# AI Module

# Contents

## Abstract:

Agriculture is the backbone of a country. It is important to note that without agriculture, there is no economic growth in the country. As technology continues to improve rapidly day by day, these technologies can be utilized in farming and agriculture so that there will be maximum utilization of crops and less wastage of crops. To achieve this, we need to come across a few challenges. Which crops can be grown depending on certain weather conditions?  Identification of disease in crops so that we can prevent it and maximum yield of crops. Prevention is better than cure the famous quote says (1).

But their rapid identification remains difficult in many parts of the world due to the lack of the necessary infrastructure. The combination of increasing global smart phone penetration and recent advances in computer vision made possible by deep learning has paved the way for smart phone-assisted disease diagnosis (2).

Artificial Intelligence is one of the greatest inventions; using AI we can train the machine with images to detect disease in crops. The problem of the underutilization of crops can be addressed (1).

In this thesis, convolutional neural network models were developed to perform plant disease detection and diagnosis using simple leaves images of healthy and diseased plants, through deep learning methodologies. Training of the models was performed with the use of an open database of images, containing different plants of [plant, disease] combinations, including healthy plants. Several model architectures were trained. The significantly high success rate makes the model a very useful advisory or early warning tool, and an approach that could be further expanded to support an integrated plant disease identification system to operate in real cultivation conditions (3).

## Section 1: Introduction:

Agriculture is essential in the country's economic growth. The primary sector of an economy comprises agriculture and other activities which is significant to the Gross Domestic Product (GDP). For decades, agriculture has been associated with the productions of a maximum of crops, food, and other raw materials to the country. At present, agriculture includes farming, fruits, vegetables, dairy, mushrooms, etc. Agriculture not only provides food it also provides employment opportunities to a very large percentage of the population of a country. Crops are living plants grown by farmers. The majority of the crops are food-related, such as fruit, vegetables, and grains. Some crops are grown for fabrics like cotton, pharmaceuticals like quinine, and other products like wood and rubber. Most of the crops are destroyed due to the rain or due to other climatic conditions. Most of the crops are grown and are affected by diseases, and these diseases are spread over the crops. If we overcome these challenges, we can achieve maximum yield and we can overcome the underutilization of crops (1). Plant disease diagnosis through optical observation of the symptoms on plant leaves, incorporates a significantly high degree of complexity. Due to this complexity and to the large number of cultivated plants and their existing phytopathological problems, even experienced agronomists and plant pathologists often fail to successfully diagnose specific diseases, and are consequently led to mistaken conclusions and treatments.

The existence of an automated computational system for the detection and diagnosis of plant diseases, would offer a valuable assistance to the agronomist who is asked to perform such diagnoses through optical observation of leaves of infected plants (Mohanty et al., 2016; Yang and Guo, 2017).

If the system was simple to use and easily accessible through a simple mobile application, it could also be a valuable tool for farmers in parts of the world lacking the appropriate infrastructure for the provision of agronomic and phytopathological advice. In addition, in the case of large-scale cultivations, the system could be combined with autonomous agricultural vehicles, to accurately and timely locate phytopathological problems throughout the cultivation field, using continuous image capturing.

All these are, of course, valid under the condition that the system could achieve high levels of performance in detecting and diagnosing specific diseases in real-life conditions (i.e., in the cultivation field), and that it could be operated through an appropriate, easy-to-use, and user-friendly mobile application (a first step towards that direction has been made by Johannes et al. (2017) for the specific case of wheat plants) (3).

## Motivation:

The main motive of our proposed modes is as follows.

- Identifies which area of the plant/leaf is affected and preventive measurements can be taken care of.
- Predicting the number of defects in individual plants to understand
- This application will be installed in the farmland once the occurrence of the disease is found necessary pesticides can be applied to the affected area.
- The system will be simple to use and easily accessible through a simple mobile application, it could be a valuable tool for farmers in parts of the world lacking the appropriate infrastructure for the provision of agronomic and phytopathological advice
- Identifying the infected plant manually consumes more time and also error in identifying them is more. This also requires more labor, using our model sorts all these issues.

## Objectives:

The main objective of our model is to help farmers in the prevention of crop diseases at an earlier stage by using deep learning techniques such as image classification, object detection, and semantic segmentation of computer vision.

If we speak on classification, Farmers have been able to identify crop diseases with their naked eye since the beginning of time and continue to do so on a daily basis, which forces them to make difficult decisions about which fertilizers to employ. It necessitates a thorough understanding of disease kinds as well as a great deal of expertise to ensure accurate disease identification. Farmers are frequently perplexed by diseases that appear to be almost identical. By using my classification model, they can accurately diagnose crop diseases. And with the object detection model the occurrence of disease in a plant leaf is detected. All the affected area is identified using the bonded box. By doing this, farmers can apply necessary medicines to that area or remove that part of the plant before it affects the entire plan.

So building an end-to-end application where image classification, object detection, and segmentation are used for identifying diseases is my goal. The more diseases we include the more benefit we provide and accuracy is important. Providing farmers with the means to identify the occurrence of diseases in their plants. Then they can spray pesticides on these plants and protect the plant before the whole plant is affected.

## Thesis Outline:

The content of this thesis is arranged as follows: The “Definition of plant disease detection and diagnosis problem” section gives the definition of the plant disease detection and diagnosis problem. Along with this, I state the “Artificial intelligence background” which starts with “machine learning”. The “Machine learning” section speaks on some concepts of machine learning. The “Image recognition technology based on deep learning” section focuses on a detailed introduction of image recognition technology based on deep learning. The “Plant disease detection and diagnosis methods based on deep learning” section analyzes the three kinds of plant disease detection and diagnosis methods based on deep learning according to network structure, including classification, detection, and segmentation networks. At that point, we go to the next section. The “Methodology” section introduces some built models of plant disease detection and diagnosis and compares the performance of existing models. I also speak on challenges I faced and global challenges of plant disease detection and diagnosis based on deep learning. The “Results” section introduces the final results of this thesis. The “Conclusions and future directions” section prospects the possible research focus and development direction in the future.

## Background:

Plant disease detection and diagnosis is a very important research area in the field of machine vision. It is a technology that uses machine vision equipment to acquire images and judge whether there are diseases and pests in the collected plant images. At present, machine vision-based plant disease detection and diagnosis equipment has been initially applied in agriculture and has replaced traditional naked eye identification to some extent.

For traditional machine vision-based plant disease detection and diagnosis methods, conventional image processing algorithms or manual design of features plus classifiers are often used. This kind of method usually makes use of the different properties of plant diseases and pests to design the imaging scheme and chooses an appropriate light source and shooting angle, which is helpful in obtaining images with uniform illumination. Although carefully constructed imaging schemes can greatly reduce the difficulty of classical algorithm design, they also increase the application cost. At the same time, under natural environments, it is often unrealistic to expect classical algorithms designed to completely eliminate the impact of scene changes on recognition results. In real complex natural environments, plant disease and pest detection faces many challenges, such as small differences between the lesion area and the background, low contrast, large variations in the scale of the lesion area and various types, and a lot of noise in the lesion image. Also, there are a lot of disturbances when collecting plant disease and pest images under natural light conditions. At this time, traditional classical methods often appear helpless and it is difficult to achieve better detection results.

In recent years, with the successful application of deep learning models represented by convolutional neural networks (CNNs) in many fields of computer vision (CV), for example traffic detection, medical image recognition, scenario text detection, expression recognition, face recognition, etc., several plant disease detection and diagnosis methods based on deep learning have been applied in real agricultural practice. Some domestic and foreign companies have developed a variety of deep learning-based plant disease detection and diagnosis systems. Therefore, plant disease and pest detection methods based on deep learning not only have important academic research value but also have a very broad market application prospect (4).

## Scope and Limitations:

In this thesis, I aim to detect and diagnose plant diseases through the analysis of leaf images. My goal is to identify the presence of disease at the earliest stage of its development, preventing it from spreading throughout the entire plant or affecting the fruit. At one stage of my research, specifically level three, I incorporate the presence of pests into my model to improve its accuracy. In future stages, such as level four, I plan to expand the scope of my research to include the diagnosis of fruit diseases.

Despite my ambitious goals, my work was limited by several constraints. I had only 50 days to study, build a model, and write the book. Another limitation was the small amount of resources available. I initially started training with my PC's CPU but discovered it was very slow. I then tried using my PC's GPU, but after a week of no progress and a destroyed GPU, I realized that was not a good idea. Finally, I turned to Kaggle, the world's largest data science community with powerful tools and resources to help achieve data science goals. Kaggle provided me with only 30 hours of GPU (P100) usage every 7 days and an input limit of about 20GB for datasets. I also used Colab for software development. The biggest challenge in plant disease detection and diagnosis is the lack of data. We often merge small datasets to create a larger one. My internet efficiency is low; it takes me 7 hours to upload 1.4GB. These are the limitations that I faced while conducting my research.

## Section2: Definition of plant disease and pests detection and diagnosis problem

### Definition of plant diseases and pests:

Plant diseases and pests are types of natural disaster that affect the normal growth of plants and even cause plant death during the whole growth process of plants from seed development to seedling and to seedling growth. In machine vision tasks, plant diseases and pests tend to be the concepts of human experience rather than a purely mathematical definition.

### Definition of plant disease and pests detection and diagnosis:

Compared to the specific classification, detection and segmentation tasks in computer vision, the requirements for plant diseases and pests detection and diagnosis are very general. In fact, its requirements can be divided into three different levels: what, where and how. In the first stage, “what” corresponds to the classification task in computer vision. As shown in Fig. 1, the label of the category to which it belongs is given. The task in this stage can be called classification and only gives the category information of the image. In the second stage, “where” corresponds to the location task in computer vision, and the positioning of this stage is the rigorous sense of detection. This stage not only identifies which types of diseases and pests exist in the image, but also gives their specific locations. As shown in Fig. 1, the affected area of gray mold is marked with a rectangular box. In the third stage, “how” corresponds to the segmentation task in computer vision. As shown in Fig. 1, the lesions of gray mold are distinguished from the background pixel by pixel, and a series of information such as the length, area, and location of the lesions of gray mold can be further obtained, which can assist the higher-level severity level evaluation of plant diseases and pests.

Classification describes the image globally through feature expression, and then determines whether there is a certain kind of object in the image by means of classification operation; while object detection focuses on local description, that is, answering which object exists in which position in an image, so in addition to feature expression, object structure is the most obvious feature that object detection differs from object classification. That is, feature expression is the main research line of object classification, while structure learning is the research focus of object detection. Although the function requirements and objectives of the three stages of plant diseases and pests detection are different, in fact, the three stages are mutually inclusive (interconnected) and can be converted. For example, the “where” in the second stage contains the process of “what” in the first stage, and the “how” in the third stage can finish the task of “where” in the second stage. Also, the “what” in the first stage can achieve the goal of the second and the third stages through some methods. Therefore, the problem in this thesis is collectively referred to as plant diseases and pests detection and diagnosis as conventions in the following text, and the terminology differentiates only when different network structures and functions are adopted (4).

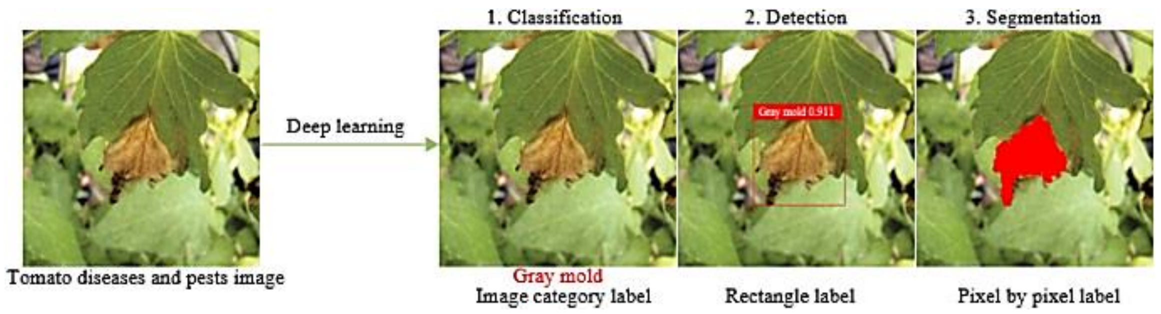


**Figure 1: Definition of plant disease and pests detection and diagnosis problem**

## Comparison with traditional plant diseases and pests detection and diagnosis methods:

To better illustrate the characteristics of plant disease and pest detection and diagnosis methods based on deep learning, “a comparison is made with traditional plant disease and pest detection and diagnosis methods in four aspects: essence, method, required conditions and applicable scenarios. Detailed comparison results are shown in Table 1 (4).

**Table 1: Contrast between traditional image processing methods and deep learning methods**

| Technology | Traditional image processing methods | Traditional image processing methods |
|---|---|---|
| **Essence** | Manual design features + classifiers (or rules) | Automatic learning of features from large amounts of data |
| **Method** | **Image segmentation method**: Threshold segmentation; Roberts, Prewitt, Sobel, Laplace and Kirsh edge detection; region segmentation<br>**Feature extraction method**: SIFT, HOG, LBP, shape, color and texture feature extraction method<br>**Classification method**: SVM, BP, Bayesian | CNN |
| **Required conditions** | Relatively harsh imaging environment requirements, high contrast between lesion and non-lesion areas, less noise | Adequate learning data and high-performance computing units |
| **Applicable scenarios** | It is often necessary to change the threshold or redesign the algorithm when imaging environment or plant diseases and pests class changes, which has poor recognition effect in real complex natural environment | It has ability to cope with certain real and complex natural environment changes |

## Section 3: Overview of Machine Learning with a Focus on CNN and YOLO

In this section, we aim to establish a solid background on the machine learning concepts used in this project. We will start by defining machine learning and discussing the three main types: supervised, unsupervised, and semi-supervised learning. We will also cover artificial neural networks (ANNs) and delve into image recognition technology based on deep learning. This will include an introduction to deep learning theory and convolutional neural networks (CNNs), followed by a brief overview of classification, detection, and segmentation networks.

### Machine Learning:

There are many definitions for machine learning. One definition states that Machine Learning is the science of programming computers so they can learn from data (5). Another more general definition states that Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed (6). A more engineering-oriented definition is: a computer program is said to learn from experience E with respect to some task T and some performance measure P if its performance on T, as measured by P, improves with experience E (7). Another definition is: in other words, machine learning is defined as: The scientific study of algorithms and statistical models that computer systems use to perform a specific task without being explicitly programmed (8).

Machine learning has many applications, for example: analyzing images of products on a production line to automatically classify them, detecting tumors in brain scans, automatically classifying news articles, automatically flagging offensive comments on discussion forums, forecasting a company’s revenue next year, detecting credit card fraud, recommending a product that a client may be interested in based on past purchases, building an intelligent bot for a game, and much more.

## Supervised vs Unsupervised Learning vs Semi-supervised:

Machine Learning systems can be classified according to the amount and type of supervision they receive during training. The three main categories are supervised learning, unsupervised learning, and semi-supervised learning.

***In supervised learning,*** the training set you feed to the algorithm includes the desired solutions, which are called labels. The two most typical supervised learning tasks are classification, where the system has to learn how to classify a particular input into one of a set of categories, and regression, where the task is to predict a target numeric value. Some important supervised learning algorithms include k-nearest neighbors, linear regression, logistic regression; support vector machines (SVMs), decision trees, random forests, and neural networks.

***In unsupervised learning***, the training data is unlabeled. Therefore, the system has to be able to learn patterns in the data without any kind of guidance. Some important unsupervised learning problems include clustering, anomaly detection, visualization, and dimensionality reduction. Some important algorithms include k-means clustering and DBSCAN.

***Semi-supervised learning*** is a branch that combines a small amount of labeled data with a large amount of unlabeled data during training. It falls between unsupervised learning (with no labeled training data) and supervised learning (with only labeled training data). Semi-supervised learning aims to alleviate the issue of having limited amounts of labeled data available for training. Unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy (9). Some important semi-supervised learning algorithms include Label Propagation, Label Spreading, and Self-Training (10).

**Artificial neural networks (ANNs)** are computing systems inspired by the biological neural networks that constitute animal brains (11)

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal to other neurons. An artificial neuron receives signals, processes them, and can signal neurons connected to it. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs called an activation function. The connections are called edges. Neurons and edges typically have a weight that adjusts as learning proceeds. The weight increases or decreases the strength of the signal at a connection. Neurons may have a threshold such that a signal is sent only if the aggregate signal crosses that threshold.

Typically, neurons are aggregated into layers. Different layers may perform different transformations on their inputs. Signals travel from the first layer (the input layer) to the last layer (the output layer), possibly after traversing the layers multiple times.

An ANN is trained by processing examples, each of which contains a known input and result. The network forms probability-weighted associations between the two, which are stored within the data structure of the net itself. The training is conducted by determining the difference between the processed output of the network and a target output. This difference is the error. The network then adjusts its weighted associations according to a learning rule and using this error value. Successive adjustments will cause the neural network to produce output that is increasingly similar to the target output. After a sufficient number of these adjustments, the training can be terminated based on certain criteria. This is a form of supervised learning (12).

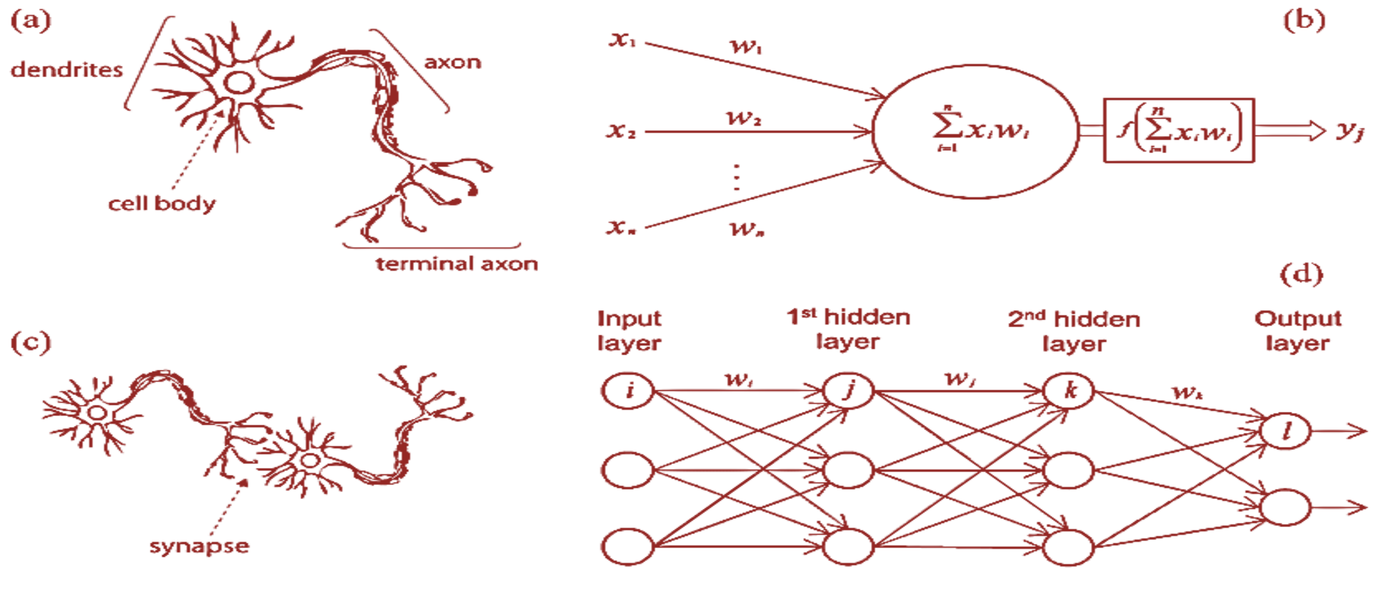


**Figure 2: Artificial neural network inspired by the biological neural networks**

## Image recognition technology based on deep learning:

Compared with other image recognition methods, image recognition technology based on deep learning does not require the extraction of specific features. Instead, it can find appropriate features through iterative learning. This allows it to acquire global and contextual features of images and results in strong robustness and higher recognition accuracy (4).

### Deep learning theory:

The concept of Deep Learning (DL) originated from a paper published in Science by Hinton et al. in 2006 (13). The basic idea of deep learning is to use neural networks for data analysis and feature learning. Data features are extracted by multiple hidden layers, with each hidden layer acting as a set of perceptrons. The perceptron is used to extract low-level features, which are then combined to obtain abstract high-level features. This can significantly alleviate the problem of local minima.

Deep learning overcomes the disadvantage of traditional algorithms relying on “artificially designed features” and has attracted more and more researchers’ attention. It has now been successfully applied in computer vision, pattern recognition, speech recognition, natural language processing, and recommendation systems (14).

Traditional image classification and recognition methods that rely on manually designed features can only extract underlying features and have difficulty extracting deep and complex image feature information (15). Deep learning can solve this bottleneck by directly conducting unsupervised learning from the original image to obtain multi-level image feature information such as low-level features, intermediate features, and high-level semantic features.

Traditional plant disease and pest detection algorithms mainly adopt the image recognition method of manually designed features, which is difficult and depends on experience and luck. It cannot automatically learn and extract features from the original image. On the contrary, deep learning can automatically learn features from large data without manual manipulation. The model is composed of multiple layers, which have good autonomous learning ability and feature expression ability. It can automatically extract image features for image classification and recognition. Therefore, deep learning can play a great role in the field of plant disease and pest image recognition.

At present, deep learning methods have developed many well-known deep neural network models, including deep belief network (DBN), deep Boltzmann machine (DBM), stack de-noising auto-encoder (SDAE), and deep convolutional neural network (CNN) (16). In the area of image recognition, the use of these deep neural network models to realize automated feature extraction from high-dimensional feature space offers significant advantages over traditional manual design feature extraction methods. In addition, as the number of training samples grows and computational power increases, the characterization power of deep neural networks is being further improved.

Nowadays, the boom of deep learning is sweeping both industry and academia, with the performance of deep neural network models significantly ahead of traditional models. In recent years, the most popular deep learning framework is the deep convolutional neural network (4).

## Convolutional neural network:

Convolutional Neural Networks (CNN) has a complex network structure and can perform convolution operations. As shown in Fig. 3, the CNN model is composed of an input layer, convolution layer, pooling layer, fully connected layer, and output layer. In one model, the convolution layer and the pooling layer alternate several times. When the neurons of the convolution layer are connected to the neurons of the pooling layer, no full connection is required.

CNN is a popular model in the field of deep learning due to its huge model capacity and complex information processing capabilities. This enables CNN to excel in image recognition tasks. The successes of CNN in computer vision have also boosted the growing popularity of deep learning.

In the convolution layer, a convolution kernel is first defined. This kernel can be considered as a local receptive field, which is one of the greatest advantages of CNN.

When processing data information, the kernel slides over the feature map to extract part of the feature information. After feature extraction by the convolution layer, the neurons are input into the pooling layer for further feature extraction. Commonly used pooling methods include calculating the mean, maximum, and random values of all values in the local receptive field.

After passing through several convolution and pooling layers, the data enters the fully connected layer where all neurons are fully connected with those in the upper layer. Finally, data in the fully connected layer can be classified using methods such as softmax before being transmitted to the output layer for result output (17) (18) (4).

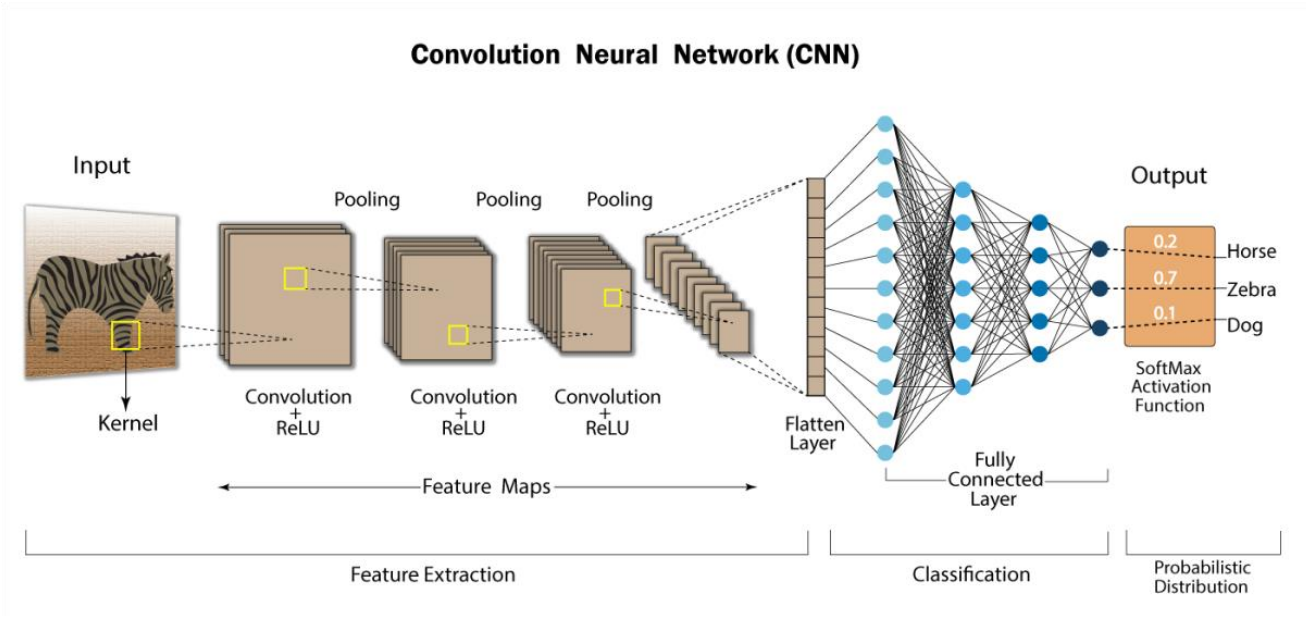


**Figure 3: Convolutional neural network**

## Plant diseases and pests detection and diagnosis methods based On deep learning:

This sub-section provides a summary overview of plant disease and pest detection and diagnosis methods based on deep learning. Since the goal achieved is consistent with computer vision tasks, these methods can be seen as an application of relevant classical networks in the field of agriculture. As shown in Fig. 4, the network can be further subdivided into classification, detection, and segmentation networks according to their different structures. This sub-section is also subdivided into several different sub-methods according to the processing characteristics of each type of method.

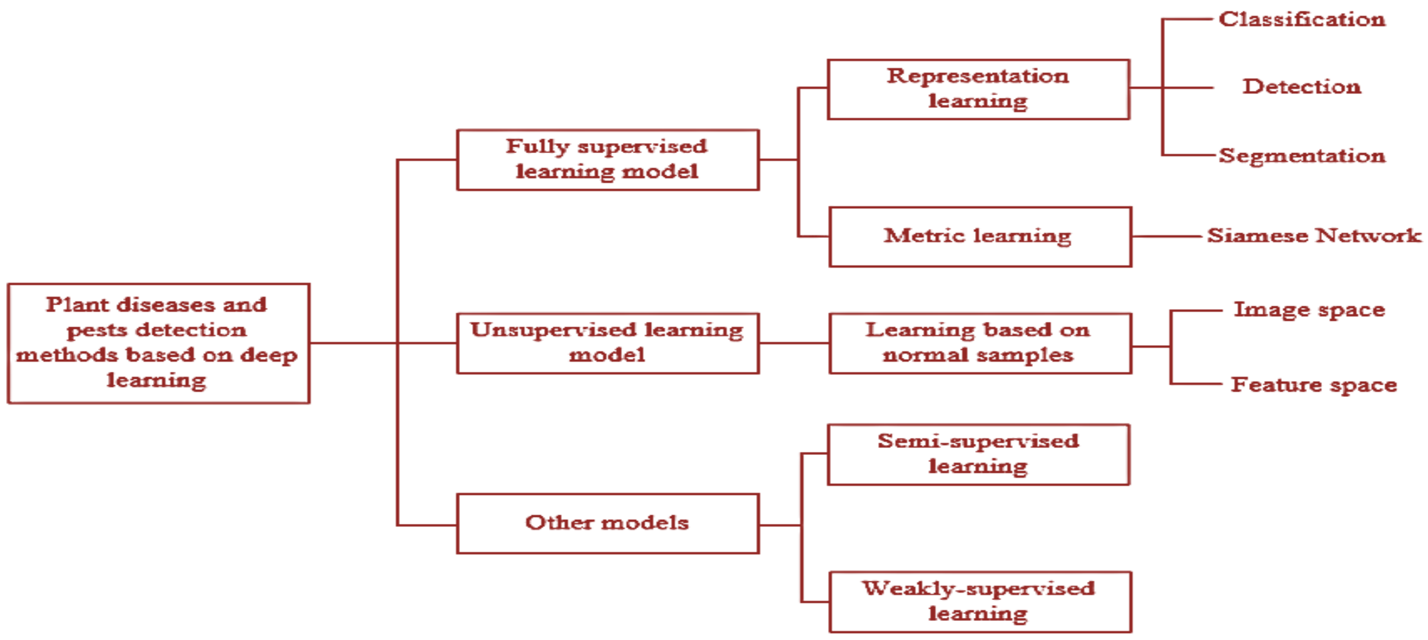

Plant diseases and pests detection methods based on deep learning
Fully supervised learning model
Unsupervised learning model
Other models
Representation learning
Metric learning
Learning based on normal samples
Semi-supervised learning
Weakly-supervised learning
Classification
Detection
Segmentation
Siamese Network
Image space
Feature space

**Figure 4: Framework of plant diseases and pests detection and diagnosis methods based on deep learning**

## Classification network:

In a real natural environment, the great differences in shape, size, texture, color, background, layout, and imaging illumination of plant diseases and pests make recognition a difficult task. Due to the strong feature extraction capability of CNNs, the adoption of CNN-based classification networks has become the most commonly used pattern in plant disease and pest classification. Generally, the feature extraction part of a CNN classification network consists of cascaded convolution layers and pooling layers, followed by a fully connected layer (or average pooling layer) and a softmax structure for classification. Existing plant disease and pest classification networks mostly use mature network structures in computer vision, including AlexNet (19), GoogleLeNet (20), VGGNet (21), ResNet (22), Inception V4 (23), DenseNets (24), MobileNet (25), and SqueezeNet (26).There are also some studies that have designed network structures based on practical problems . By inputting a test image into the classification network, the network analyzes the input image and returns a label that classifies the image. According to the differences in tasks achieved by the classification network method, it can be subdivided into three subcategories: using the network as a feature extractor, using the network for direct classification, and using the network for lesion location.

### *Using network as feature extractor*:

In early studies on plant disease and pest classification methods based on deep learning, many researchers took advantage of the powerful feature extraction capability of CNNs. These methods were combined with traditional classifiers (27). First, images are input into a pre-trained CNN network to obtain image characterization features. The acquired features are then input into a conventional machine learning classifier (e.g., SVM) for classification. This approach was used by Yalcin et al. (28)

### *Using network for classification directly:*

Directly using a classification network to classify lesions is the earliest common means of applying CNN in plant disease and pest detection. According to the characteristics of existing research work, it can be further subdivided into original image classification, classification after locating the Region of Interest (ROI), and multi-category classification.

1. **Original image classification**: This involves directly inputting the collected complete plant disease and pest image into the network for learning and training. Thenmozhi et al. (29) proposed an effective deep CNN model and used transfer learning to fine-tune the pre-training model.

2. **Classification after locating the Region of Interest ROI**: For the whole image acquired, we should focus on whether there is a lesion in a fixed area. We often obtain the Region of Interest (ROI) in advance and then input the ROI into the network to judge the category of diseases and pests. This approach was used by Nagasubramanian et al. (30).

3. **Multi-category classification**: When the number of plant disease and pest classes to be classified exceeds 2 classes, the conventional plant disease and pest classification network is the same as the original image classification method. That is, the output nodes of the network are the number of plant disease and pest classes + 1 (including normal class). However, multi-category classification methods often

use a basic network to classify lesions and normal samples, and then share feature extraction parts on the same network to modify or increase the classification branches of lesion categories. This approach is equivalent to preparing a pre-training weight parameter for subsequent multi-objective plant disease and pest classification networks, which is obtained by binary training between normal samples and plant disease and pest samples. Picon et al. (31) used this approach.

### *Using a network for lesion location:*

Generally, a classification network can only complete the classification of image labels. However, it can also achieve the location of lesions and pixel-by-pixel classification by combining different techniques and methods. According to the different means used, it can be further divided into three forms: sliding window, heatmap, and multi-task learning network.

1. **Sliding window:** This is the simplest and most intuitive method to achieve coarse lesion location. The image in the sliding window is input into the classification network for plant disease and pest detection by redundantly sliding on the original image through a smaller size window. Finally, all sliding windows are connected to obtain the results of the lesion location.

2. **Heatmap**: This is an image that reflects the importance of each region in the image; the darker the color represents the more important it is. In the field of plant disease and pest detection and diagnosis, the darker the color in the heatmap represents the greater probability that it is a lesion.

3. **Multi-task learning network**: If a pure classification network does not add any other skills, it could only realize image-level classification. Therefore, to accurately locate plant diseases and pests, the designed network should often add an extra branch, with both branches sharing the results of feature extraction. In this way, the network generally has both classification and segmentation output for plant diseases and pests, forming a multi-task learning network that takes into account

the characteristics of both networks. For segmentation network branches, each pixel in the image can be used as a training sample to train the network. Therefore, a multi-task learning network not only uses segmentation branches to output specific segmentation results of lesions but also greatly reduces sample requirements for classification networks.

In summary, methods based on classification networks are widely used in practice and many scholars have carried out application research on plant disease and pest classification. At the same time, different sub-methods have their own advantages and disadvantages as shown in Table 2 (4).

**Table 2: Comparison of advantages and disadvantages of each sub-method of classification network**

| Method | Advantages | Disadvantages |
|---|---|---|
| **Using network as feature extractor** | Obtaining effective lesion features | Relying on other classifiers for final classification results |
| **Original image classification** | Classic in structure, it is also the basis of other classification network sub-methods and can refer to many existing networks | Lesions need to account for a certain proportion in the image, otherwise their characteristics are easily pooled out, and generally only one class of lesion is allowed in an image |
| **Classification after locating ROI** | Obtaining ROI information of the lesions | Additional methods are needed to obtain ROI |
| **Multi-category classification** | Solving sample imbalance to some extent | Secondary training is needed |
| **Sliding window** | Get rough localization of lesions in images | Sliding window size requires accurate selection, and can only get rough position, slow speed of traversal and sliding |
| **Heatmap** | Generate more accurate lesion areas | Accurate lesions location depends on network classification performance |
| **Multi-task learning network** | Combining other networks to obtain exact location and category of lesions simultaneously, and reducing the number of training samples required | The network structure is relatively complex, and a pixel-by-pixel label is required when adding segmentation branches |

## Detection network:

Object positioning is one of the most basic tasks in the field of computer vision and is also closely related to plant disease and pest detection. Its purpose is to obtain accurate location and category information for objects. Currently, there are many deep learning-based object detection methods. Generally speaking, plant disease and pest detection networks based on deep learning can be divided into two types: two-stage networks represented by Faster R-CNN and one-stage networks represented by SSD and YOLO. The main difference between the two types of networks is that the two-stage network first generates a candidate box (proposal) that may contain lesions and then further executes the object detection process. In contrast, the one-stage network directly uses the features extracted in the network to predict the location and class of lesions.

### *Plant disease and pest detection and diagnosis based on two-stage networks:*

The basic process of a two-stage detection network (Faster R-CNN) is to first obtain the feature map of the input image through the backbone network, then calculate the anchor box confidence using RPN and get the proposal. Then, input the feature map of the proposal area after ROIpooling into the network, fine-tune the initial detection results, and finally get the location and classification results of lesions. Therefore, according to the characteristics of plant disease and pest detection, common methods often improve on the backbone structure or its feature map, anchor ratio, ROIpooling, and loss function.

### *Plant disease and pest detection and diagnosis based on one-stage networks:*

The one-stage object detection algorithm has eliminated the region proposal stage but directly adds a detection head to the backbone network for classification and regression, thus greatly improving the inference speed of the detection network. The single-stage detection network is divided into two types: SSD and YOLO. Both use the whole image as input to the network and directly return the position of bounding boxes and their categories at the output layer.

Compared with traditional convolutional neural networks, SSD selects VGG16 as its trunk network and adds a feature pyramid network to obtain features from different layers and make predictions.

“You Only Look Once” (YOLO) considers the detection task as a regression problem and uses global information to directly predict the bounding box and category of an object to achieve end-to-end detection with a single CNN network. YOLO can achieve global optimization while satisfying higher accuracy with greatly improved detection speed (4). YOLO has many versions starting from 1 (32) up to 8 which we used. For more information go to (33) (34) . To give an overview: YOLO takes an entire image in a single shot and predicts bounding box coordinates and class probabilities for these boxes. The biggest advantage of YOLO is its lightning-fast execution speed. YOLO also understands generalized object representation. The network does not look at the entire image but only at parts of images that have a higher chance (probabilities) of containing an object.

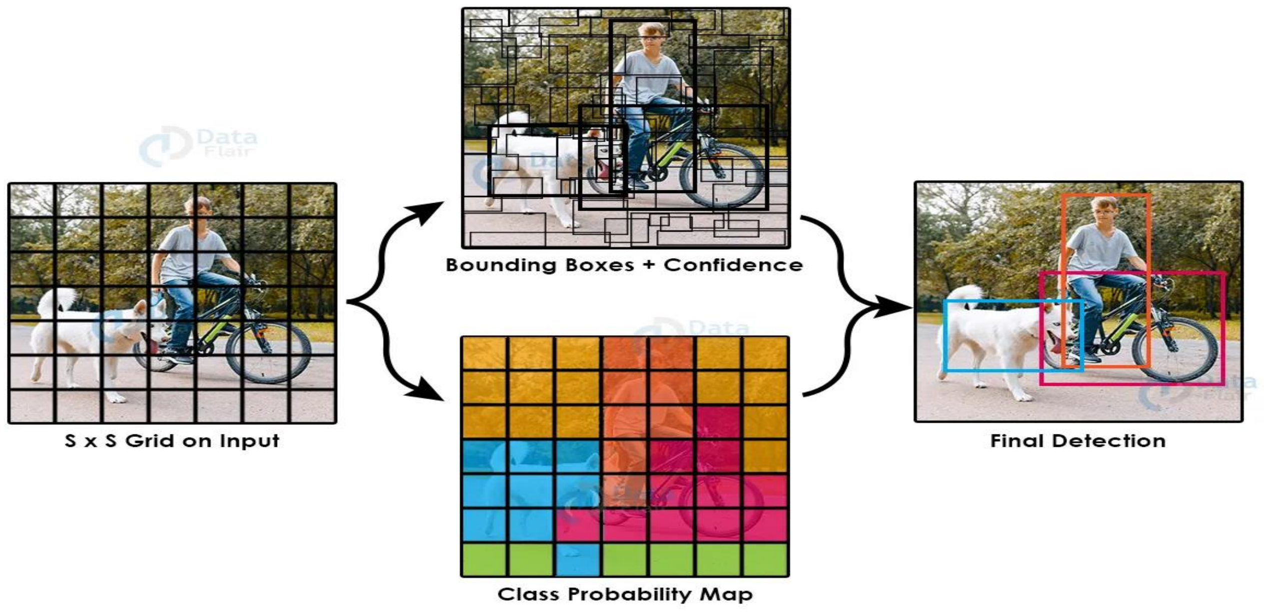


**Figure 5: Yolo Object detection**

YOLO first takes input images then divides them into grids (see figure 5). For each box, it checks whether an object is present or not. It calculates X-Coordinate of bounding box center inside cell (0;1 with respect to grid cell size), Y-Coordinate of bounding box center inside cell (0;1 with respect to grid cell size), W-Bounding box width [0;1] with respect to image, H-Bounding box height [0;1] with respect to image, C-Bounding box confidence (object in box) (see figure 6). PC represents whether an object is present in grid or not (Probability), X,Y,H,W represents bounding box of an object, C1,C2,C3 represent class of objects such as car, bike, bus etc. If PC = 0 then C1, C2, C3 = 0. If PC = 1 it tries to find the center of the available object. If it finds multiple bounding boxes it uses Intersection Over Union (IOU) and Non-max suppression algorithms to find the best bounding box. Once we get the best bounding box for which object exists it uses CNN to classify the images (1).

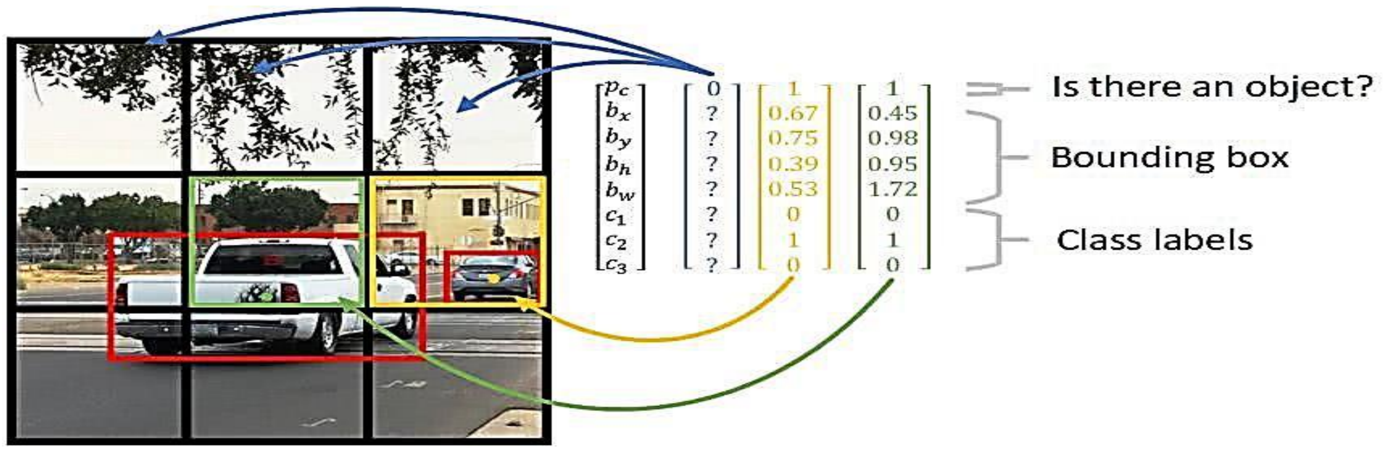


**Figure 6: How does the algorithm predict bounding boxes**

In summary, in the field of plant disease and pest detection that emphasizes detection accuracy at this stage, more models based on two-stage networks are used. In contrast, in the field of plant disease and pest detection that pursues detection speed, more models based on one-stage networks are used.

## ***Can a detection network replace a classification network?*** (*Important question for our work*)

The task of a detection network is to solve the location problem of plant diseases and pests. The task of a classification network is to judge the class of plant diseases and pests. Visually, the hidden information of a detection network includes category information; that is, the category information of plant diseases and pests that need to be located needs to be known beforehand, and corresponding annotation information should be given in advance to judge the location of plant diseases and pests. From this point of view, the detection network seems to include the steps of the classification network; that is, the detection network can answer “what kind of plant diseases and pests are in what place.” However, there is a misconception in which “what kind of plant diseases and pests” is given a priori; that is, what is labeled during training is not necessarily the real result. In the case of strong model differentiation; that is, when the detection network can give accurate results, it can answer “what kind of plant diseases and pests are in what place” to a certain extent. However, in the real world, in many cases it cannot uniquely reflect the uniqueness of plant disease and pest categories but can only answer “what kind of plant diseases and pests may be in what place.” Then the involvement of a classification network is necessary. Thus, a detection network cannot replace a classification network.

## Segmentation network:

A segmentation network converts the plant disease and pest detection and diagnosis task to semantic and even instance segmentation of lesions and normal areas. It not only finely divides the lesion area but also obtains the location, category, and corresponding geometric properties (including length, width, area, outline, center, etc.). It can be roughly divided into Fully Convolutional Networks (FCN) (35) and Mask R-CNN (36).

### *FCN:*

A Full Convolutional Neural Network (FCN) is the basis of image semantic segmentation. Currently, almost all semantic segmentation models are based on FCN. FCN first extracts and encodes the features of the input image using convolution, then gradually restores the feature image to the size of the input image by deconvolution or up-sampling. Based on differences in FCN network structure, plant disease and pest segmentation methods can be divided into conventional FCN, U-net, and SegNet.

1. **Conventional FCN.**
2. **U-net:** U-net is not only a classical FCN structure but also a typical encoder-decoder structure. It is characterized by introducing a layer-hopping connection that fuses the feature map in the encoding stage with that in the decoding stage, which is beneficial for recovering segmentation details.
3. **SegNet**: It is also a classical encoder-decoder structure. Its feature is that the up-sampling operation in the decoder takes advantage of the index of the largest pooling operation in the encoder.

### *Mask R-CNN:*

**Mask R-CNN** is one of the most commonly used image instance segmentation methods at present. It can be considered as a multi-task learning method based on detection and segmentation networks. When multiple lesions of the same type have adhesion or overlap, instance segmentation can separate individual lesions and further count their number. However, semantic segmentation often treats multiple lesions of the same type as a whole (4).

**YOLOv8** also introduces segmentation service and classification.

Compared with classification and detection network methods, the segmentation method has advantages in obtaining lesion information. However, like detection networks, it requires a lot of annotation data, and its annotation information is pixel by pixel, which often takes a lot of effort and cost.

## Evaluation indices used in object detection evaluation of our work:

Common evaluation indices include Precision, Recall, mean Average Precision (mAP) and the harmonic Mean F1 score based on Precision and Recall.

Precision and Recall is defined as:

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP}+\boldsymbol{FP}}\,.\,\mathbf{100\%} \qquad (1)$$

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP}+\boldsymbol{FN}}\,.\,\mathbf{100\%} \qquad (2)$$

In Formula (1) and Formula (2), TP (True Positive) is true-positive, predicted to be 1 and actually 1, indicating the number of lesions correctly identified by the algorithm. FP (False Positive) is false-positive, predicted to be 1 and actually 0, indicating the number of lesions incorrectly identified by the algorithm. FN (False Negative) is false-negative, predicted to be 0 and actually 1, indicating the number of unrecognized lesions.

Detection accuracy is usually assessed using mAP. The average accuracy of each category in the dataset needs to be calculated first:

$$\boldsymbol{P_{Average}} = \sum_{j=1}^{N(class)} \boldsymbol{Precision(j).Recall(j).100\%} \qquad (3)$$

In the above-mentioned formula, N(class) represents the number of all categories, Precision(j) and Recall(j) represents the precision and recall of class *j* respectively. Average accuracy for each category is defined as mAP:

$$\boldsymbol{mAP = \frac{P_{average}}{N(class)}} \qquad (4)$$

The greater the value of mAP, the higher the recognition accuracy of the algorithm; conversely, the lower the accuracy of the algorithm. F1 score is also introduced to measure the accuracy of the model. F1 score takes into account both the accuracy and recall of the model. The formula is

$$\boldsymbol{F1 = \frac{2Precision \,.\, Recall}{Precision + Recall} \,.\, 100\%} \qquad (5)$$

Frames per second (FPS) is used to evaluate the recognition speed. The more frames per second, the faster the algorithm recognition speeds; conversely, the slower the algorithm recognition speed (4).

## Section 4: Methodology

To simplify my work and organize my efforts, I divided my work into three levels. At level one, I worked with the PlantVillage open-source dataset which contains 38 classes of [plant, disease] combinations, including a healthy class. My work focused only on image classification methods. I tried four models of increasing complexity, starting with a scratch model, then Vgg16, Vgg19, and Resnet50V2 which gave the most accuracy. At level two, I went deeper with classes, which included 88 classes of [plant, disease] combinations and also healthy classes collected from a large open-source dataset and more complex model which is ResNet152V2 which is 152 layers and has nearly 53 million parameters. I also started version one of object detection, building two models, xlarge and nano, using the Yolov8 algorithm on the PlanDocPlus dataset built by me, also object detection on an apple dataset collected from PlantApple_2020, Plant-pathology-2021-fgvc8_4 competition which contains 8,837 images of four classes, At level three, At the time of writing this book, I am still working on it.

### Level 1:

At this level, I am working with a classification network. I am using the network for classification directly and the sub-category I use is 'original image classification'. I am following the approach outlined in the paper titled 'Deep learning models for plant diseases detection and diagnosis' (3). My dataset consists of 38 classes and is an open-source dataset on Kaggle that was collected in the lab. I trained a total of 5 models and achieved the highest accuracy of 99.812% using the ResNet50v2 model.

#### The papers:

- Deep learning models for disease detection and diagnosis. (3)
- Using Deep learning for Image-Based plant Disease Detection. (2)
- Plant Disease Detection Using Deep Learning. (37)
- Plant disease identification from individual lesions and spots using deep learning (38)

## Objectives:

After reading the research paper, my goal is to train a classification network model that can accurately distinguish between healthy and diseased plant leaf images. I aim to achieve a validation accuracy of at least 99% for a dataset consisting of 38 classes.

## Data Collection:

For my data collection, I utilized a version of the PlantVillage dataset (39) (40), an open access repository of plant health images introduced by Hughes et al (41)to facilitate the development of mobile disease diagnostics. This dataset contains 54,303 images of healthy and unhealthy plant leaves, categorized into 38 classes based on species and disease. I primarily used the New Plant Diseases Dataset (42), which was derived from the original dataset (PlantVillage) using offline augmentation techniques. This dataset comprises approximately 87K RGB images of healthy and diseased crop leaves, organized into 38 distinct classes. The dataset is split into an 80/20 ratio for training and validation purposes. Additionally, a set of 33 test images was later generated for use in prediction tasks.

**Table 3: New Plant Diseases Dataset**

| class | Plant common name | Disease common name | Image |
|---|---|---|---|
| C0 | Apple | Apple scab | 2520 |
| C1 | Apple | Black rot | 2484 |
| C2 | Apple | Cedar_apple_rust | 2200 |
| C3 | Apple | healthy | 2510 |
| C4 | Blueberry | healthy | 2270 |
| C5 | Cherry(&sour | healthy | 2282 |
| C6 | Cherry(&sour | Powdery_mildew | 2104 |
| C7 | Corn(maize) | Cercospora\|\|Gray_leaf_spot | 2052 |
| C8 | Corn(maize) | Common_rust_ | 2384 |
| C9 | Corn(maize) | healthy | 2324 |
| C10 | Corn(maize) | Northern_Leaf_Blight | 2385 |
| C11 | Grape | black rot | 2360 |
| C12 | Grape | Esca_(Black_Measles) | 2400 |
| C13 | Grape | healthy | 2115 |
| C14 | Grape | Leaf_blight_ (Isariopsis_Leaf_Spot) | 2152 |
| C15 | Orange | Haunglongbing_ (Citrus_greening) | 2513 |
| C16 | Peach | Bacterial_spot | 2297 |
| C_17 | Peach | healthy | 2160 |
| C_18 | Pepper,bell | Bacterial_spot | 2391 |
| C_19 | Pepper,bell | healthy | 2485 |
| C_20 | Potato | Early_blight | 2424 |
| C_21 | Potato | healthy | 2280 |
| C_22 | Potato | Late_blight | 2424 |
| C_23 | Raspberry | healthy | 2226 |
| C_24 | Soybean | healthy | 2527 |
| C_25 | Squash | Powdery_mildew | 2170 |
| C_26 | Strawberry | healthy | 2280 |
| C_27 | Strawberry | Leaf_scorch | 2218 |
| C_28 | Tomato | Bacterial_spot | 2127 |
| C_29 | Tomato | Early_blight | 2400 |
| C_30 | Tomato | healthy | 2407 |
| C_31 | Tomato | Late_blight | 2314 |
| C_32 | Tomato | Leaf_Mold | 2352 |
| C_33 | Tomato | Septoria_leaf_spot | 2181 |
| C_34 | Tomato | Spider_mites Two-spotted_spider_mite | 2176 |
| C_35 | Tomato | Target_Spot | 2284 |
| C_36 | Tomato | mosaic_virus | 2238 |
| C_37 | Tomato | Yellow_Leaf_Curl_Virus | 2451 |

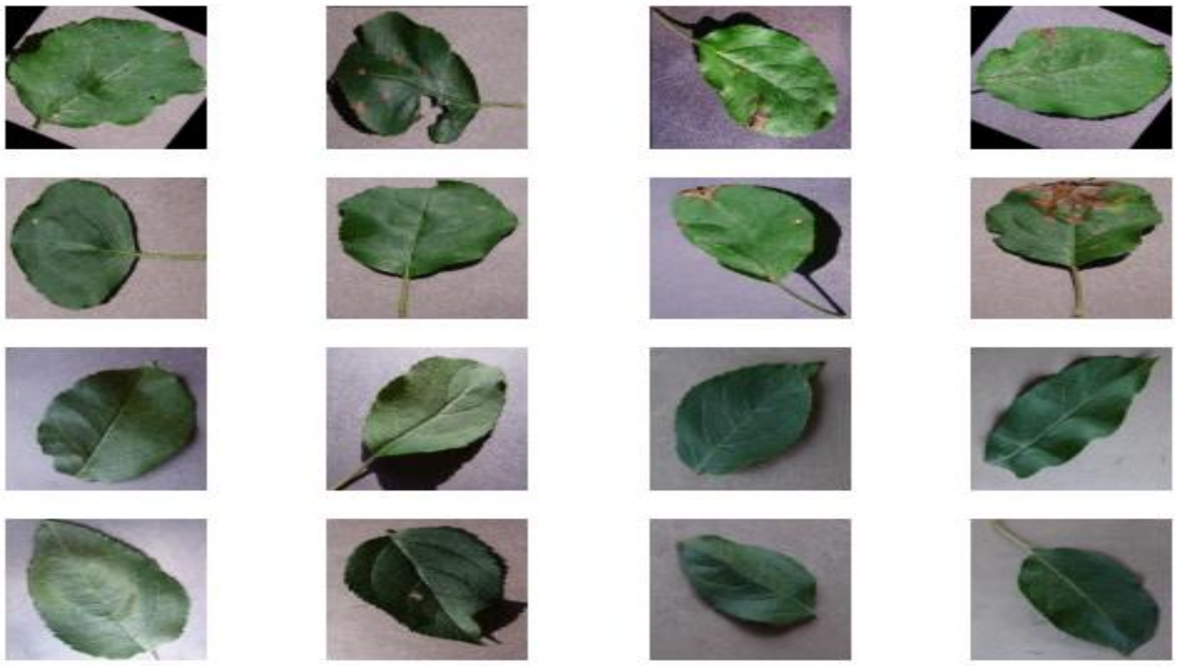


**Figure 7: New Plant Diseases Dataset**

## Training:

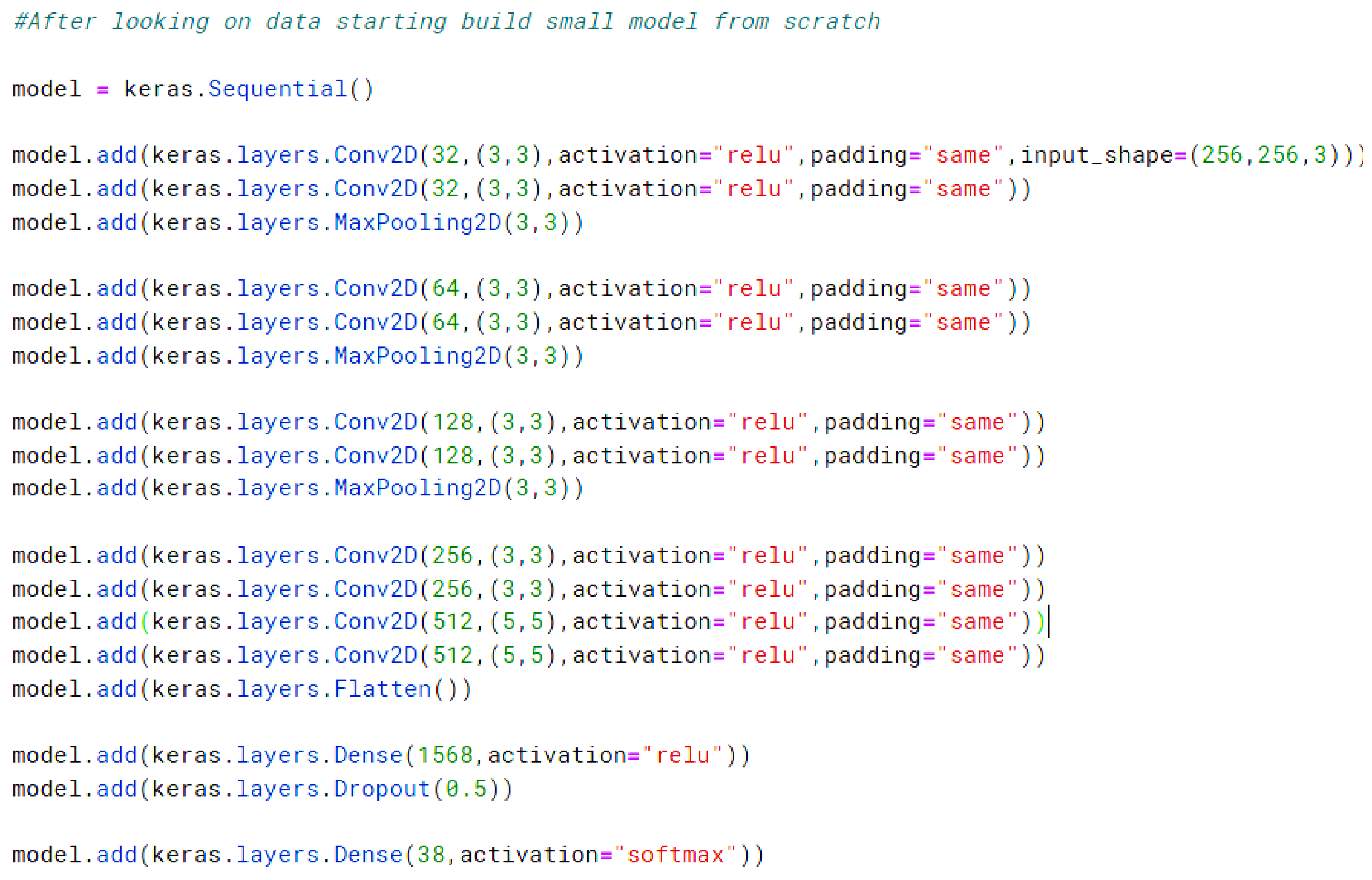
To develop my models, I utilized the TensorFlow (43) and Keras (44) platforms for coding and implementation. I also leveraged pre-trained models, which I downloaded for transfer learning purposes. All of my work was conducted on the Kaggle platform (45), which provided a convenient and powerful environment for experimentation and development.

Unfortunately, I didn't anticipate that I would be writing this book and accidentally overwrote some of the models. However, I still have two models remaining: VGG19 and ResNet50V2. I will provide their details and mention any information I have from previous versions.

In all models, I used the Adam optimizer (46), which stands for Adaptive Moment Estimation. It is an optimization technique for gradient descent that is highly efficient when working with large problems involving a lot of data or parameters. It requires less memory and is computationally efficient. Intuitively, it can be seen as a combination of the 'gradient descent with momentum' algorithm and the 'RMSProp' algorithm. For the loss function, I used Categorical Crossentropy, which is commonly used in classification problems where the classes are mutually exclusive. It measures the dissimilarity between the true class labels and predicted probabilities.

## *Scratch model:*

I started working by building a CNN model from scratch, as shown in Figure 7. I began training the model using the previously mentioned training conditions, with a batch size of 32 and an image scheme of ‘RGB’. I trained the model for 50 epochs and achieved an accuracy of approximately 92.5%, which did not meet my objective. So, I decided to move forward.



```
#After looking on data starting build small model from scratch

model = keras.Sequential()

model.add(keras.layers.Conv2D(32,(3,3),activation="relu",padding="same",input_shape=(256,256,3)))
model.add(keras.layers.Conv2D(32,(3,3),activation="relu",padding="same"))
model.add(keras.layers.MaxPooling2D(3,3))

model.add(keras.layers.Conv2D(64,(3,3),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(64,(3,3),activation="relu",padding="same"))
model.add(keras.layers.MaxPooling2D(3,3))

model.add(keras.layers.Conv2D(128,(3,3),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(128,(3,3),activation="relu",padding="same"))
model.add(keras.layers.MaxPooling2D(3,3))

model.add(keras.layers.Conv2D(256,(3,3),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(256,(3,3),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(512,(5,5),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(512,(5,5),activation="relu",padding="same"))
model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(1568,activation="relu"))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(38,activation="softmax"))
```

**Figure 8: Scratch Model**

*Note: It’s important to note that Kaggle provides a session time of 9 hours. As a result, the number of epochs I was able to run was limited.*

## *Fine-tuning VGG16:*

At this stage, I fine-tuned the VGG16 model by downloading a pre-trained version that had been trained on the ImageNet dataset. I fixed the weights of all layers except the last one and began training the last layer on my 38-class dataset using the same conditions as the scratch model. This resulted in an accuracy of approximately 97%, which did not meet my objective. As a result, I decided to move forward and explore other options.

### *Full fine-tuning VGG16:*

At this stage, I performed full fine-tuning on the VGG16 model by downloading a pre-trained version that had been trained on the ImageNet dataset. I began training all layers of the model on my 38-class dataset using the same conditions as the scratch model. This resulted in an accuracy of approximately 98.5%, which did not meet my objective. As a result, I decided to move forward and explore other options

### *Full fine-tuning VGG19:*

At this stage, I performed full fine-tuning on the VGG19 model by downloading a pre-trained version that had been trained on the ImageNet dataset. I began training all layers of the model on my 38-class dataset using the same conditions as the scratch model with 55 epochs. I also used another feature of TensorFlow which includes `ModelCheckpoint`, `EarlyStopping`, and `CSVLogger`. The most important one is `ModelCheckpoint`. It is a callback in TensorFlow that can be used with Keras to save a model or its weights at some frequency. It can be used to serialize your network to disk each time there is an improvement during training. An "improvement" can be defined as either a decrease in loss or an increase in accuracy. My model saved every time validation accuracy improved. This resulted in an accuracy of 99.06% at epoch 45, which is very close to my objective but I decided to move forward and explore other options. The model was saved at "checkpoint7". The last epoch's model is at "potatoes7.h5". You can also find the notebook, logs, and test results.

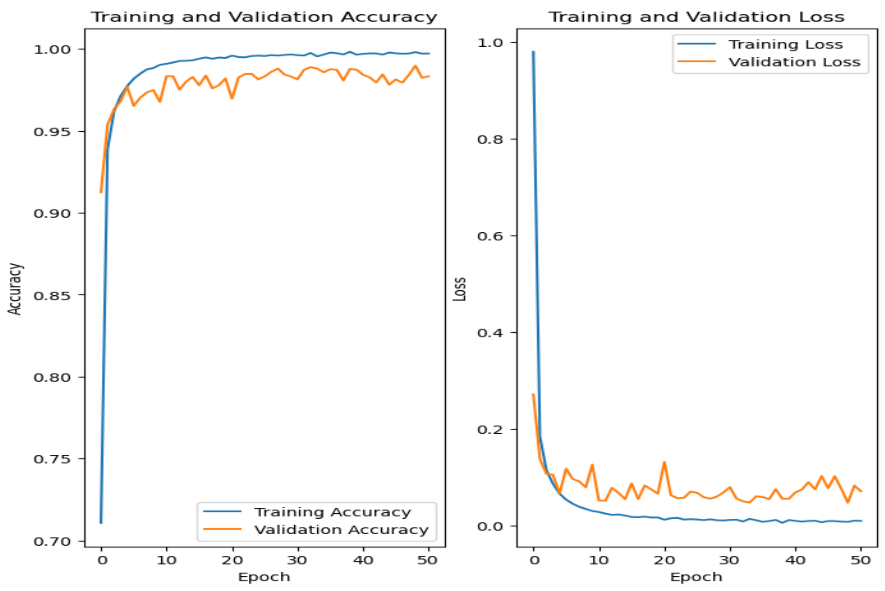


**Figure 9:Vgg19**

### *Full fine-tuningResNet50V2:*

At this stage, I performed full fine-tuning on the ResNetV2 model by downloading a pre-trained version that had been trained on the ImageNet dataset. I began training all layers of the model on my 38-class dataset using the same conditions as the Vgg19 model with 55 epochs and another version with 95 epochs. However, there was no improvement in accuracy between the two versions. As a result, I achieved an accuracy of 99.812% at epoch 46, which surpassed my objective. So I stopped and started thinking about the obstacles of all models at this level. The model was saved at "checkpoint1". The last epoch's model is at "potatoes7.h5". You can also find the notebook, logs, and test results

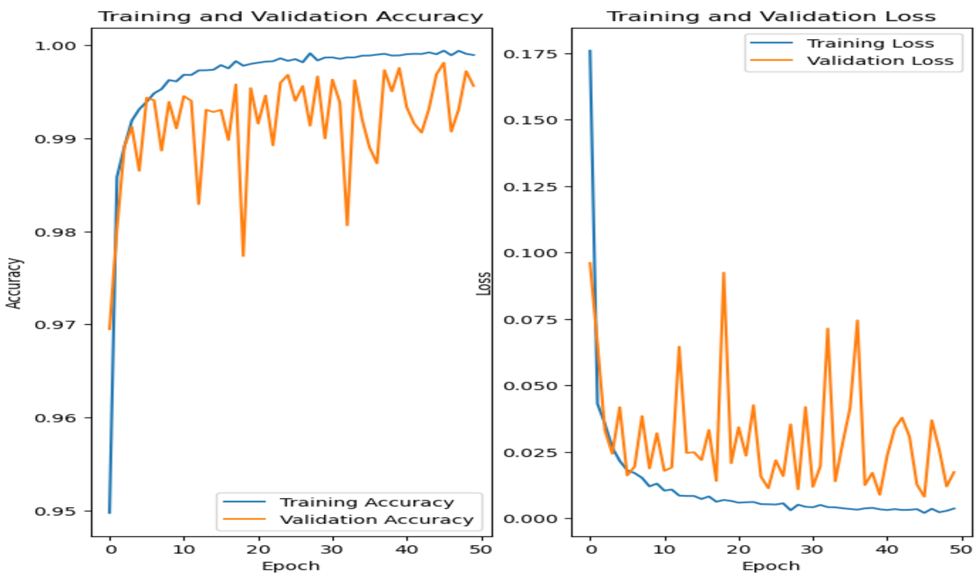


**Figure 10: ResNet50V2**

## Upgrading to the next level:

One limitation of our current approach is that our photographic material consists solely of images taken in experimental laboratory setups, rather than in real-world cultivation fields. This lack of real-world data can reduce the accuracy of our model when applied to actual field conditions. Additionally, all of our images come from a single source or dataset, which limits the variety of images available for training our model. This can further reduce accuracy when our model is used by a diverse group of users with varying image quality and conditions. Furthermore, the number of classes in our current model is relatively small compared to the wide variety of diseases and pests that can affect crops. To address these issues, we are considering upgrading our model to improve its accuracy and effectiveness.

## Level 2:

At this stage, I am working on improving the classification network to address the limitations of the previous version at level one. I am utilizing a dataset called "Plant Disease Classification Merged Dataset" (47), which was compiled from 14 different datasets and includes both laboratory and field images. The dataset comprises 88 classes. To date, I have developed four models: ResNet50V2, ResNet152V2, InceptionResNetV2, and EfficientNetB5. The InceptionResNetV2 model achieved an accuracy of nearly 96.5%. Additionally, I have begun my initial work on an object detection network, but I am encountering difficulty in finding a suitable dataset. Nevertheless, with the assistance of two distinct datasets - PlantDocPlus (an expanded version of PlantDoc) and "Apple Leaf Disease Object Detection dataset" (48) (a private dataset from paper (49)) - I was able to construct two models. Each model has two versions: one for real-time operation and another with more parameters for backend processing. These models were developed using Yolov8, which was the most recent version available at the time.

## The papers:

- Crop Infection Detection using YOLO. (1)
- MGA-YOLO: A lightweight one-stage network for apple leaf disease detection (49)
- Real-Time Flying Object Detection with YOLOv8. (33)

## Objectives:

Since I was unable to locate the dataset used in the study (Plant Village with annotation) (1) or any other suitable dataset for building a robust model, my goal is to develop an object detection network that can effectively interact with the data. The network should be capable of identifying the presence of a disease at a specific location and suggesting its potential type. To ensure accuracy, a classification network can be employed. Using the apple dataset, I aim to construct a reliable predictor and indicator model.

## Data Collection:

### *Classification Networks:*

This dataset was created as part of a study project on plant disease classification. The requirements for the dataset included having a large number of images, with at least one healthy and one diseased image for each plant, covering the most common diseases, annotated images, laboratory and field images, and featuring important staple food and highest global production plant species. To meet these requirements, a new dataset was compiled by combining images from 14 different existing datasets. The properties and imported images of all included datasets are listed in the table below along with their proper references. This new dataset contains both laboratory and field images. Images of non-food plants, singular condition classes, watermarked images, and classes with less than 50 examples were removed. The final dataset used for training within the project contains 88 classes with over 76,000 images and an overall size of 17.6GB. However, it is not possible to completely avoid bias among the classes since different shooting conditions were used for some images (e.g., classes with mainly laboratory images or different soil in the background).

Table 4: Plant Disease Classification Merged Dataset

| Dataset | properties | Merged images |
|---|---|---|
| Plant disease 65 (50) | 65 classes, 62,600 files , contains plantVillage images and additional plants, photos of single plant leaves in front of plain backgrounds | all images beside images of nonfood plants and solitary classes |
| PlantDoc (51) | 2598 images in 27 classes , mostly field images , taken from various angles , lightning condition and with various backgrounds , some laboratory images | all beside solitary classes |
| Coffee plant disease (52) | 1000 images, 3 classes, field images of coffee plants | all |
| Wheat leaf dataset (53) | 407 images in 3 classes , real field wheat images from the Holeta wheat farm in Ethiopia , sorted with the assistance of plant pathologist | all |
| Chili plant disease (54) | 500 images in 5 classes of chili plants , field images of whole plant or plant parts | all |
| Images of soybean leaves (55) | 6410 images in 3 classes , field images of soybean plants captured with smartphones and drones in different heights and different times of day | all |
| Rice leaf diseases (56) | 120 pictures in 3 classes , single rice leaf in front of plain white background | all |
| Rice leafs (57) | 3355 files in 4 classes , single rice leaf in front of plain back ground | only Hispa disease |
| Cucumber plant diseases dataset (58) | 691 images in 2 classes ill and healthy , cucumber field images of plant leaves | all |
| Plant disease expert (59) | single plant leaves in front of plain backgrounds | added 1829 images of 6 classes of tea leaves and 11328 images of grape black rot |
| Leaf disease (combination) (60) | field images | added 2596 images in 5 classes for cassava |
| PDDB (61) | image database of plant dis ease symptoms with images from a multitude of plants ,labelled by phytopathologists | added 1 rice leaf blast class , 5 classes of coffee , 5 classes of sugarcane , 4 of cassava , 5 of corn , 7 of soy and 3 of wheat |
| Sugarcane disease (62) | 299 field images in 3 classes of sugarcane leaves | all |
| Sugarcane leaf disease (63) | 224 images 3 classes , field images from sugarcane farms | all |

**Table 5: Plant Disease Classification Merged Dataset**

| class | Class Name |
|---|---|
| C0 | Apple__black_rot |
| C1 | Apple__healthy' |
| C2 | Apple__rust |
| C3 | Apple__scab |
| C4 | Cassava__bacterial_blight |
| C5 | Cassava__brown_streak_disease |
| C6 | Cassava__green_mottle |
| C7 | Cassava__healthy |
| C8 | Cassava__mosaic_disease |
| C9 | Cherry__healthy |
| C10 | Cherry__powdery_mildew |
| C11 | Chili__healthy |
| C12 | Chili__leaf curl |
| C13 | Chili__leaf spot |
| C14 | Chili__whitefly |
| C15 | Chili__yellowish |
| C16 | Coffee__cercospora_leaf_spot |
| C17 | Coffee__healthy |
| C18 | Coffee__red_spider_mite |
| C19 | Coffee__rust |
| C20 | Corn__common_rust |
| C21 | Corn__gray_leaf_spot |
| C22 | Corn__healthy |
| C23 | Corn__northern_leaf_blight |
| C24 | Cucumber__diseased |
| C25 | Cucumber__healthy |
| C26 | Gauva__diseased |
| C27 | Gauva__healthy |
| C28 | Grape_black_measles |
| C29 | Grape__black_rot |
| C30 | Grape__healthy |
| C31 | Grape_leaf_blight_(isariopsis_leaf_spot) |
| C32 | Jamun__diseased |
| C33 | Jamun__healthy |
| C34 | Lemon__diseased |
| C35 | Lemon__healthy |
| C36 | Mango__diseased |
| C37 | Mango__healthy |
| C38 | Peach__bacterial_spot |
| C39 | Peach__healthy |
| C40 | Pepper_bell__bacterial_spot |
| C41 | Pepper_bell__healthy |
| C42 | Pomegranate__diseased |
| C43 | Pomegranate__healthy |
| C44 | Potato__early_blight |
| C45 | Potato__healthy |
| C46 | Potato__late_blight |
| C47 | Rice__brown_spot |
| C48 | Rice__healthy |
| C49 | Rice__hispa |
| C50 | Rice__leaf_blast |
| C51 | Rice__neck_blast |
| C52 | Soybean__bacterial_blight |
| C53 | Soybean__caterpillar |
| C54 | Soybean__diabrotica_speciosa |
| C55 | Soybean__downy_mildew |
| C56 | Soybean__healthy |
| C57 | Soybean__mosaic_virus |
| C58 | Soybean__powdery_mildew |
| C59 | Soybean__rust |
| C60 | Soybean__southern_blight |
| C61 | Strawberry___leaf_scorch |
| C62 | Strawberry__healthy |
| C63 | Sugarcane__bacterial_blight |
| C64 | Sugarcane__healthy |
| C65 | Sugarcane__red_rot |
| C66 | Sugarcane__red_stripe |
| C67 | Sugarcane__rust |
| C68 | Tea__algal_leaf |
| C69 | Tea__anthracnose |
| C70 | Tea__bird_eye_spot |
| C71 | Tea__brown_blight |
| C72 | Tea__healthy |
| C73 | Tea__red_leaf_spot |
| C74 | Tomato__bacterial_spot |
| C75 | Tomato__early_blight |
| C76 | Tomato__healthy |
| C77 | Tomato__late_blight |
| C78 | Tomato__leaf_mold |
| C79 | Tomato__mosaic_virus |
| C80 | Tomato__septoria_leaf_spot |
| C81 | Tomato_spider_mites_(two_spotted_spider_mite) |
| C82 | Tomato__target_spot |
| C83 | Tomato__yellow_leaf_curl_virus |
| C84 | Wheat__brown_rust |
| C85 | Wheat__healthy |
| C86 | Wheat__septoria |
| C87 | Wheat__yellow_rust |

## *Object Detection:*

**Apple Leaf Disease Object Detection dataset (yolo_dataset):**

The dataset consists of 8,839 images and their corresponding annotation labels. It also includes train.txt, test.txt, and valid.txt files which contain the names of the images that can be used for training, testing, and validation. Additionally, there is a datasets.yaml file that contains the relative directory for building the train, test, and validation directories. I started by developing a software program to read these text files and build the train directory which contains an images directory and its associated images labels. I did the same for the test and validation directories. The images in the dataset contain only four classes: 'Healthy', 'Black rot', 'Scab', and 'Rust'.

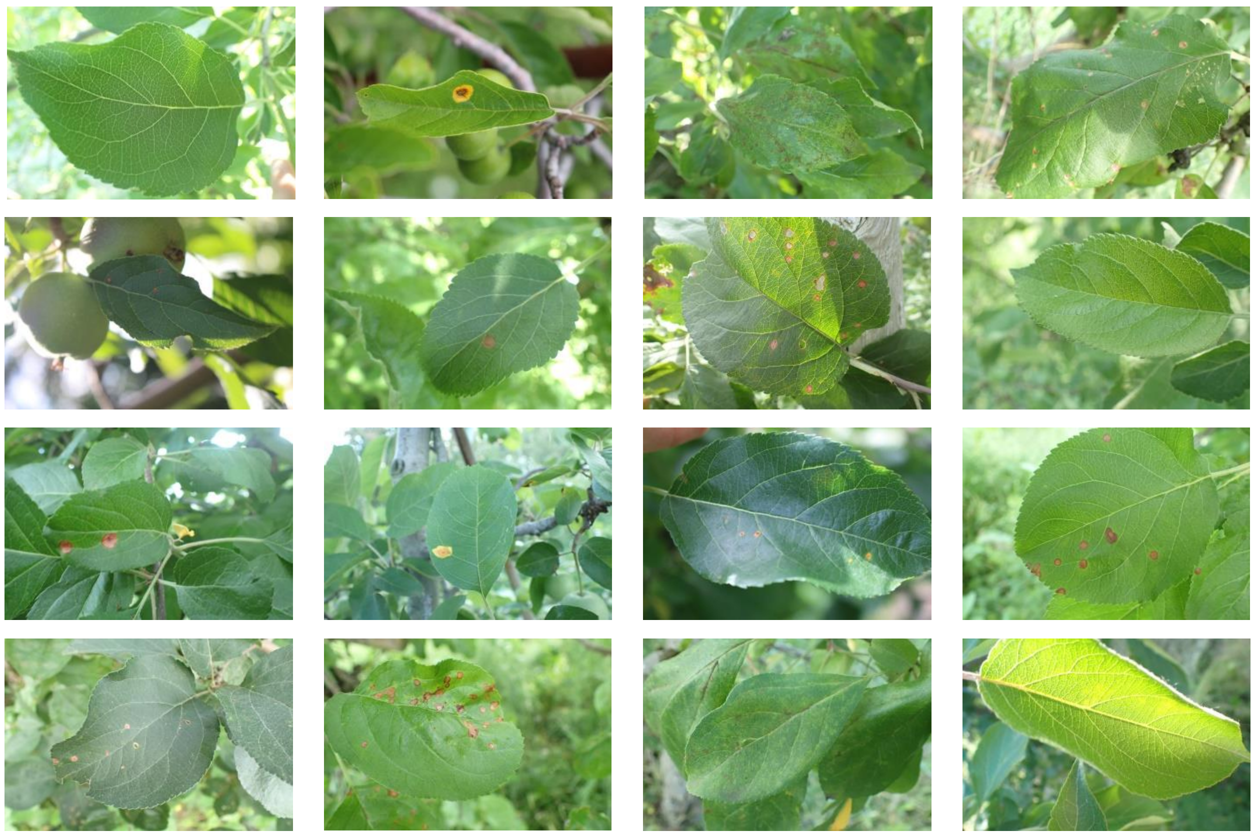


**Figure 11: Apple Leaf Disease Object Detection dataset**

**PlantDocPlus Dataset:**

The PlantDoc dataset is a comprehensive collection of 2598 images, spanning 27 distinct classes. These images, primarily captured in the field, showcase a diverse range of angles, lighting conditions, and backgrounds. Additionally, the dataset includes laboratory images for further analysis. To improve the classification of healthy and diseased tomato and potato plants, I incorporated various small datasets from Roboflow (64). Roboflow is a cutting-edge platform that empowers engineers to build and deploy computer vision models with ease. With over 250,000 engineers utilizing its tools to create datasets, train models, and deploy to production, Roboflow offers a range of utilities to seamlessly integrate computer vision into any application. The final dataset boasts 3812 annotated images, with 3721 designated for training and 541 for validation.

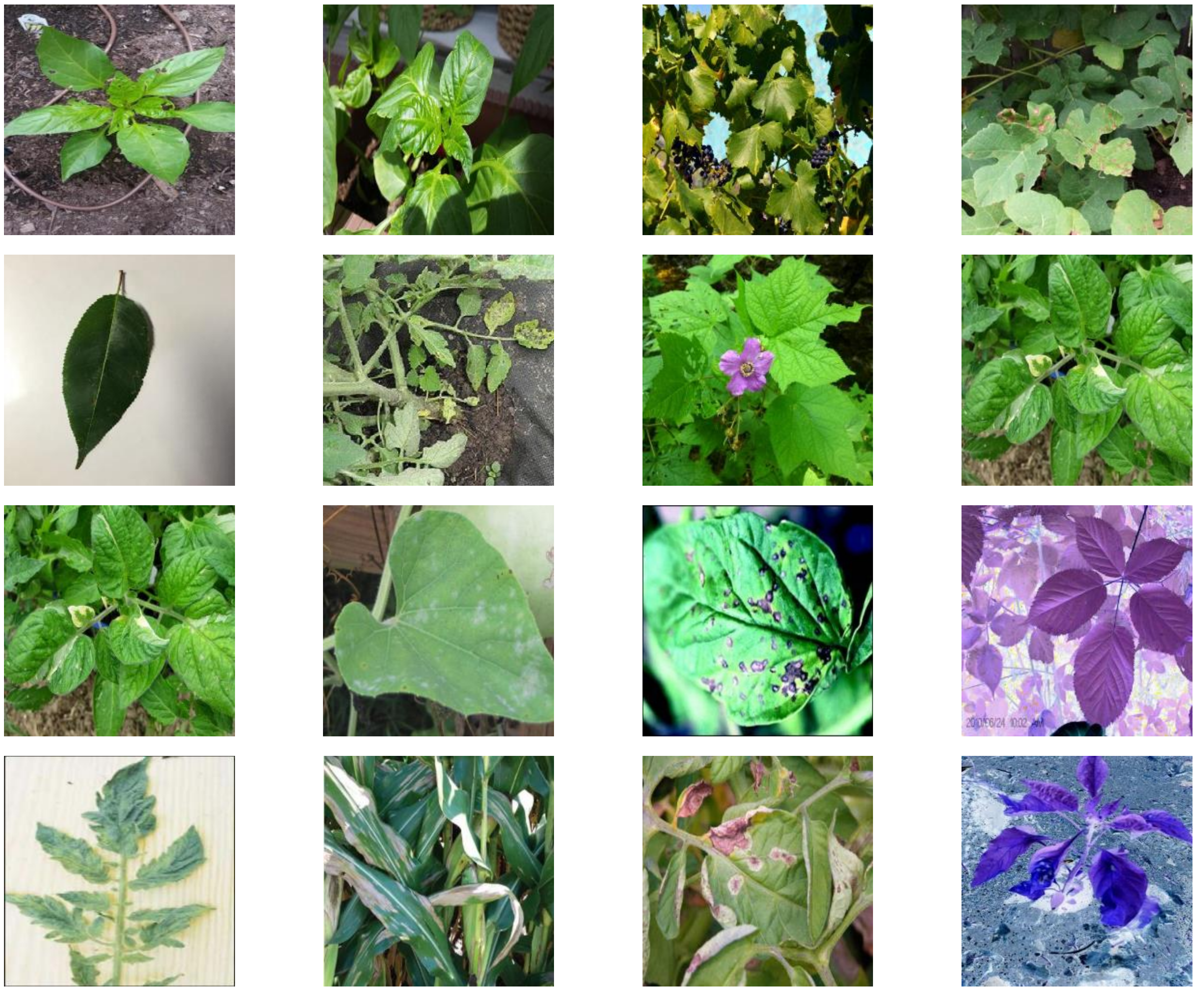


**Figure 12: PantDocPlus**

## Training:

### *Classification Model:*

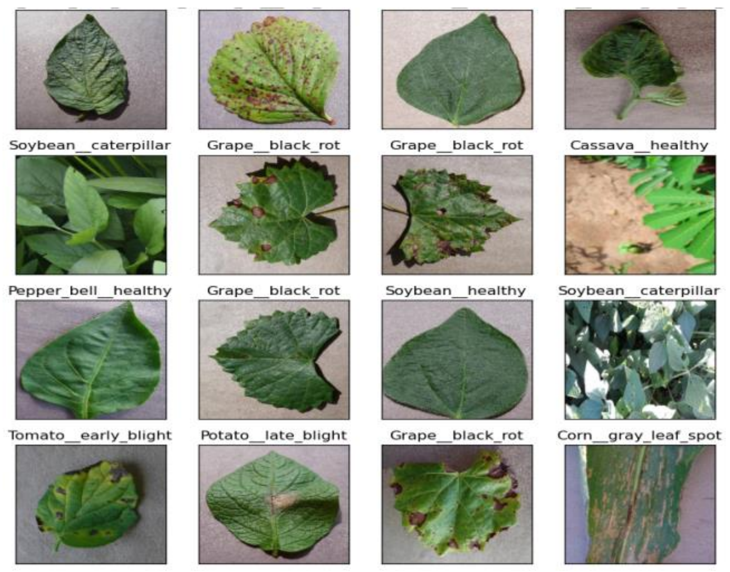
In my work, I have continued to use the same approach as in the previous level. I have chosen to use the Adam optimizer and the Categorical Crossentropy loss function. To download, develop, and train my model, I have utilized TensorFlow and Keras, all of which was done on the Kaggle platform.

**Full fine-tuningResNet50V2:**

Initially, I utilized a model that achieved an accuracy of 77% at the beginning and 86% at the end, which was not satisfactory. As a result, I decided to experiment with another model that had more parameters and layers. Despite the fact that these models were trained on 1000 classes and demonstrated good performance, the task of detecting and diagnosing plant diseases and pests proved to be challenging. This was due to the subtle differences between each class, which were difficult to discern even with the naked eye.

**Full fine-tuning ResNet152V2:**

I began by working with a dataset of 79,087 images, divided into 88 directories, each representing a class defined by the name of the plant and its disease or healthy state. The images were not pre-processed, so I performed some basic augmentation using the ImageDataGenerator library from TensorFlow, with parameters such as rescale=1/255, rotation_range=40, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2, horizontal_flip=True and vertical_flip=True. I also split the images into a test set comprising 5% of the total images and a training set comprising the remaining 95%, which was further split into an 80/20% training/validation split. The input image size was 256x256 pixels with a batch size of 32 and an RGB color scheme. Due to the large number of parameters (nearly 53 million), I could only complete 17 epochs in a 9-hour session. I then reloaded the last saved weights and trained for another 17 epochs but saw no progress, which led me to try a more complex model. The difference between the training accuracy (0.9795) and validation accuracy (0.9609) was small. I implemented early stopping when the validation loss did not improve for 10 epochs, so training stopped at epoch 26. All these factors led me to try a more complex model. The model was saved at “checkpoint4”. The last epoch’s model is at “potatoes7.h5”. You can also find the notebook, logs, and test results.



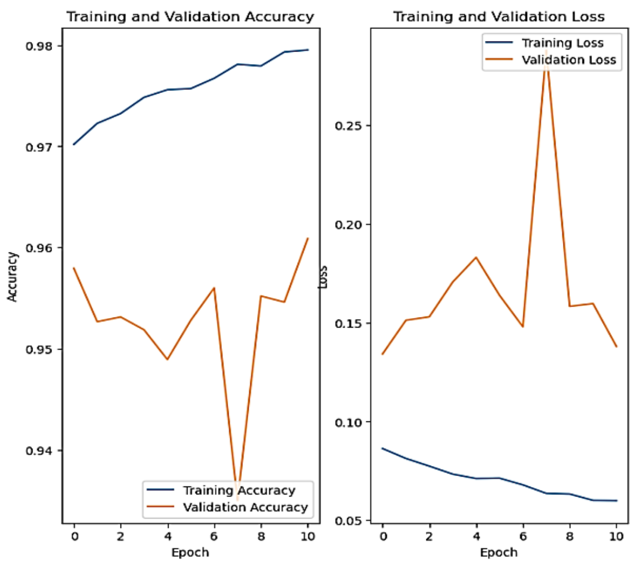


**Figure 13: ResNet152v2**

## Full fine-tuning InceptionResNetV2:

I followed the same approach as with the ResNet152V2 model, limiting the number of training epochs per session to 10. After two sessions, I observed that the output for each epoch did not change significantly. The accuracy started at 86.53% in the first epoch and increased to 95.91% by the tenth epoch. It peaked at 96.646% in epoch 17 before plateauing until epoch 20. Given that the training accuracy was 98.45%, I realized that it was necessary to stop here to avoid overfitting. Notably, the VGG19 model, which has 143.7 million trainable parameters, achieved an increase in accuracy from .71% to 99% over just 11 epochs. As a result, I decided to stop here. (It is worth mentioning that I tried many other modes, but due to technical reasons, the output was distorted.)

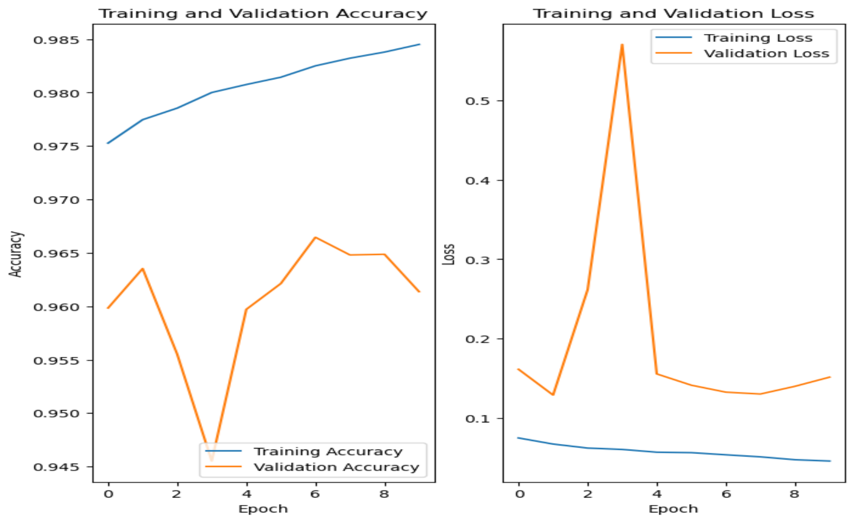


**Figure 14:InceptionResNetV2**

## *Object Detection:*

To further develop my work, I relied on the PyTorch (65) platform and the Ultralytics (66) package, which includes the YOLOv8 class. All of this was executed on the Kaggle platform.

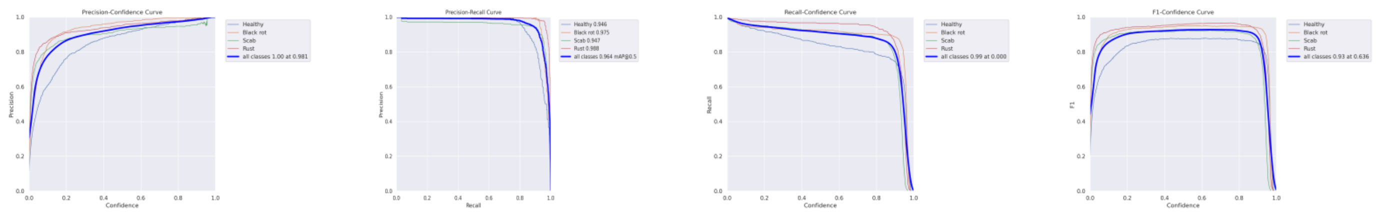
### Yolov8_Apple:

To train the model, we had several options: we could build a YOLOv8 model with our own specifications and train it from scratch, add pre-trained weights to it, or use a pre-trained model with its weights. For our purposes, we chose to use a pre-trained model with its weights and fine-tune it to our dataset. YOLOv8 is a set of models for classification, detection, and segmentation, available in nano, small, medium, large, and xlarge versions. The larger the model, the more parameters it has, making it more complex to train but also better at prediction. The nano version has fewer parameters and is designed for real-time operation. I built a nano version for use with live cameras or live video and a medium version for prediction and processing large videos on the backend with a GPU.

### Nano_Version:

I trained the nano version of the model for 98 epochs, using an input size of 640x640 pixels and a batch size of 16. The Adam optimizer was used for training. To evaluate the performance of the object detection model, I used the mAP metric, which gave me a mAP50 score of 0.964 and a mAP50-95 score of 0.945. The best model saves at best.pt.

**Note:** mAP (mean Average Precision) is a common metric used to evaluate the performance of object detection models. mAP50 refers to the mAP calculated at an Intersection over Union (IoU) threshold of 0.50. This means that for a predicted bounding box to be considered a true positive, it must have an IoU of at least 0.50 with the ground truth bounding box. An mAP50 of 0.964 means that the model has a high level of accuracy in detecting objects and correctly assigning them to their respective classes when evaluated at an IoU threshold of 0.50



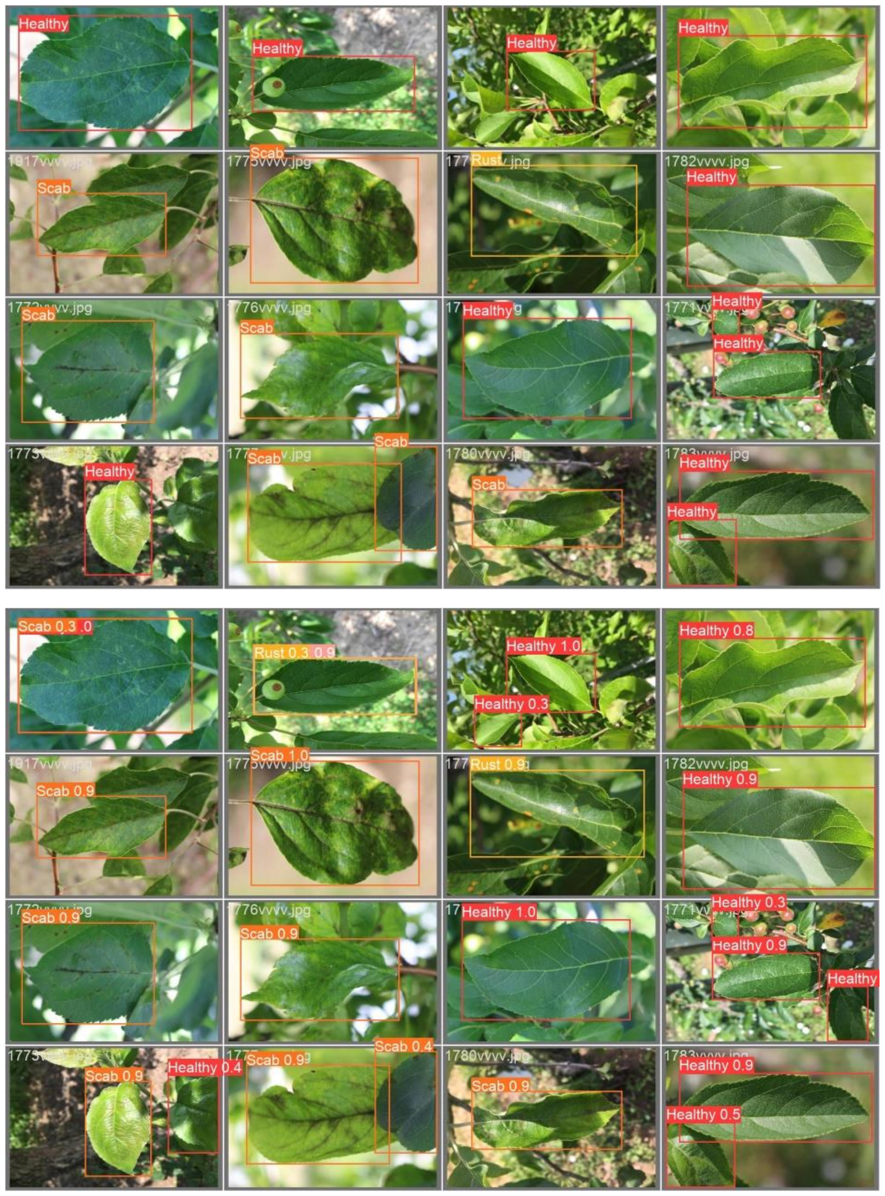
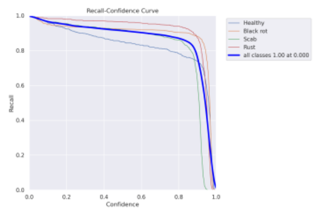


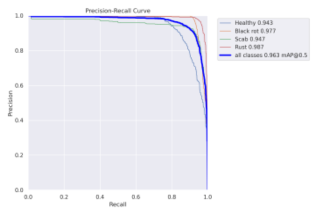
**Figure 15: Nano Version. Top is labeled images and down is  predication**

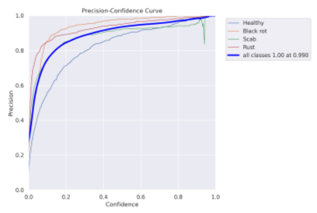
Medium **_Version:**

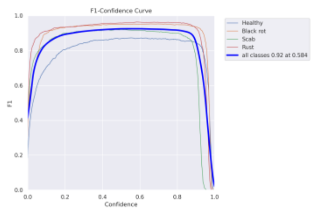
I trained the medium version of the model for 93 epochs, as the time per epoch was longer for the medium version. I chose the medium version because it had only 4 classes. The input size was 640x640 pixels and the batch size was 16, with the Adam optimizer used for training. To evaluate the performance of the object detection model, I used the mAP metric, which gave me a mAP50 score of 0.963 and a mAP50-95 score of

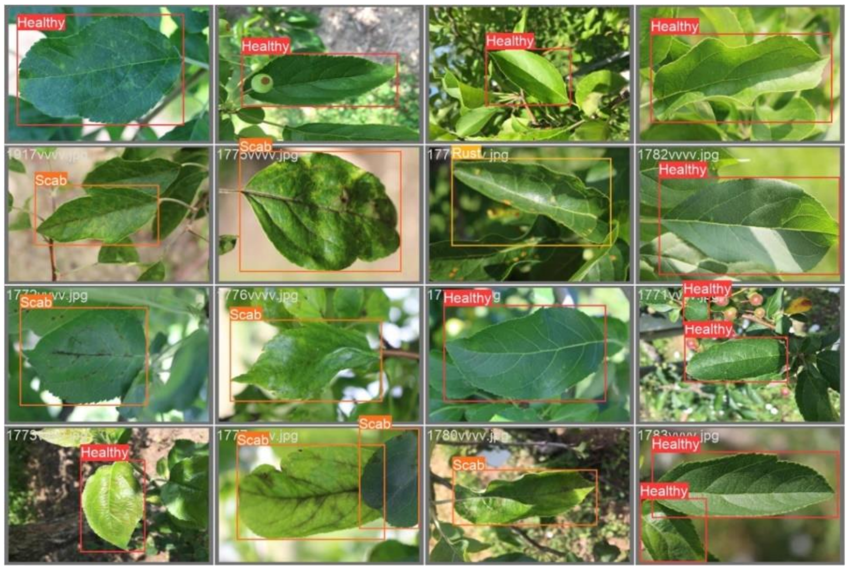
0.948. Although the difference in accuracy was not large, the medium version still performed better than the small version and was able to detect more objects in the same image. The best model saves at best.pt











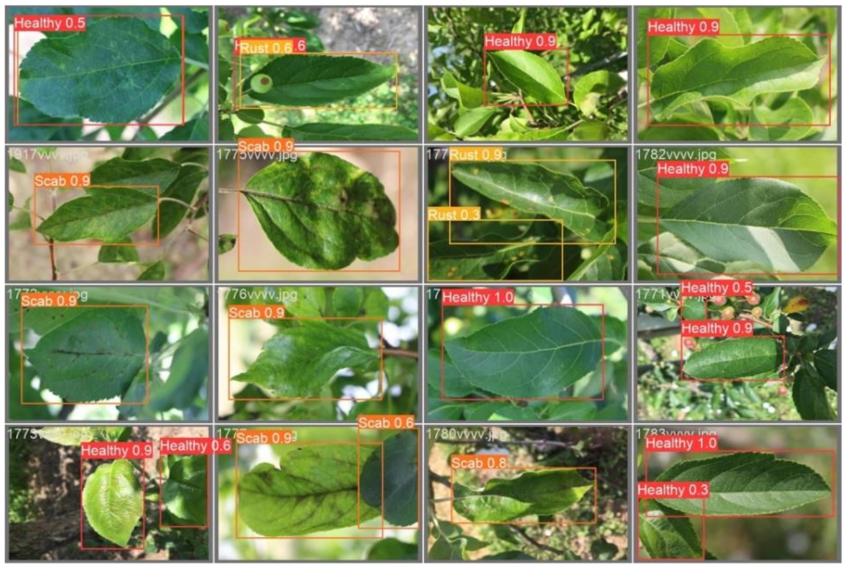


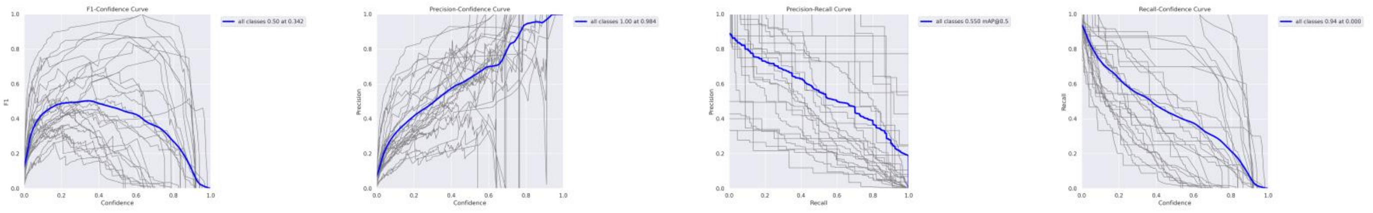
**Figure 16: medium version- Top is labeled images and down is  predication**

## PlantDocPlus:

I followed the same method that I used with the apple dataset, with the only differences being the number of epochs and the use of an xlarge model instead of a medium model, due to the number of classes 27.

## Nano Version :

I trained this model for 190 epochs, which resulted in a mAP50 score of 0.552 and a mAP50-95 score of 0.418. The best model saves at best.pt



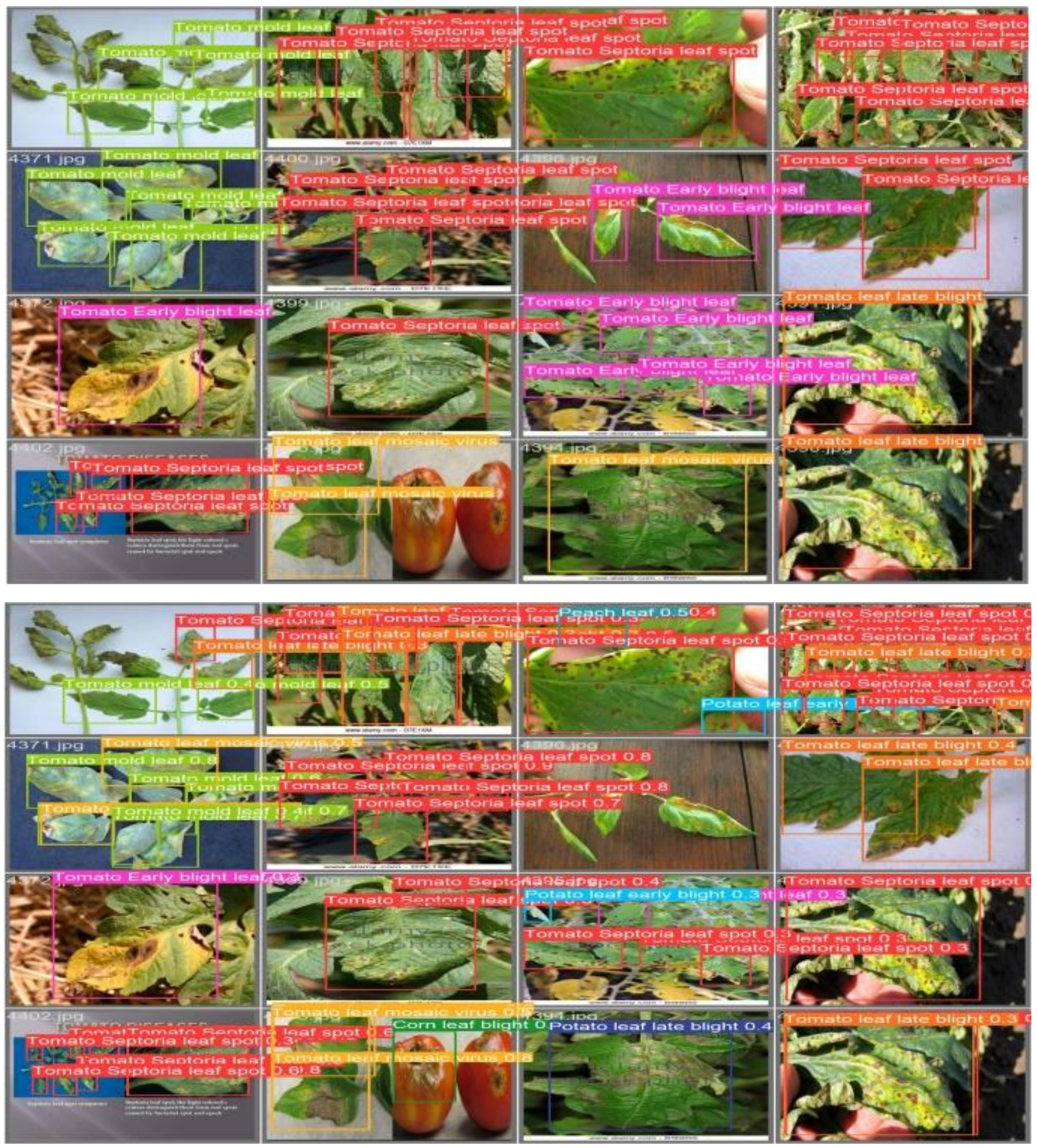
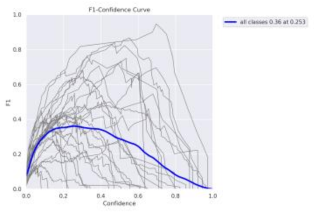


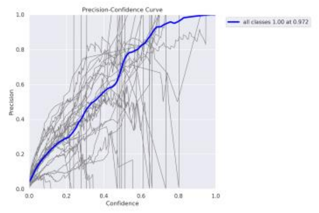
Figure 17: On the Top is the annotated images and down is the prediction images **(Nano Version)**

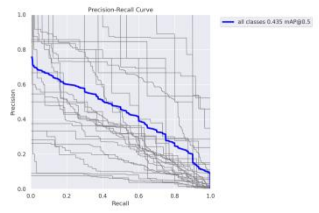
## Xlarge_Version:

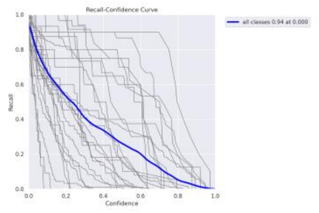
I trained this model for 295 epochs by reloading the last saved weights, which resulted in a mAP50 score of 0.435 and a mAP50-95 score of 0.331. The best model was saved as ‘best.pt’.

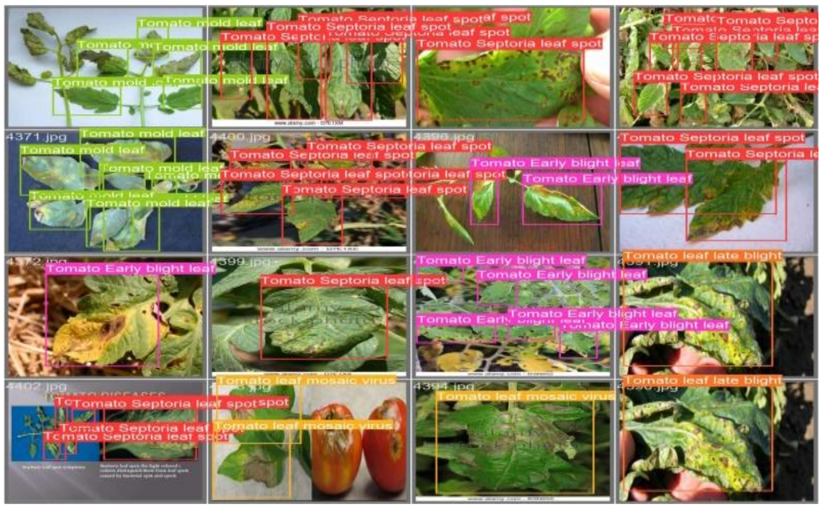
**Note:** that when reloading the last saved weights and resuming training, the mAP accuracy did not continue from where it left off but instead started from an intermediate value. I am not sure why this happened. (I think that the learning rate or other hyperparameters were changed when resuming training, which could affect the mAP accuracy. So reload is not working well)











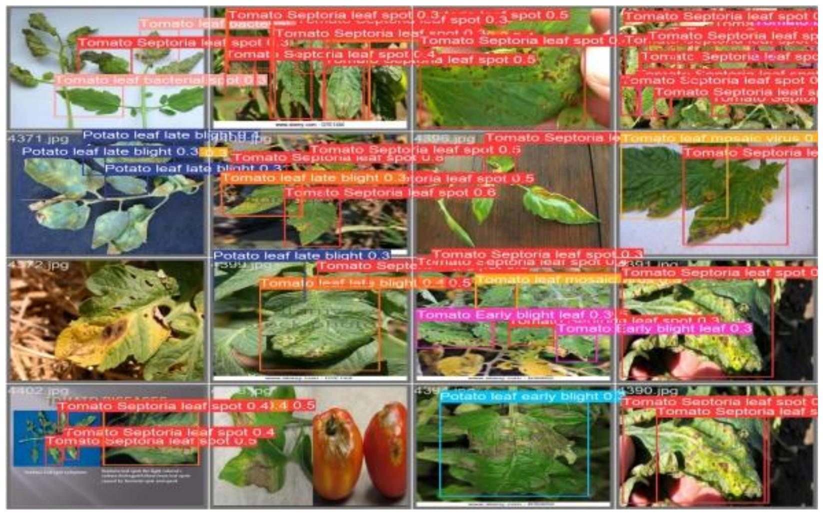


**Figure 18: On the Top is the annotated images and down is the prediction images (Xlarg_version)**

## Upgrading to the next level:

At this time, my work does not include pests and I believe that the number of images for each class is small, as is the number of classes. We have not yet worked with segmentation, so there is room to go deeper.

## Level 3:

At the time of writing this book, I am still working on it.

Due to the significant amount of time required to write the book, I have limited time remaining to complete level three. As such, I would like to note that if I am unable to finish my work on this level, future work on these topics will involve three networks: a classification network(V3)., an object detection network (V2), and a segmentation network (V1). The classification network begins by gathering data from resource  (65) - the Plant Disease Classification Merged Dataset - and Dataset (66)from paper (67). Resource  (65) refers to a GitHub site that compiles all open-source datasets available online. The Plant Disease Classification Merged Dataset is a dataset I utilized in level two, while Dataset (66) is a private dataset from paper (67), available on Google Drive, containing 97 classes. I anticipate that the final result will include 150,000 images and 150 classes of plant diseases and pests. The object detection network is constructed using Yolov8 and is based on the Plant Village dataset.

**Note: Please note that all the models I used are from Keras, as referenced in the table below.**

Table 6: Available models

| Model | Size (MB) | Top-1 Accuracy | Top-5 Accuracy | Parameters | Depth | Time (MS) per inference step (CPU) | Time (MS) per inference (GPU) |
|---|---|---|---|---|---|---|---|
| Xception | 88 | 79.0% | 94.5% | 22.9M | 81 | 109.4 | 8.1 |
| VGG16 | 528 | 71.3% | 90.1% | 138.4M | 16 | 69.5 | 4.2 |
| VGG19 | 549 | 71.3% | 90.0% | 143.7M | 19 | 84.8 | 4.4 |
| ResNet50 | 98 | 74.9% | 92.1% | 25.6M | 107 | 58.2 | 4.6 |
| ResNet50V2 | 98 | 76.0% | 93.0% | 25.6M | 103 | 45.6 | 4.4 |
| ResNet101 | 171 | 76.4% | 92.8% | 44.7M | 209 | 89.6 | 5.2 |
| ResNet101V2 | 171 | 77.2% | 93.8% | 44.7M | 205 | 72.7 | 5.4 |
| ResNet152 | 232 | 76.6% | 93.1% | 60.4M | 311 | 127.4 | 6.5 |
| ResNet152V2 | 232 | 78.0% | 94.2% | 60.4M | 307 | 107.5 | 6.6 |
| InceptionV3 | 92 | 77.9% | 93.7% | 23.9M | 189 | 42.2 | 6.9 |
| InceptionResNetV2 | 215 | 80.3% | 95.3% | 55.9M | 449 | 130.2 | 10.0 |
| MobileNet | 16 | 70.4% | 89.5% | 4.3M | 55 | 22.6 | 3.4 |
| MobileNetV2 | 14 | 71.3% | 90.1% | 3.5M | 105 | 25.9 | 3.8 |
| DenseNet121 | 33 | 75.0% | 92.3% | 8.1M | 242 | 77.1 | 5.4 |
| DenseNet169 | 57 | 76.2% | 93.2% | 14.3M | 338 | 96.4 | 6.3 |
| DenseNet201 | 80 | 77.3% | 93.6% | 20.2M | 402 | 127.2 | 6.7 |
| NASNetMobile | 23 | 74.4% | 91.9% | 5.3M | 389 | 27.0 | 6.7 |
| NASNetLarge | 343 | 82.5% | 96.0% | 88.9M | 533 | 344.5 | 20.0 |
| EfficientNetB0 | 29 | 77.1% | 93.3% | 5.3M | 132 | 46.0 | 4.9 |
| EfficientNetB1 | 31 | 79.1% | 94.4% | 7.9M | 186 | 60.2 | 5.6 |
| EfficientNetB2 | 36 | 80.1% | 94.9% | 9.2M | 186 | 80.8 | 6.5 |
| EfficientNetB3 | 48 | 81.6% | 95.7% | 12.3M | 210 | 140.0 | 8.8 |
| EfficientNetB4 | 75 | 82.9% | 96.4% | 19.5M | 258 | 308.3 | 15.1 |
| EfficientNetB5 | 118 | 83.6% | 96.7% | 30.6M | 312 | 579.2 | 25.3 |
| EfficientNetB6 | 166 | 84.0% | 96.8% | 43.3M | 360 | 958.1 | 40.4 |
| EfficientNetB7 | 256 | 84.3% | 97.0% | 66.7M | 438 | 1578.9 | 61.6 |
| EfficientNetV2B0 | 29 | 78.7% | 94.3% | 7.2M | - | - | - |
| EfficientNetV2B1 | 34 | 79.8% | 95.0% | 8.2M | - | - | - |
| EfficientNetV2B2 | 42 | 80.5% | 95.1% | 10.2M | - | - | - |
| EfficientNetV2B3 | 59 | 82.0% | 95.8% | 14.5M | - | - | - |
| EfficientNetV2S | 88 | 83.9% | 96.7% | 21.6M | - | - | - |
| EfficientNetV2M | 220 | 85.3% | 97.4% | 54.4M | - | - | - |
| EfficientNetV2L | 479 | 85.7% | 97.5% | 119.0M | - | - | - |
| ConvNeXtTiny | 109.42 | 81.3% | - | 28.6M | - | - | - |
| ConvNeXtSmall | 192.29 | 82.3% | - | 50.2M | - | - | - |
| ConvNeXtBase | 338.58 | 85.3% | - | 88.5M | - | - | - |
| ConvNeXtLarge | 755.07 | 86.3% | - | 197.7M | - | - | - |
| ConvNeXtXLarge | 1310 | 86.7% | - | 350.1M | - | - | - |

## Conclusion and future work:

In this study, specialized deep learning models were developed based on specific convolutional neural network architectures to identify plant diseases through the analysis of simple leaf images of healthy or diseased plants. The models were trained using openly available photographs from datasets, taken under both laboratory and real-world conditions in cultivation fields. The work is divided into two levels. Level one involves a classification network trained on the PlantVillage dataset, which contains nearly 54,000 images. Five models were trained, with the best performance achieved by ResNet50V2 at 99.812%. Level two comprises three main models and data: classification (V2), data collected from 14 open-source datasets containing 88 classes of plant disease combination. three models were trained, with the best performance achieved by InceptionResNetV2 at 96.646%. Object detection was performed on an open-source apple dataset with 8,839 annotated images. Nano and medium models of Yolov8 were trained, achieving mAP50 scores of 0.964 and 0.963 and mAP50-95 scores of 0.945 and 0.948, respectively. Object detection was also performed on PlantDocPlus (an extended version of PlantDoc), which contains 3,812 annotated images. Nano and Xlarge models of Yolov8 were trained, achieving mAP50 scores of 0.552 and 0.435 and mAP50-95 scores of 0.418 and 0.331, respectively.

The high level of performance achieved by the classification network demonstrates the suitability of convolutional neural networks for the automated detection and diagnosis of plant diseases through the analysis of simple leaf images. Additionally, the results indicate the importance of including real-condition images (captured in cultivation fields) in the training data to enhance model robustness. This suggests that object detection should be used on PlantDocPlus to indicate the presence of disease, while classification should be used to accurately identify the type of disease. Apple object detection should be used for direct prediction.

Furthermore, the low computational power required by the trained model to classify a given image makes it feasible to integrate into mobile applications for use on mobile devices such as smartphones or drones and other autonomous agricultural vehicles for real-time monitoring and dynamic disease detection on large-scale open-field cultivations. In the former case, a farmer in a remote location could receive an early warning about a potential threat to their cultivation, while an agronomist could have a valuable advisory tool at their disposal. A future possibility could be the development of an automated pesticide prescription system that would require confirmation by the automated disease diagnosis system before allowing farmers to purchase appropriate pesticides. This would significantly reduce the uncontrolled acquisition of pesticides that leads to their overuse and misuse, with consequent catastrophic effects on the environment.

Despite the high success rate of the developed system, there are several reasons why a better model could be achieved. For instance, the dataset I used contains only 88 classes and is sourced from various countries, which may not be particularly relevant to Egyptian farmers. If I were to merge two datasets, I would be unable to do so because I am not familiar with the properties of the diseases and cannot differentiate between similar diseases with the naked eye. If I were to make an error while merging the data, it would go unnoticed, and the model would not indicate this mistake. It is akin to a man harming himself. When it comes to object detection annotation of the dataset, my hands are tied. Although some leaves have only one disease, others have complex diseases. How can I draw a square around something that I cannot identify? Even though the data contains field images, they are not from Egypt. I have read numerous papers and have seen many countries construct their own datasets, making them available exclusively to their communities. The model becomes more robust when trained under similar prediction conditions.

As such, I propose the following: (1) anyone wishing to undertake a module on plant disease and pest detection and diagnosis in their final year project at Fayoum University must commence where I left off in order to develop a more robust and advanced prediction model that is updated periodically. (2) Collaboration between the artificial intelligence group and final year students in the College of Agriculture is essential, with at least 12 students from the College of Agriculture working on data. If you would like my assistance in your future endeavors, please lend me a hand now. (3) The Department of Computers at the Faculty of Engineering at Fayoum University must establish a database containing data collected and reviewed during the project, subject to evaluation and supervised by a specialized doctor for ease of access and organization.

## Reference:

1. *Crop Infection Detection using YOLO.* **Satwik Ram Kodandaram, Kushal Honnappa.** 2021, ResearchGate.

2. *UsingDeepLearningforImage-BasedPlantDiseaseDetection.* **Sharada P.Mohanty, DavidP.Hughes, MarcelSalathé.** 2016, frontiers, p. 10.3389/fpls.2016.01419.

3. *Deep learning models for plant disease detection and diagnosis.* **Ferentinos, Konstantinos P.** 2018, Elsevier, pp. Computers and Electronics in Agriculture 145 (2018) 311–318.

4. *Plant diseases and pests detection based.* **Jun Liu and Xuewei Wang.** 2021, Plant Methods, pp. 10.1186/s13007-021-00722-9.

5. **Géron, Aurélien.** *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow".* s.l. : O'Reilly, 2019.

6. *Some Studies in Machine Learning Using the Game of Checkers.* **Samuel, Arthur L.** 1959, IBM.

7. **Mitchell, Tom M.** *Machine Learning.* s.l. : McGraw-Hill , 1997.

8. **Mahesh, Batta.** *Machine Learning Algorithms -A Review.* s.l. : ResearchGate, 2019.

9. **wikipedia.** Weak supervision. *wikipedia.* [Online] [Cited: 6 24, 2023.] https://en.wikipedia.org/wiki/Weak_supervision.

10. **Brownlee, Jason.** Semi-Supervised Learning With Label Spreading. *machinelearningmastery.* [Online] 1 2021, 4. [Cited: 6 24, 2023.] https://machinelearningmastery.com/semi-supervised-learning-with-label-spreading/.

11. **Brahme, Anders.** *Comprehensive Biomedical Physics.* s.l. : Elsevier , 2014.

12. **wikipedia.** Artificial neural network. *wikipedia.* [Online] [Cited: 6 24, 2023.] https://en.wikipedia.org/wiki/Artificial_neural_network#cite_note-2.

13. *Reducing the Dimensionality of Data with Neural Networks.* **Hinton and Salakhutdinov.** 2006, Science, pp. 313(5786):504–7.

14. *A survey of deep neural network architectures and their applications.* **al, Liu et.** 2017 , Neurocomputing, pp. 2017;234:11–26.

15. *Deep Learning Methods for Vision.* **presented, Rob Fergus.** s.l. : 2012 Conference on Computer Vision and Pattern Recognition (CVPR), 2012. 2012 Conference on Computer Vision and Pattern Recognition (CVPR).

16. *Representation learning: a review and new perspectives.* **al., Bengio et.** 2013 , IEEE Trans Pattern Anal Mach Intell, pp. 2013;35(8):1798–828.

17. *Ask the Locals: Multi-Way Local Pooling for Image Recognition.* **Y-Lan Boureau, Nicolas Le Roux, Francis Bach, Jean Ponce, and Yann LeCun.** Barcelona, Spain : ICCV), 2011. International Conference on Computer Vision (ICCV). pp. 2651-2658 .

18. *Stochastic Pooling for Regularization of Deep Convolutional Neural Networks.* **Matthew D. Zeiler and Rob Fergus.** s.l. : arXiv, 2013, arXiv.

19. *ImageNet Classification with Deep Convolutional Neural Networks.* **Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton.** 2012, research highlights , p. DOI:10.1145/3065386.

20. *Going Deeper with Convolutions.* **Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich .** 2015, Cornell University.

21. *Very deep convolutional networks for.* **Karen Simonyan and Andrew Zisserman .** s.l. : arxiv, 2014. 1409.1556.

22. *Aggregated Residual Transformations for Deep Neural Networks.* **Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, Kaiming He.** s.l. : arxiv, 2017. 1611.05431.

23. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning.* **Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, Alex Alemi.** s.l. : arxiv., 2016. 1602.07261.

24. *Densely Connected Convolutional Networks.* **Gao Huang, Zhuang Liu, Laurens van der Maaten, and others .** s.l. : IEEE, 2017. 8099726.

25. *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications.* **Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam.** s.l. : arxiv, 2017. 1704.04861.

26. *SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size.* **Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, Kurt Keutzer.** s.l. : arxiv, 2016. 1602.07360.

27. *Recent Studies of Image and Soft Computing Techniques for Plant Disease Recognition and Classification.* **H. Sabrol, Satish Kumar.** 2015. 10.5120/ijca2015905982.

28. *Plant classification using convolutional neural networks.* **Hulya Yalcin; Salar Razavi.** s.l. : IEEE, 2016. 7577698.

29. *Crop pest classification based on deep convolutional neural network and transfer learning.* **K. Thenmozhi, U. Srinivasulu Reddy.** s.l. : Computers and Electronics in Agriculture, 2019. S0168169919310695.

30. *Plant disease identification using explainable 3D deep learning on hyperspectral images.* **Koushik Nagasubramanian , Sarah Jones , Asheesh K Singh , Soumik Sarkar , Arti Singh , Baskar Ganapathysubramanian .** s.l. : pubmed, 2019. 10.1186/s13007-019-0479-8.

31. *Crop conditional Convolutional Neural Networks for massive multi-crop plant disease classification over cell phone acquired images taken on real field conditions.* **A. Picón, Maximiliam Seitz, J. Echazarra.** s.l. : semanticscholar, 2019. 208643496.

32. *You Only Look Once: Unified, Real-Time Object Detection.* **Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi.** s.l. : arxiv, 2016. 1506.02640.

33. *Real-Time Flying Object Detection with YOLOv8.* **Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi.** s.l. : arXiv, 2023. 2305.09972.

34. **glenn-jocher.** ultralytics. *github.* [Online] [Cited: 6 25, 2023.] https://github.com/ultralytics/ultralytics.

35. *"Fully Convolutional Networks for Semantic Segmentation.* **Jonathan Long, Evan Shelhamer, Trevor Darrell.** s.l. : arXiv, 2015. 1411.4038.

36. *Mask R-CNN.* **Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick.** s.l. : arxiv, 2017. 1703.06870.

37. *Plant Disease Detection using Deep Learning.* **Murk Chohan, Adil Khan, Rozina Chohan, Saif Hassan Katpar, Muhammad Saleem Mahar.** 1, s.l. : International Journal of Recent Technology and Engineering, 2020, Vol. 9. 2277-3878.

38. *Plant disease identification from individual lesions.* **Barbedo, Jayme Garcia Arnal.** s.l. : ScienceDirect, 2019. 13083-886,.

39. **ARUN PANDIAN J,.** Data for: Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network. *mendeley.* [Online] [Cited: 6 26, 2023.] https://data.mendeley.com/datasets/tywbtsjrjv/1.

40. **spMohanty.** PlantVillage-Dataset. *github.* [Online] [Cited: 6 26, 2023.] https://github.com/spMohanty/PlantVillage-Dataset.

41. *An open access repository of images on plant health to enable the development of mobile disease diagnostics.* **David. P. Hughes, Marcel Salathe.** s.l. : arXiv, 2016. 1511.08060.

42. **vipoooool.** New Plant Diseases Dataset. *kaggle.* [Online] [Cited: 6 26, 2023.] https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset.

43. *tensorflow.* [Online] https://www.tensorflow.org/.

44. *keras.* [Online] https://keras.io/api/applications/.

45. *kaggle.* [Online] https://www.kaggle.com/.

46. *Adam: A Method for Stochastic Optimization.* **Diederik P. Kingma, Jimmy Ba.** s.l. : arxiv., 2017 . 1412.6980.

47. *MGA-YOLO: A lightweight one-stage network for apple leaf disease detection.* **Yiwen Wang, Yaojun Wang and Jingbo Zhao.** s.l. : frontiersin, 2022. 2022.927424.

48. [Online] https://www.kaggle.com/datasets/fabinahian/plant-disease-65-classes.

49. [Online] https://doi.org/10.1145/3371158.3371196.

50. [Online] https://www.kaggle.com/datasets/coffeedisease/coffee-plant-disease.

51. [Online] https://www.kaggle.com/datasets/olyadgetch/wheat-leaf-dataset.

52. [Online] https://www.kaggle.com/datasets/dhenyd/chili-plant-disease.

53. [Online] https://data.mendeley.com/datasets/bycbh73438/1.

54. [Online] https://www.kaggle.com/datasets/vbookshelf/rice-leaf-diseases.

55. [Online] https://www.kaggle.com/datasets/shayanriyaz/riceleafs.

56. [Online] https://www.kaggle.com/datasets/kareem3egm/cucumber-plant-diseases-dataset.

57. [Online] https://www.kaggle.com/datasets/sadmansakibmahi/plant-disease-expert.

58. [Online] https://www.kaggle.com/datasets/asheniranga/leaf-disease-dataset-combination.

59. *Identifying multiple plant diseases using digital image processing.* **L. V. S. T. T. BARBEDO, J. G. A.; KOENIGKAN.** s.l. : Biosystems Engineering, 2016, Vol. 147. Biosystems Engineering.

60. [Online] https://www.kaggle.com/datasets/prabhakaransoundar/sugarcane-disease-dataset.

61. [Online] https://www.kaggle.com/datasets/pungliyavithika/sugarcaneleaf-disease-classification.

62. [Online] https://roboflow.com/.

63. [Online] https://pytorch.org/.

64. [Online] https://ultralytics.com/.

65. **xm194.** Plant_disease_recognition. *github.* [Online] [Cited: 6 30, 2023.] https://github.com/xml94/Plant_disease_recognition.

66. *Google Drive.* [Online] https://drive.google.com/file/d/1HhtA939IwSjrN2XKRyeTgMQnTaY4zniA/view.

67. *Machine learning techniques for plant disease detection: an evaluation with a customized dataset.* **Amatullah Fatwimah Humairaa Mahomodally, Geerish Suddul, Sandhya Armoogum.** 2252-8776, s.l. : International Journal of Informatics and Communication Technology, 2023, Vol. 12. 10.11591/ijict.v12i2.pp127-139.

68. **ALINE DOBROVSKY.** Plant Disease Classification Merged Dataset. *kaggle.* [Online] [Cited: 6 27, 2023.] https://www.kaggle.com/datasets/alinedobrovsky/plant-disease-classification-merged-dataset.

69. **LIAOFANYISHI.** Apple Leaf Disease Object Detection dataset. *kaggle.* [Online] [Cited: 6 27, 2023.] https://www.kaggle.com/datasets/df248b05bcf0246cc0b7add831501d83a30396900302f3e6d01ab293471150f4.