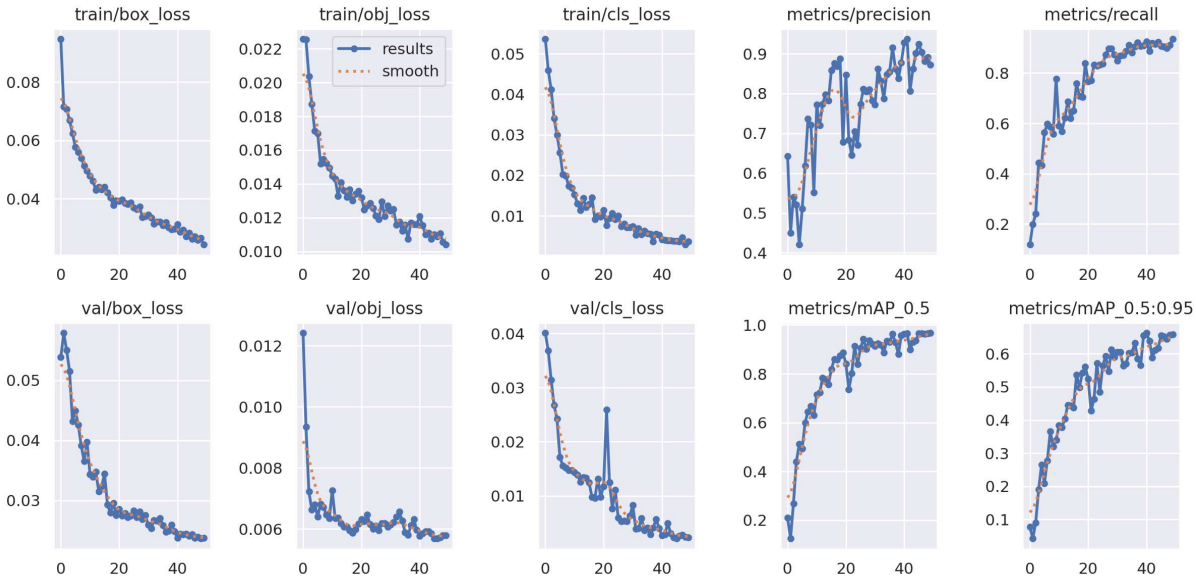



**Fig. 7**: Accuracy and loss graph of leaf disease detection training using YOLOv5x model

0.965, indicating improved overall performance in object detection. The average mAP across varying IoU thresholds from 0.5 to 0.95 (mAP 0.5:0.95) was computed as 0.69, demonstrating the model's ability to maintain high performance across different IoU thresholds.

During the validation or testing phase for YOLOv5x, the recorded errors for box-loss, obj-loss, and cls-loss were 0.0237, 0.0057, and 0.0023, respectively. These values suggest an improvement in the accuracy of these specific components compared to YOLOv5s.

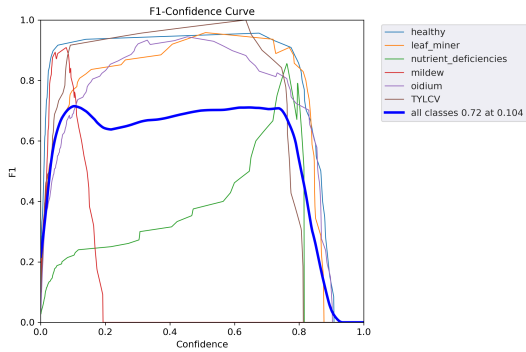
Based on these results, it can be inferred that YOLOv5x outperforms YOLOv5s in terms of both loss metrics and accuracy measures. The lower loss values and higher accuracy metrics for YOLOv5x indicate its enhanced capability in detecting leaf diseases with improved precision and recall. The higher mAP values for YOLOv5x also indicate its better performance in object detection across different IoU thresholds.

These findings have practical implications for the application of YOLOv5 models in detecting leaf diseases. The superior performance of YOLOv5x suggests that it can provide more accurate and reliable results compared to YOLOv5s. However, it is important to consider the trade-off between performance and resource efficiency, as YOLOv5x requires more memory and has a longer training time compared to YOLOv5s.

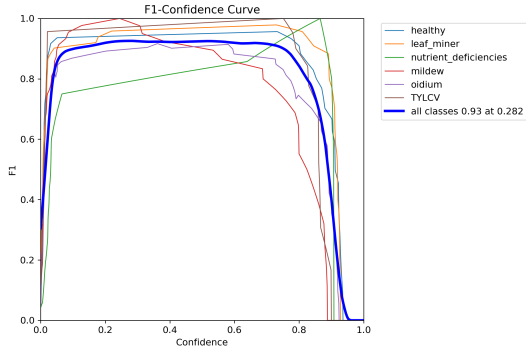
These values were graphically represented in the form of training loss, accuracy metrics, and testing or validation loss, for the two models as depicted in Fig. 6 and Fig. 7.

### 5.3.2 Training Performance of Leaf Disease Detection

The F1 score is a statistical measure commonly used to evaluate performance. It provides a balance between precision and recall by considering both metrics. In our study, we utilized the F1 score to evaluate the leaf disease detection capabilities of the YOLOv5s and YOLOv5x models. Fig. 8 displays the F1 curve for training the models, illustrating the relationship between the confidence score (x-axis) and the corresponding F1 score (y-axis). Each class's F1 scores are depicted in different colors, with the overall F1 score across all classes represented in blue. For the YOLOv5s model (left graph), the highest F1 score of 0.72 is attained at a confidence score of 0.104. This implies that at this threshold, the model achieves a good balance of precision and recall in detecting leaf diseases. The F1 scores for different classes vary, suggesting varying levels of accuracy for each specific disease. On the other hand, the YOLOv5x model (right graph) exhibits a significantly higher F1-score. The peak F1-score of 0.93 is achieved at a confidence score of 0.282, indicating a higher level of accuracy in detecting leaf diseases compared to the YOLOv5s model. The improved F1-score suggests that the YOLOv5x model has a better trade-off between precision and recall, leading to more reliable and accurate predictions across different disease classes.




(a) YOLOv5s.

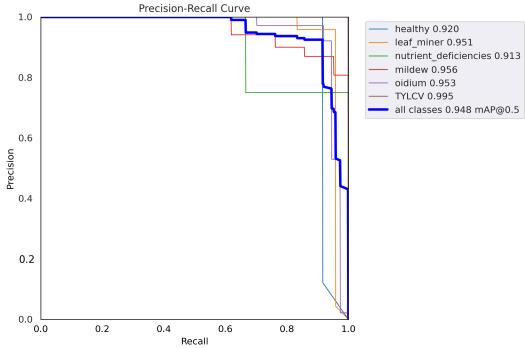



(b) YOLOv5x.

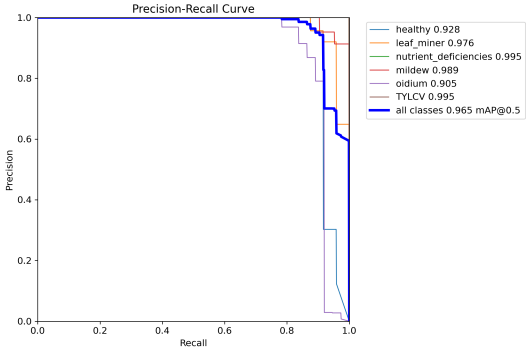
**Fig. 8**: F1 Curves for Leaf Disease Detection.

Fig. 9 shows the precision-recall (PR) curves for YOLOv5s and YOLOv5x to further evaluate the models' performance. At various probability thresholds, these curves depict the trade-off between accuracy and recall. Each curve

represents a separate model, as seen in (a) YOLOv5s and (b) YOLOv5x. The recall grows while accuracy drops when the probability threshold decreases in the left graph (a), which corresponds to YOLOv5s. This means that at lower thresholds, YOLOv5s is more likely to discover positive cases but may also generate more false positives.




(a) YOLOv5s.




(b) YOLOv5x.

**Fig. 9**: Precision-Recall Curves.

At an mAP of 0.5, the PR-score for YOLOv5s is measured at 0.948, demonstrating its performance in achieving a balance between precision and recall. On the other hand, the right graph (b) represents the PR curve for YOLOv5x. Similar to YOLOv5s, as the probability threshold decreases, recall increases and precision decreases. However, YOLOv5x exhibits a higher overall performance compared to YOLOv5s. At an mAP of 0.5, the PR-score for YOLOv5x is measured at 0.965, indicating its superior precision-recall trade-off and stronger detection capabilities. The PR curves provide valuable insights into the models' ability to balance precision and recall at different probability thresholds. The results emphasize the improved performance of YOLOv5x over YOLOv5s, showcasing its higher precision and recall values in detecting leaf diseases.

The detection results for disease detection using the YOLOv5s and YOLOv5x models are shown in Fig. 10. This figure presents the images after the prediction phase, where the models have identified and localized the disease regions on the plant leaves. Each image in the figure depicts a different instance of disease detection, with the detected regions highlighted by bounding boxes. The labels that accompany the bounding boxes describe the diseases that the models discovered. The Figure not only visualizes the illness identification findings but also offers precise information for each bounding box prediction. The accuracy indicates how confident the model is in its forecast for each illness event. This data enables us to measure the dependability of the models' disease-detection skills. By examining the accuracy values linked with the bounding boxes, we can determine how well the models perform in correctly detecting and localizing various types of diseases on the leaves. In summary, this visualization presents a thorough overview of the illness detection process, highlighting disease region localization and offering insights into the accuracy of the model's predictions. It represents the models' effectiveness in identifying plant leaf diseases visually.

## 5.4 Comparative Study

A comparison study was performed using the mean Average Precision (mAP) measure to evaluate the performance of the suggested technique in the context of leaf disease detection. Table 6 compares mAP values obtained using the proposed approach to those acquired from recent studies in the field. The mAP is a reliable measure of the model's capacity to detect and classify leaf diseases, with larger values suggesting better performance.

Dai et al.[2] proposed the YOLOv5-CAcT network for crop leaf disease detection using the PlantVillage dataset, which is based on the YOLOv5 model. The mean Average Precision (mAP) score for this approach was 95.6%. Fang

**Table 6**: Comparative study

| Ref | Model | Dataset | mAP (%) |
|---|---|---|---|
| [2] | YOLOv5-CAcT | 51721 images PlantVillage | 95.6% |
| [5] | Based on YOLOv5 model | 1170 images Self-generated | 95.24% |
| [19] | Based on YOLOv5 model | 4000 images Plantvillage | 90.7% |
| [18] | Based on YOLOv5 model | 4353 images Plantvillage | 95.2% |
| [27] | GMLDD Based on YOLOv5 model | 2542 images Self-generated | 71.0% |
| [41] | YOLOv5-CA model | 820 images Self-generated | 90.2% |
| **Proposed Approach** | **YOLOv5x model** | **3640 images Self-generated** | **96.5%** |

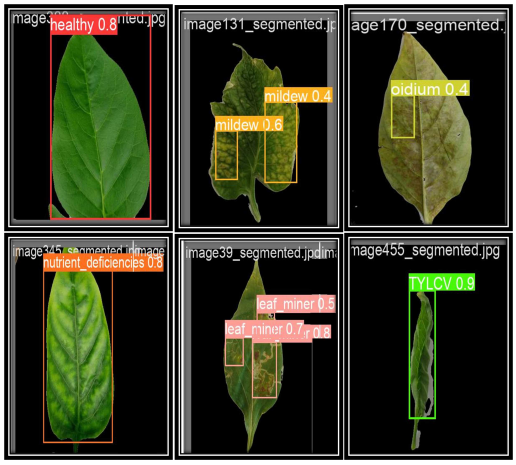
et al.[5] presented an improved method for detecting wormholes in soybean leaves that incorporates the YOLO-v5s network. Using a self-generated dataset consisting of 1170 images, the model achieved a mAP of 95.24%. Mathew et al.[19] used YOLOv5s model for the detection of bell pepper leaf diseases. By leveraging a dataset comprising 4000 images sourced from the PlantVillage dataset, the model reached a mAP of 90.7%. Ma et al.[18] developed an enhanced variant of the YOLOv5n model, denoted as (CTR_YOLOv5ns), which integrates a Coordinate Attention (CA) mechanism and a Swin Transformer (STR) detection head, for the detection of maize leaf diseases. Using a dataset of 4353 images from PlantVillage dataset, the approach achieved a mAP of 95.2%. Rashid et al.[27] presented the Guava Leaf Disease Detection (GMLDD) model, based on the YOLOv5 architecture, for the identification of different diseases in guava leaves. To evaluate its performance, the authors utilized a self collected dataset consisting of 2542 images. The GMLDD model achieved a mean Average Precision (mAP) of 71.0%. Zhang et al.[41] proposed a detection model called (YOLOv5_CA), which combines the YOLOv5 architecture with a coordinate attention mechanism, specifically designed for the detection of grape downy mildew disease in plant leaves. The authors trained the model using a self-generated dataset comprising 820 images. The (YOLOv5_CA) model achieved a mean Average Precision (mAP) of 90.02%.

Comparing the performance measures of several studies, we can see that the suggested strategy regularly outperforms the other methods. This demonstrates the suggested approach's efficacy and resilience in properly detecting illnesses from the supplied datasets. The suggested approach's high mAP score shows its capacity to detect and classify diseases effectively, making it a potential option for disease detection in the domain of interest.

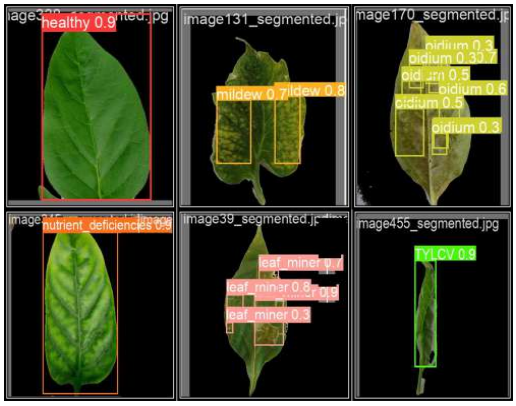
# 6 Conclusion and Future Work

## 6.1 Conclusion

In this research, a method for detecting and localizing tomato and pepper leaf diseases based on




(a) YOLOv5s model




(b) YOLOv5x model

**Fig. 10**: Detection Results: (a) YOLOv5s model, (b) YOLOv5x model

the YOLOv5 model is provided. To improve the model's accuracy and performance, a backdrop removal segmentation method is used to improve the clarity and visibility of the leaf structures and to focus entirely on the leaf areas, eliminating any potential interference or noise. In addition, data augmentation techniques were used to improve the sample dataset. Furthermore, the original targeting frame was optimized using k-means++ clustering, which improved the detecting process even further. In ablation study, the performance of the YOLOv5s and YOLOv5x models was compared using the mAP metric. The results revealed that the YOLOv5x model achieved a higher mAP value of 96.5% compared to the YOLOv5s model which demonstrated a performance of 94.8%. These findings indicate that the YOLOv5x model outperforms the YOLOv5s model in terms of overall detection accuracy. It should be noted, however, that the YOLOv5s model requires significantly less memory and has a shorter training time compared to the YOLOv5x model.

### 6.2 Future Work

Future work might involve investigating optimization and compression techniques like quantization and binarization to construct a disease detection methodology on an FPGA platform. The purpose of using these technologies is to improve the efficiency and resource consumption of the system on an FPGA while maintaining a high degree of accuracy. Quantization and other optimization strategies can reduce the accuracy of model weights and activations, resulting in reduced processing needs and a smaller memory footprint. Binaries approaches, on the other hand, try to express model parameters and calculations using binary numbers, which leads to additional resource minimization. Implementing such technologies on an FPGA platform can result in shorter inference times and lower power consumption, making the disease detection system more suited for deployment in resource-constrained contexts. This future work will involve investigating the effect of different optimization and compression techniques on system performance, evaluating trade-offs in terms of accuracy and resource utilization, and fine-tuning them to achieve optimal balance.

**Funding Statement** The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.
**Data Availability** Data are available on request from the authors.

## Declarations

**Conflict of interest** The authors have no conflicts of interest.
**Ethical Statement** No ethical issue in this paper.

19

# References


[1] Alberta Odamea Anim-Ayeko, Calogero Schillaci, and Aldo Lipani. Automatic blight disease detection in potato (solanum tuberosum l.) and tomato (solanum lycopersicum, l. 1753) plants using deep learning. *Smart Agricultural Technology*, page 100178, 2023.

[2] Guowei Dai and Jingchao Fan. An industrial-grade solution for crop disease image detection tasks. *Frontiers in Plant Science*, 13: 2012, 2022.

[3] Guowei Dai, Lin Hu, Jingchao Fan, et al. Da-actnn-yolov5: Hybrid yolo v5 model with data augmentation and activation of compression mechanism for potato disease identification. *Computational Intelligence and Neuroscience*, 2022, 2022.

[4] Guowei Dai, Jingchao Fan, Zhimin Tian, and Chaoyu Wang. Pplc-net: Neural network-based plant disease identification model supported by weather data augmentation and multi-level attention mechanism. *Journal of King Saud University-Computer and Information Sciences*, 35(5):101555, 2023.

[5] Wenbo Fang, Fachun Guan, Helong Yu, Chunguang Bi, Yonggang Guo, Yanru Cui, Libin Su, Zhengchao Zhang, and Jiao Xie. Identification of wormholes in soybean leaves based on multi-feature structure and attention mechanism. *Journal of Plant Diseases and Protection*, 130(2):401–412, 2023.

[6] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[7] Alex Graves and Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012.

[8] Hemant Kumar Gupta and Hare Ram Shah. Deep learning-based approach to identify the potato leaf disease and help in mitigation using iot. *SN Computer Science*, 4(4):333, 2023.

[9] R Ilahy, T R'him, I Tlili, J Hager, et al. Effect of different shading levels on growth and yield parameters of a hot pepper (capsicum annuum l.) cultivar'beldi'grown in tunisia. *Food*, 7(Special Issue 1):32–35, 2013.

[10] Siddhi Jain, Rahul Sahni, Tuneer Khargonkar, Himanshu Gupta, Om Prakash Verma, Tarun Kumar Sharma, Tushar Bhardwaj, Saurabh Agarwal, and Hyunsung Kim. Automatic rice disease detection and assistance framework using deep learning and a chatbot. *Electronics*, 11(14):2110, 2022.

[11] Glenn Jocher. YOLOv5 by Ultralytics, May 2020. URL https://github.com/ultralytics/yolov5.

[12] Glenn Jocher, Alex Stoken, and Jirka Borovec. Nanocode012. *Kwon, Y*, 2021.

[13] Hyun-Ki Jung and Gi-Sang Choi. Improved yolov5: Efficient object detection using drone images under various conditions. *Applied Sciences*, 12(14):7255, 2022.

[14] Saim Khalid, Hadi Mohsen Oqaibi, Muhammad Aqib, and Yaser Hafeez. Small pests detection in field crops using deep learning object detection. *Sustainability*, 15(8):6815, 2023.

[15] Huishan Li, Lei Shi, Siwen Fang, and Fei Yin. Real-time detection of apple leaf diseases in natural scenes based on yolov5. *Agriculture*, 13(4):878, 2023.

[16] Shasha Li, Yongjun Li, Yao Li, Mengjun Li, and Xiaorong Xu. Yolo-firi: Improved yolov5 for infrared image object detection. *IEEE access*, 9:141861–141875, 2021.

[17] Ji Lin, Di Bai, Renjie Xu, and Haifeng Lin. Tsba-yolo: An improved tea diseases detection model based on attention mechanisms and feature fusion. *Forests*, 14(3):619, 2023.

[18] Li Ma, Qiwen Yu, Helong Yu, and Jian Zhang. Maize leaf disease identification based

on yolov5n algorithm incorporating attention mechanism. *Agronomy*, 13(2):521, 2023.

[19] Midhun P Mathew and Therese Yamuna Mahesh. Leaf-based disease detection in bell pepper plant using yolo v5. *Signal, Image and Video Processing*, pages 1–7, 2022.

[20] Marriam Nawaz, Tahira Nazir, Ali Javed, Momina Masood, Junaid Rashid, Jungeun Kim, and Amir Hussain. A robust deep learning approach for tomato plant leaf disease localization and classification. *Scientific Reports*, 12(1):18568, 2022.

[21] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *18th international conference on pattern recognition (ICPR'06)*, volume 3, pages 850–855. IEEE, 2006.

[22] Thanh-Hai Nguyen, Thanh-Nghia Nguyen, and Ba-Viet Ngo. A vgg-19 model with transfer learning and image segmentation for classification of tomato leaf disease. *AgriEngineering*, 4(4):871–887, 2022.

[23] Kshyanaprava Panda Panigrahi, Himansu Das, Abhaya Kumar Sahoo, and Suresh Chandra Moharana. Maize leaf disease detection and classification using machine learning algorithms. In *Progress in Computing, Analytics and Networking: Proceedings of ICCAN 2019*, pages 659–669. Springer, 2020.

[24] Rutuja Rajendra Patil, Sumit Kumar, Shwetambari Chiwhane, Ruchi Rani, and Sanjeev Kumar Pippal. An artificial-intelligence-based novel rice grade model for severity estimation of rice diseases. *Agriculture*, 13(1): 47, 2022.

[25] Yingshu Peng and Yi Wang. Leaf disease image retrieval with object detection and deep metric learning. *Frontiers in Plant Science*, 13, 2022.

[26] B Rajesh, M Vishnu Sai Vardhan, and L Sujihelen. Leaf disease detection and classification by decision tree. In *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*, pages 705–708. IEEE, 2020.

[27] Javed Rashid, Imran Khan, Ghulam Ali, Shafiq ur Rehman, Fahad Alturise, and Tamim Alkhalifah. Real-time multiple guava leaf disease detection from a single leaf using hybrid deep learning technique. *CMC-COMPUTERS MATERIALS & CONTINUA*, 74(1):1235–1257, 2023.

[28] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[29] Amani Romdhane, Anissa Riahi, Gabriella Piro, Marcello Salvatore Lenucci, and Chafik Hdider. Agronomic performance and nutraceutical quality of a tomato germplasm line selected under organic production system. *Horticulturae*, 9(4):490, 2023.

[30] Tamas Roska and Leon O Chua. The cnn universal machine: an analogic array computer. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 40 (3):163–173, 1993.

[31] Muhammad Hammad Saleem, Sapna Khanchi, Johan Potgieter, and Khalid Mahmood Arif. Image-based plant disease identification by deep learning meta-architectures. *Plants*, 9(11):1451, 2020.

[32] YY Shang, QR Zhang, and HB Song. Application of deep learning using yolov5s to apple flower detection in natural scenes. *Trans. Chin. Soc. Agric. Eng*, 9:222–229, 2022.

[33] Ravshan Shirinboyev, Jahongirjon Bobolov, et al. Image segmentation by otsu method. *International Journal of Contemporary Scientific and Technical Research*, (Special Issue):64–72, 2023.

[34] Chinna Gopi Simhadri and Hari Kishan Kondaveeti. Automatic recognition of rice leaf diseases using transfer learning. *Agronomy*, 13(4):961, 2023.

[35] Jaskaran Singh and Harpreet Kaur. Plant disease detection based on region-based segmentation and knn classifier. In *Proceedings of the International Conference on ISMAC in Computational Vision and Bio-Engineering 2018 (ISMAC-CVB)*, pages 1667–1675. Springer, 2019.

[36] Md Janibul Alam Soeb, Md Fahad Jubayer, Tahmina Akanjee Tarin, Muhammad Rashed Al Mamun, Fahim Mahafuz Ruhad, Aney Parven, Nabisab Mujawar Mubarak, Soni Lanka Karri, and Islam Md Meftaul. Tea leaf disease detection and identification based on yolov7 (yolo-t). *Scientific reports*, 13(1):6078, 2023.

[37] Emilija Strelcenia and Simant Prakoonwit. A survey on gan techniques for data augmentation to address the imbalanced data issues in credit card fraud detection. *Machine Learning and Knowledge Extraction*, 5(1):304–329, 2023.

[38] Bin Yan, Pan Fan, Xiaoyan Lei, Zhijie Liu, and Fuzeng Yang. A real-time apple targets detection method for picking robot based on improved yolov5. *Remote Sensing*, 13(9): 1619, 2021.

[39] Abu Sarwar Zamani, L Anand, Kantilal Pitambar Rane, P Prabhu, Ahmed Mateen Buttar, Harikumar Pallathadka, Abhishek Raghuvanshi, and Betty Nokobi Dugbakie. Performance of machine learning and image processing in plant leaf disease detection. *Journal of Food Quality*, 2022:1–7, 2022.

[40] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[41] Zhao Zhang, Yongliang Qiao, Yangyang Guo, and Dongjian He. Deep learning based automatic grape downy mildew detection. *Frontiers in Plant Science*, 13, 2022.