PROJECT

Tomato Leaf Disease Detection Using YOLOv5 Model

Lecturer: Dr. Vo Bich Hien

Group members:

Vo Nguyen Minh Duy - 17105

Nguyen Huynh Trung Hieu - 17727

Tran Hoang Kim - 17251

December 2023

I. Introduction:

In the world, tomatoes are among the most important and widely consumed crops. The amount of tomatoes varies based on the method of fertilization. One of the main factors affecting crop production quantity and quality is leaf disease. Tomato leaves are vulnerable to many common diseases caused by fungi, bacteria, mildew, viruses and so on. Most common diseases are caused by types of fungus that favor certain weather conditions. Excessive rain during cool or warm periods creates an ideal environment. Therefore, early detection and diagnosing the signal of specific diseases appearing on the leaves is important for finding sufficient treatments. One of the reasonable methods is applying the artificial machine learning technique to identify the problems with tomato leaves. The Convolutional Neural Network (CNN) is used to accurately describe and categorize tomato illnesses. The entire experiment is carried out using the YOLOv5 model on Google Colab to detect four different diseases, namely Bacteria spot, Early blight, Late blight, Powdery mildew as well as healthy leaves. The dataset has 6380 images, which include 5480 Internet images and 900 Field images which were collected in tomato gardens in Da Lat city.

II. Teamwork:

Nr.	Name	Student ID	Tasks done
1	Vo Nguyen Minh Duy	17105	- Label images of all diseases and healthy leaves. - Train and detect Early blight and Late blight leaves. - Improve the model to get higher detection accuracy between Early blight and Late blight leaves. - Prepare for the trip: practice taking photos of leaves in university. - Go to Da Lat city to collect data. - Summarize and report the process of Field data.
2	Nguyen Huynh Trung Hieu	17727	- Label images of all diseases and healthy leaves. - Find the tool to label images.

			- Find the training model. - Test model with Internet data. - Train and detect Bacterial spot and healthy leaves. - Augment the Field data.
3	Tran Hoang Kim	17251	- Manage tasks on Notion. - Label images of all diseases and healthy leaves. - Find the places to collect data and plan the field trip. - Discuss with the teacher to arrange the field trip. - Prepare for the trip: research how to collect real data, and practice taking photos of leaves in university. - Go to Da Lat city to collect data. - Summarize and report the result of Field data. - Train and detect Powdery mildew leaves.

III. Methodology:

1. Dataset:

Data is essential for most machine-learning algorithms to produce more accurate detection and better training outcomes. Higher overall accuracy is often achieved in classification and object detection when working with a larger data set. A good dataset can also be determined based on its variation and balance in the number of images among the classes. In this report, we have used two types of data, namely Internet data and Field data.

a) Source of Internet data:

The Internet dataset used for the project is Tomato Leaves Dataset, which collected on the Kaggle platform [7]. The main purpose of this dataset is to provide material for developing a lightweight model that can detect or classify various tomato leaf diseases.

b) Source of Field data:

Additionally, our team made a field trip to tomato gardens in Da Lat city to take an amount of tomato leaf images, which serve as an extension for the current dataset. A new disease class named Powdery mildew was added after the trip because of plentiful Field data. This dataset provides more variety and reality factors to the model, resulting in more accuracy in different types of input.

c) Type and amount of data:

The dataset in the report has 6380 images which include 5480 Internet images and 900 Field images. In the Internet dataset, there are a total of eleven different common diseases on tomato leaves, however, this project will only cover four types of disease leaf classes, namely Bacteria spot, Early blight, Late blight, Powdery mildew, and a Healthy leaf class. In the Field dataset, there are 568 images of disease and healthy leaves which are collected from four places in Da Lat city: Green Box Cafe, Yersin University of Da Lat, HiFar, and VietFarm. For an effective training process, the image augmentation method is used to extend the Field dataset up to 900 images in total. Table I shows the precise number of Internet and Field images included in each type of class.

Types of class	**Number of Internet images**	**Number of Field images**
Bacterial spot	1100	100
Early blight	1520	300
Late blight	1520	100
Powdery mildew	20	300
Healthy	1320	100

Table I. Table of dataset.

Lastly, the reason for only working on 5 classifications and processing with a few Field data are the restrained resources, limited time and risks. However, the value of this model is still decent in terms of modern agriculture technology.

d) Process of collecting Field data:

Step 1: Skim the garden to verify the presence of leaf diseases in it

- Preferably, two or three rows of tomato trees in a small-scale garden.
- It may extend up to five rows or more if the garden size is the size of a farm.

- We quickly identify the kinds of diseases commonly appearing across the garden.

Step 2: Prioritize the diseases

- Bacterial spot disease comes first since, among the list, it is the most distinctive sickness we research.
- Early blight or Late blight disease is optional since both are rare to find and hard to distinguish.
- Healthy leaf comes last after gathering sufficient data for other diseases since they are abundant to find and capture.

Step 3: Observe each row visited earlier to get a more thorough look at each leaf

- Determine which diseases an interested leaf might have since a leaf can have symptoms of various diseases at once.

Step 4: Once decided, utilize the camera to focus on the infected leaf to capture

- Modify the lighting to get the clearest image possible.
- Focus the leaf to be as big and close as possible, covering the whole camera frame.
- Blur the background image to avoid any confusion from the model.

e) Dataset sizes:

Table II shows the specific number of images included in each specific dataset. The images in the datasets were categorized as disease or healthy.

Type	Number of Internet images	Number of Field images
Training dataset	4600	540
Testing dataset	100	270
Validation dataset	780	90
Total number of dataset	5480	900

Table II. Table of dataset sizes.

2. Label images:

Before any training progress begins, data or images must be entirely labeled, so that every label must provide the necessary detail about the class and the location of the concerned object in the image. There are several basic types associated with labeling an image, including Bounding Boxes, Polygon Segmentation, Semantic Segmentation, 3D Cuboids, Key-Point and Landmark,

Lines, and Splines, therefore, depending on the target of each situation, the suitest type is chosen. YOLOv5 is most effective when dealing with Bounding Boxes annotations, which was the chosen label method for this project.

a) Label tool:

LabelImg is a graphical image annotation written in Python, the tool was used by our group for the labeling tasks. Annotations were saved in YOLO format to support both consistency and configuration.

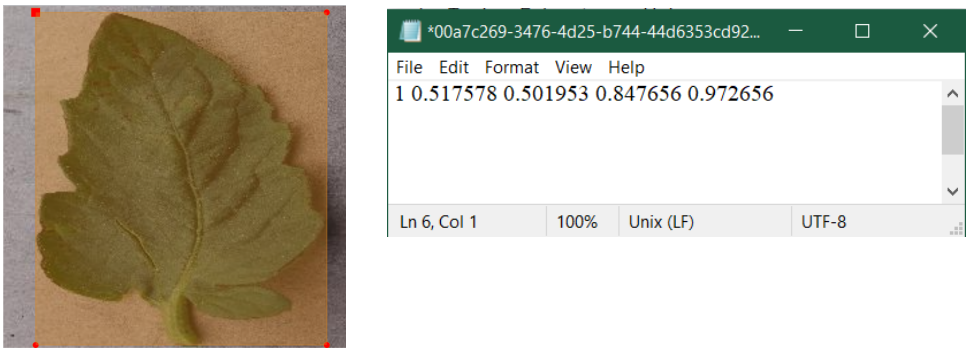

Figure 1. The image has been labeled (left) and the class ID as well as the label position (right).

b) Technique:

To maximize the precision and training performance, each image must be labeled carefully, avoiding covering unrelated parts and focusing only on the leaf. Because the project is about distinguishing between healthy and diseased leaves, every leaf must be completely fixed in the bounding box. However, there were still some odd images that some leaves spanned too much, so we just tried to focus on the disease and part of a leaf.

3. Training model using YOLOv5:

YOLOv5 is described as a powerful object detection algorithm developed by Ultralytic. Some specific advantages of YOLOv5 are its speed and accuracy in object detection, classification, capable of processing images in real time. Importantly, YOLOv5 is built on the PyTorch framework, making it easy for developers to use and deploy. Here, we will have a sketch of the overall architecture of this model.

a) Overview:

YOLOv5's architecture consists of three main parts:

- Backbone: Mainbody of the network, using New CSP-Darknet53 structure, a modification of Darknet architecture.
- Neck: This part is a connection between the backbone and the head of YOLOv5.

- Head: The last sections from the network, are responsible for generating the final output for each image.

b) Detail of model:

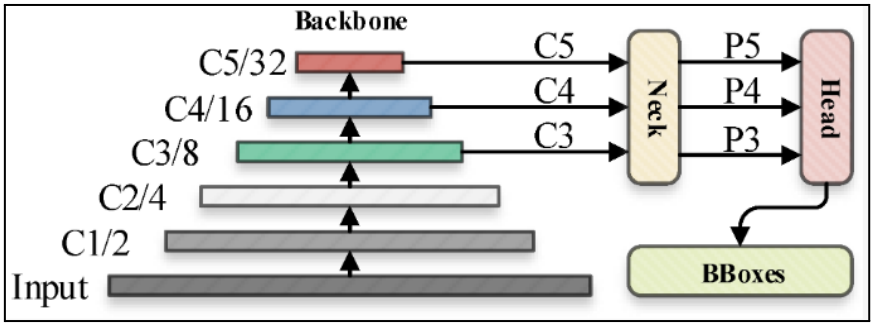

Figure 2. The default interference flowchart of YOLOv5.

The image was processed through an input layer (input) and sent to the backbone for feature extraction. By applying a series of convolutional layers, the backbone can analyze the images on different scales and levels of abstraction, capturing crucial visual patterns and structure. Feature extraction is important to improve training performance, reduce computation costs, and a better understanding of data.

After getting feature maps from differentiation sizes, these features are fused through a feature fusion network (neck) to generate three feature maps P3, P4, and P5 (in the YOLOv5, the dimensions are expressed with the size of 80x80, 40x40, and 20x20) to detect small, medium and large objects in the picture, respectively.

Then, the three feature maps were sent to the prediction head (head), where confidence calculation and bounding-box regression were executed for each pixel in the feature map using the preset anchor, the output of these steps is a multidimensional array or a BBoxes (bounding boxes), which contain information of the object class, class confidence, box coordinate, width, and height.

4. Training and interpreting:

a) Training Process:

All training processes are conducted on the Google Colab coding platform, where we are granted access to a temporary coding session with T4 GPU as virtual computing hardware, as well as a file explorer in order to train the model. Priorly, the training dataset is uploaded to the session's explorer as a zip file before extracting it to a folder. Additionally, a file called path.yaml is created in order to contain the directories for training samples, testing samples, and validating samples of the training dataset.

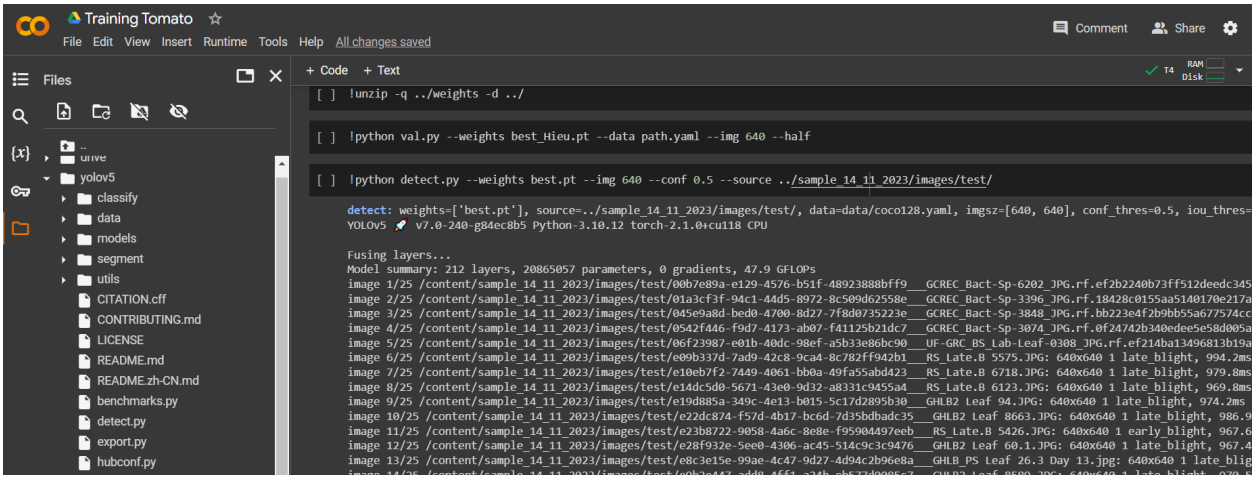

Figure 3. Google Colab workspace with YOLOv5 main directory (on the left) and coding environment (on the right).

The initial attempt includes training a new model based on two of YOLOv5's lightweight models, yolov5s.pt or yolov5m.pt, for 300 epochs, which will repetitively train the model on a whole dataset for 300 times. The following options are also declared in every attempt in order to customize the training:

- *–img 640*: allow YOLOv5 to automatically resize all the images down to an image size of 640 x 640 before training. This makes the images equally sized for the model to easily learn and extract necessary features from each group of pixels.
- *–batch 32*: split the training images into different image batches containing 16 images on each batch. For each optimizing step in an epoch, a batch is evaluated by the model to calculate the weight and optimize the loss of the results. All batches are iterated through when an epoch ends.
- *–epochs 300*: allow the model to be re-trained 300 times. On each epoch, the previous model is used to fit different batches of images to optimize the weights further while slowly minimizing the loss between the results and labels in the batch. By default, the process will automatically stop when for the last 100 epochs, all loss values between epochs are similar to each other or do not linearly decrease, thus marking the model the most optimized model it can be trained given the dataset and training conditions.
- *–cache disk*: allow YOLOv5 to cache the images to the session's file storage, which significantly increases the training speed.

```
1 !python train.py --img 640 --batch 32 --epochs 300 --data path.yaml --weights bestDemo.pt --cache disk --save-period 50
```

Figure 4. Typical command to initialize the training of a pre-trained model to a new dataset.

b) Interpret the result and test:

When the training is completed, the results are by default logged to the directory runs/exp/train/exp. To determine how well the model is doing, we are interested in reading the measurements such as loss value, precision value, recall value, and F1 value. All interested values are logged inside the result folders under different formats such as graphs and confusion matrices. YOLOv5 logs the following types of loss value for each epoch the model is trained:

- Box_loss: the difference between the predicted box and the actual labeled box of an image. The lower the value, the closer the predicted bounding box is to the actual bounding box.
- Cls_loss: the difference between the predicted class and the actual class of an image. The lower the value, the higher chance the predicted class is the same as the actual class.
- Obj_loss: the difference between object position in the predicted box and object position in the actual box of an image. The lower the value, the closer the object's position to the actual position of the actual object.

During finalizing the training process, YOLOv5 performs a validating process on a chosen amount of images, the results of which are also recorded inside the folder. Based on the pair of predicted class and actual class for each image, the confusion matrix is formed by counting these pairs and dividing them into 4 categories:

- True-Positive pair: the Predicted class and the Actual class share same class 1.
- False-Positive pair: the Predicted class is class 1 while the Actual class is class 0.
- False-Negative pair: the Predicted class is class 1 while the Actual class is class 0.
- True-Negative pair: the Predicted class and the Negative class share the same class 0.

Additionally, the metric Accuracy, Precision, Recall, and F1-score are calculated during each training epoch based on the value of Positive and Negative pairs:

- Accuracy: indicate how good the model is at making correct class predictions to the images. The higher the value, the more accurate the model predicts the correct classes of the labeled images.
- Precision: indicate how good the model is at predicting Positive classes for the images. The higher the value, the more accurately the model predicts the positive classes of the labeled images while rarely mistaking Positive predictions on images labeled with Negative classes.
- Recall: indicate how good the model is at predicting images labeled with Positive classes. The higher the value, the more accurately the model predicts the positive classes of the

labeled images while rarely mistaking Negative predictions on images labeled with Positive classes.

Finally, the model's effectiveness can also be determined based on the images it is fed to predict their class, without knowing the labeled classes. Typically, a predicted image comes with a class name and a confidence number that indicates how certain it is about the predicted class. The scale starts as low as 0.00 (not certain at all) to 1.00 (very certain) as the highest value.

Figure 5. An image is predicted to be 93% of Bacterial spot class.

IV. Results:

1. First attempt - 2 classes:

The first training attempt serves as the accustomation to train the model using the YOLOv5 algorithm on Google Colab. The data used consists of two classes: Bacterial spot and Healthy.

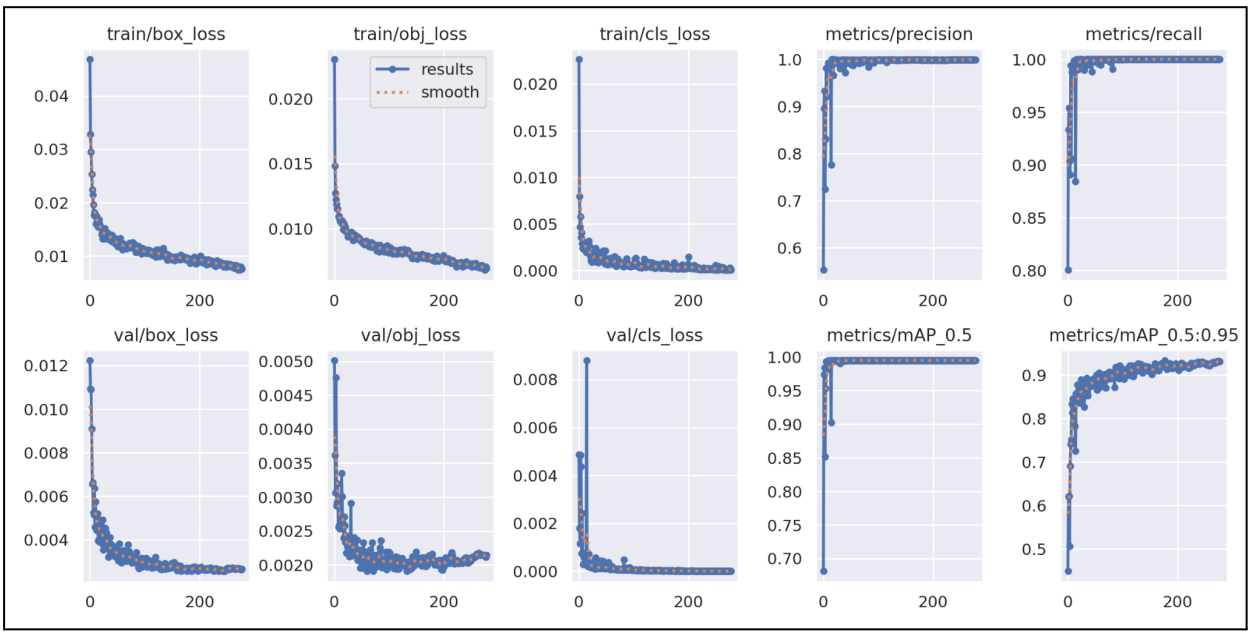

Figure 6. File results.png graphing the changes of loss metrics, precision metrics, and recall metrics throughout 250 epochs during training and validating.

The above figure illustrates the change of loss value during training to be very smooth and hardly fluctuating, due to the data simplicity. Additionally, the graph of class loss (cls_loss) plummets at very near epoch 0, indicating that the provided yolov5m.pt is getting accustomed to the image pattern quickly and effectively. Beyond the sharp drop, the line goes on gradually with a slight decrease, implying that the model is slowly adjusting its parameters to be as optimized as possible, with the cls_loss value reaching just slightly more than 0.000. This can be regarded as the main reason why the model is stopped at epoch 250 instead of epoch 300 as instructed since the model was observed to see no more significant improvement for the last 100 epochs. Meanwhile, heavy fluctuations in both the val/obj_loss graph and val/box_loss graph indicate that the model has slight difficulty in positioning its prediction box to be as accurate to the original labeled box on the object as possible.

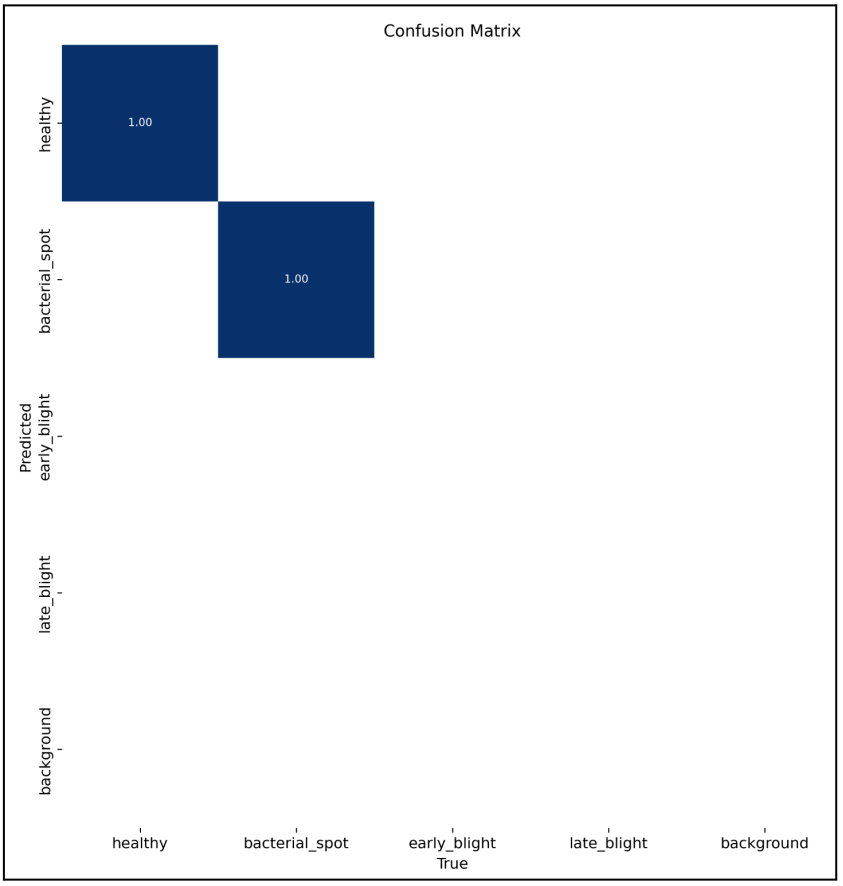

Figure 7. The confusion matrix after validating images of 2 classes.

The above confusion matrix shows 100% accuracy in comparing the label predictions made by the model against the actual label during the validation process. This is rather expected, since the difference between a Healthy leaf and a Bacterial spot leaf is very apparent, and as such, the model's effectiveness is regarded as success in accurately distinguishing between healthy leaves and infected leaves.

2. Second attempt - 3 classes:

This training attempt has added a new disease class called Early blight, which requires the model to make much more effort in distinguishing between either Early blight and Healthy or Early blight and Bacterial spot.

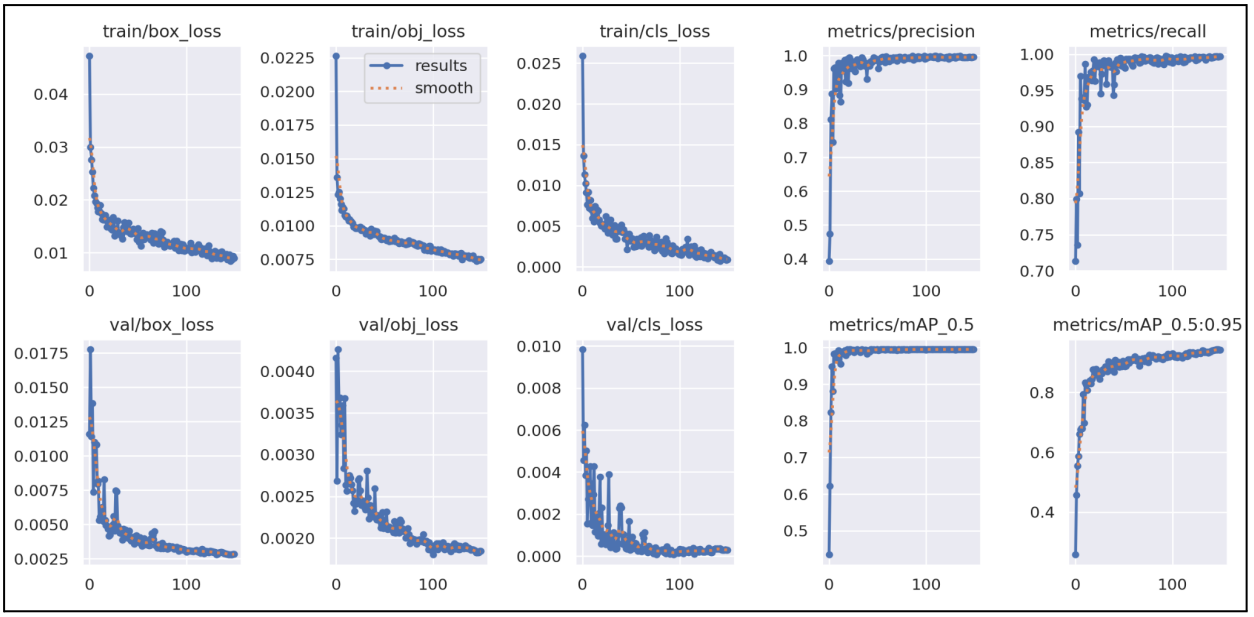

Figure 8. File results.png graphing the changes of loss metrics, precision metrics, and recall metrics throughout 150 epochs during training and validating.

Different from the first attempt, the class_loss graph decreases less significantly after epoch 10 and fluctuates more heavily between each epoch. The final class_loss value can never reach below 0.005 but instead is concluded slightly more than 0.005. Nevertheless, the very low result is maintained ensuring the model has little trouble differentiating between Early blight, Bacterial spot, and Healthy class. Additionally, both the precision graph and recall graph also slightly fluctuate as they increase, and both will begin to stabilize shortly after. Compared to the previous attempt, the val/box_loss graph, val/obj_loss, and val/clss_loss graph are much more stable, despite having some spikes on early epochs. This indicates that the model achieves a very low error while validating during the early stage of training and thus getting more and more accustomed to drawing accurate prediction boxes for the images.

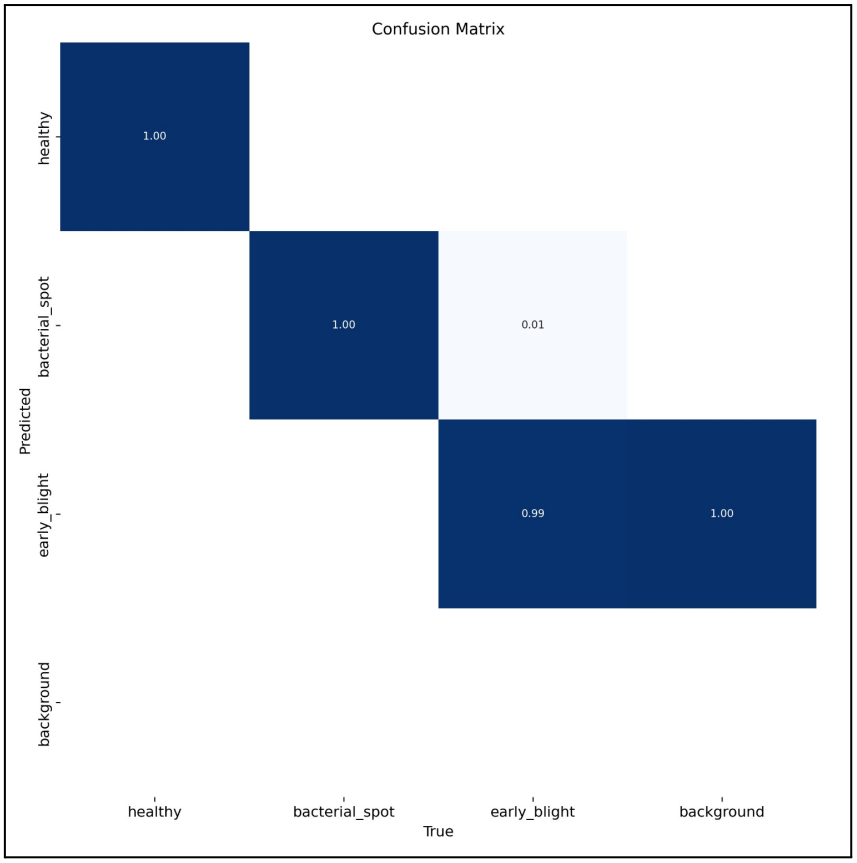

Figure 9. The confusion matrix after validating images of 3 classes.

The confusion matrix comes out perfectly, as it reflects the model's high effectiveness in separating the Early blight class, Bacterial spot class, and Healthy class among the labeled leaves. A 100% accuracy in Healthy leaves is very desirable due to the importance of identifying a healthy leaf among infected leaves in real-life scenarios. The model can be trusted to recognize between Bacterial spot and Early blight diseases through its very high accuracy when predicting leaves labeled with Bacterial spot and leaves labeled with Early blight. However, there exists 1% of Early blight images are incorrectly labeled as Bacterial spot due to the high skewness of the data. Notably, there present certain Early blight predictions for background images, primarily due to certain leaves in the background of several images recognizably clear enough for the model to evaluate.

3. Third attempt - 4 classes:

The third model adds the Late blight leaf disease to the list. Due to the disease's traits being easily confused with Early blight and Bacterial spot, the expectancy for the model's effectiveness is slightly reduced.

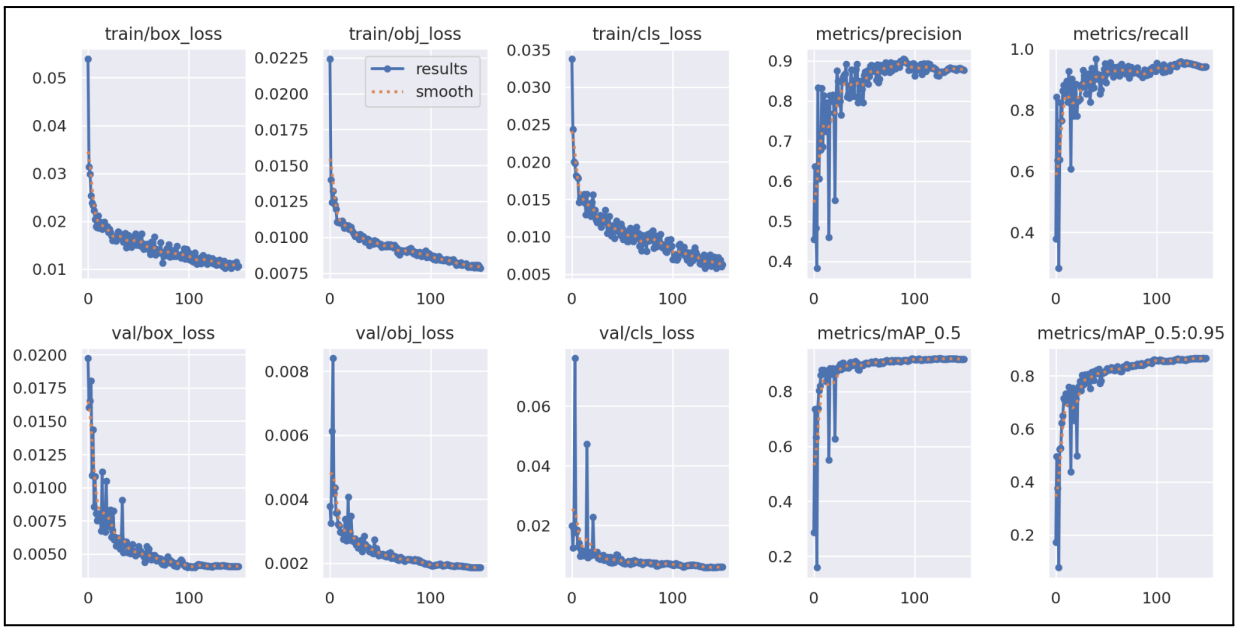

Figure 10. File results.png graphing the changes of loss metrics, precision metrics, and recall metrics throughout 150 epochs during training and validating.

Information from the above figure shows that the cls_loss value starts to fluctuate under the value 0.015, compared to 0.01 of the second model and 0.005 of the first model. Additionally, metrics/precision graphs and metrics/recall illustrate a massive fluctuation in the first 30 epochs before steadily increasing after 50 epochs. This implies the high skewness presented in the data troubles the model during the early validation when the model is still very inexperienced. Contradictorily, the validation graph of both 3 loss values sees an overall stable decrease with a few spikes during early epoch. This partially implies the model still manages to draw accurate prediction boxes with matching labels to the images.

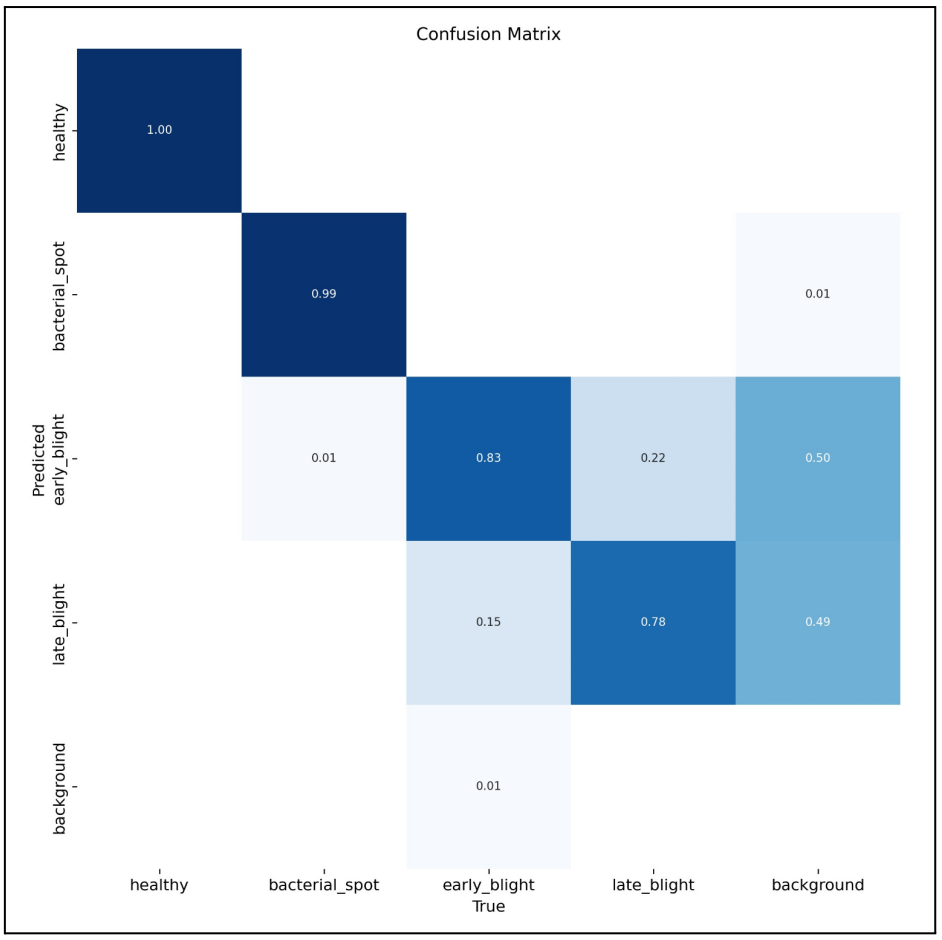

Figure 11. The confusion matrix after validating images of 4 classes.

Compared to the previous attempt's confusion matrix, the matrix from this attempt appears to have some notable mislabeling between Early blight and Late blight. This results in Early blight holding true accuracy of 83% and Late blight having as low as 78%. Also, 1% of Bacterial spot images are mispredicted as Early blight, and several backgrounds across the dataset have been misjudged as Bacterial spot, Early blight, and Late blight. Regardless, Healthy class detection still maintains a 100% accuracy, indicating that the model can separate Healthy leaf images from a pool of infected leaf images with no difficulty at all. Meanwhile, the lower-than-expected accuracies of Early blight and Late blight pose a notable challenge for future optimization and adaption of new classes.

4. Fourth attempt - 5 classes:

The final attempt is adding various Field-labeled images of all previous diseases and healthy leaves into the dataset, as well as an addition of a new disease named Powdery mildew. The main focus of this attempt is to measure how well the model would react against Field images being much more different than Internet counterparts.

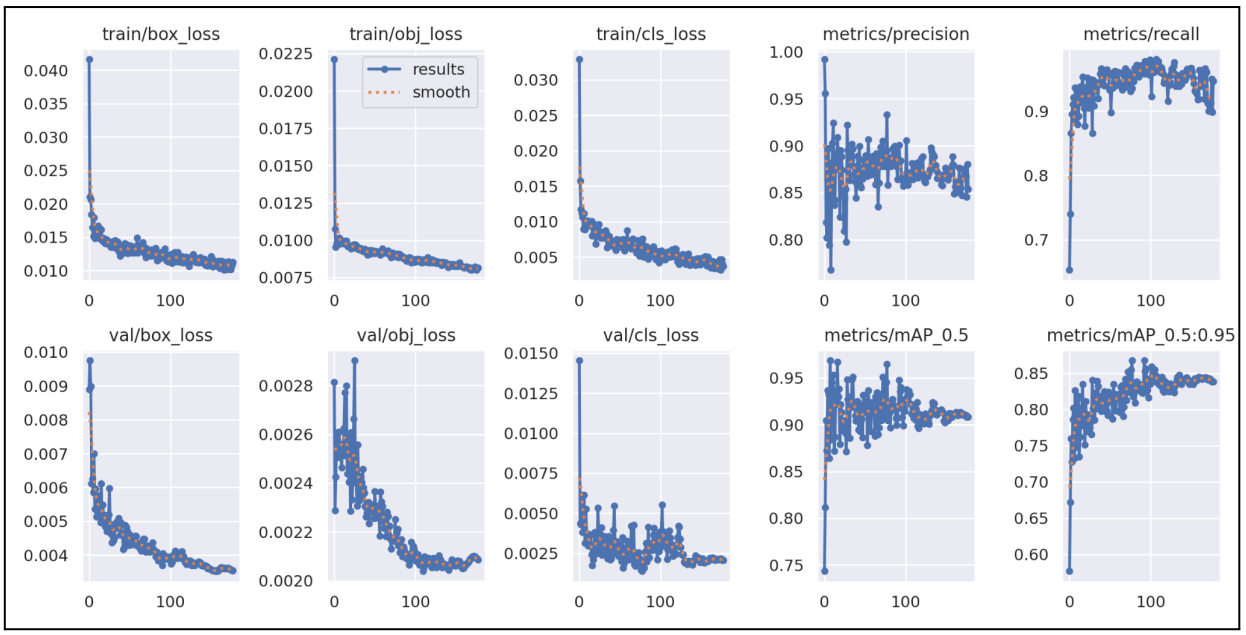

Figure 12. File results.png graphing the changes of loss metrics, precision metrics, and recall metrics throughout 178 epochs during training and validating.

The train/cls_loss illustrates differently from three previous attempts that the loss value massively declines down to as close as 0.010 for the first few epochs, while steadily moving down with slight fluctuation afterward. The graph concludes with a very low cls_loss value of 0.0045, implying that the model is trained effectively with the combination of Internet data and Field data. Furthermore, both metrics/precision graph and metrics/recall graph experience massive fluctuations at different value ranges and epochs; for the metrics/precision graph it is observed to waver between 0.80 and 0.925 at epoch 0 to epoch 100, for metrics/recall graph the value range is 0.87 to 0.97 at epoch 0 to epoch 100. Regardless of said oscillation on both graphs, the precision mean and recall mean are both approximately 0.85 and 0.9, respectively, implying that the model can effectively predict correct classes of the labeled images while rarely mistaking the classes with other classes the images have. The val/obj_loss and val/cls_loss graphs are shown to have slight fluctuations as well, possibly indicating that the model struggles to validate the new Field data.

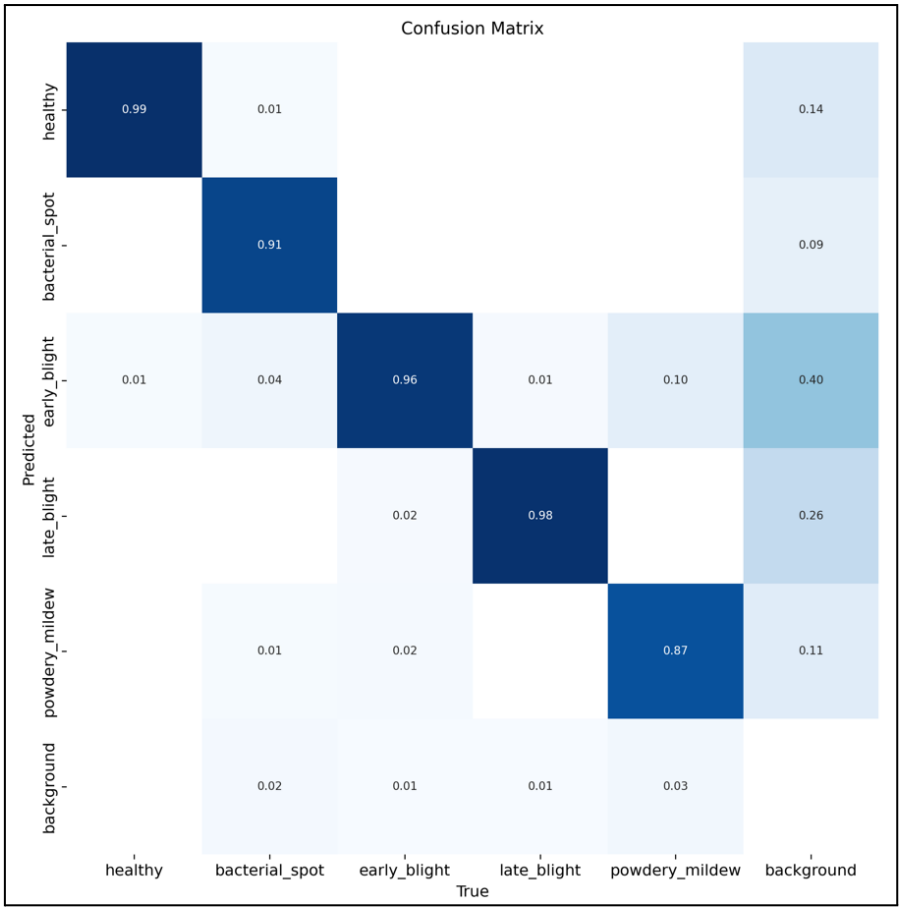

Figure 13. The confusion matrix after validating images of 5 classes.

Illustrated by the above confusion matrix, all 5 classes have very high True Positive detection, showing that the model is well-trained and can manage to accurately predict a large percentage of the dataset of their respective labels. Due to the much smaller amount of training images per class for class Powdery mildew, the accuracy is recorded to be slightly smaller than the rest of the classes. Regardless, the model can still achieve as high as more than 80% accuracy in predicting the Powdery mildew class. Furthermore, the misconfusion between background and defined classes appears more frequently, mainly due to the Field images having unnecessary leaves in the background. There is also slight confusion between the disease classes in terms of validating the Field leaves because some leaves might have been labeled incorrectly from their actual class, or the model hardly recognizes their features to confidently decide their classes.

V. Discussion:

Throughout the processes, the model is viewed to perform differently based on the input data's image variation and the increase in number of new classes. For starters, the model is graded perfectly in separating the healthy leaves from the infected leaves, provided by its near 100% accuracy in predicting the Healthy leaves and strong confidence score when detecting Healthy leaves. As shown in the confusion matrices, the Healthy class frequently keeps the accurate

detection rate as high as 99%, with 1% occasionally confused due to certain images predicted as Early blight instead of Healthy.

Figure 14. Samples of images predicted to be of the Healthy class.

Next, Bacterial spot class is regarded as the most distinctive disease class out of the 4, due to its notable feature of disease on the leaf that can be easily captured, as well as plenty of images in the class. It is illustrated in confusion matrices that the Bacterial spot class is occasionally confused with other classes such as Early blight and Late blight, due to similar features. Early blight and Late blight come next as a pair of diseases that usually are confused with one another, due to little differences in their features, suggested further by rather low confidence scores in the detected image of the classes. Regardless, the model can still achieve close to 80% for both classes, implying that their feature distinction can be extracted and recognizable by the model.

Figure 15. Samples of images predicted to be of Early blight class (left) and Late blight class (right). Confidence score from left to right image: 0.56 and 0.36.

Powdery mildew is also considered as another disease with highly distinctive features, comparable to Bacterial spot. However, due to the lack of images per class compared to the other classes and the data mostly being Field images, Powdery mildew has a validating accuracy of 87%, lower than most classes, as well as is easily mis-detected with other classes and possesses a rather low confidence score.

In practice, the model is expected to be very effective in contributing to on-time disease detection, treatment, and prevention of the tomato trees. Because of the model's excellent accuracy and differentiation in the Healthy class, it can readily discriminate between healthy and diseased tree leaves, possibly recognizing issues on the tomato tree as early and accurately as feasible. Moreover, high accuracy on individual disease classes with very infrequent misconfusion helps diagnose precisely the potential illness presented on the leaves, and possibly of the whole tree. Different diseases require different treatment and prevention methods, and the model's detection speed can potentially tackle the challenge of providing enough information so that the correct solution can be decided and conducted as soon as possible. This will greatly improve the quality-of-life of the tomatoes, effectively increasing the tomato quality.

VI. Conclusion and Future Work:

This project employed the YOLOv5 model to distinguish between healthy and four types of disease tomato leaves. The initial phase was to gather images of tomato leaves, which were subsequently divided into two types: from the internet and collected from the field trip in Da Lat city. The most important phase in preparation was labeling, which had to be done in a way that satisfied the selective neural network used to recognize items and signal an area of interest. According to the testing results, YOLOv5 had the highest classification accuracy among the algorithms, with around 99% in healthy leaf 91% in Bacterial spot leaf, 96% in Early blight leaf, 98% in Late blight leaf, and 87% in Powdery mildew leaf. This method can assist farmers in identifying diseased leaves and prompting the deployment of preventive measures to limit the spread of tomato plant diseases.

Moving forward, this project can be improved with three ideas. First, expanding the capabilities of the AI model by incorporating data on a wider range of diseases. This expanded scope will widen its possible applications and strengthen its standing as a vital diagnostic tool. Second, the model will reduce the accuracy of background detection, which accounts for 40%, in some ways. As a result, the accuracy of some classes would be improved further. Third, there will be a user-friendly web or mobile application. An interface like this would democratize access to AI capabilities, allowing users to quickly acquire correct diagnoses and crucial information.

VII. References:

1. Yang Wu, Lihong Xu, E. D. Goodman (2021). Tomato Leaf Disease Identification and Detection Based on Deep Convolutional Neural Network. Retrieved from https://www.researchgate.net/publication/350551801_Tomato_Leaf_Disease_Identification_and_Detection_Based_on_Deep_Convolutional_Neural_Network

2. A. Verma, M. Agrawal, K. Gupta, A. Jamshed, A. Mishra, H. Khatter, G. Gupta, S. C. Neupane (2022). Plantosphere: Next Generation Adaptive and Smart Agriculture System. Retrieved from https://www.hindawi.com/journals/js/2022/5421312/

3. M. Agarwala, A. Singhb, S. Arjariac , A. Sinhad, S. Guptaa (2020). ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network. Retrieved from https://doi.org/10.1016/j.procs.2020.03.225

4. R. Rajamohanan, B. C. Latha (2023). An Optimized YOLO v5 Model for Tomato Leaf Disease Classification with Field Dataset: Engineering, Technology & Applied Science Research. Retrieved from https://www.etasr.com/index.php/ETASR/article/view/6377/3309

5. Davies, D. (2021). YOLOv5 Object Detection on Windows (Step-By-Step Tutorial). Retrieved from https://wandb.ai/onlineinference/YOLO/reports/YOLOv5-Object-Detection-on-Windows-Step-By-Step-Tutorial---VmlldzoxMDQwNzk4

6. Ultralytics. (n.d.). ultralytics/yolov5: YOLOv5 in PyTorch > ONNX > CoreML > TFLite. Retrieved from https://github.com/ultralytics/yolov5

7. Motwani, A. (2022). Tomato Leaves Dataset. Retrieved from https://www.kaggle.com/datasets/ashishmotwani/tomato/data

8. Nelson, J. (2022). LabelImg for Computer Vision Annotation (Guide). Retrieved from https://blog.roboflow.com/labelimg/