



# **Cotton Disease Detection Using Deep Learning**

Prepared By

Name: Youssif Ali Alshalakany

ID No.: 243000646

Name: Youssef Abdelmotaleb Saad

ID No.: 243000658

Submitted in fulfillment of the requirements for the degree Master of  
Science in Artificial Intelligence

At

College of Informatics

Midocean University

UAE

Supervisor Name: Prof. Dr. Yasser Abdelhamid

College of Informatics, Midocean University, Comoros

Email: 243000646@st.midocean.edu.km

Email: 243000658@st.midocean.edu.km

## **Acknowledgements**

First and foremost, I would like to express my deepest gratitude to Prof. Dr. Yasser Abdelhamid, my supervisor, for his invaluable guidance, continuous support, and patience throughout the course of this research. His insightful feedback and encouragement have been instrumental in shaping the direction and quality of this work.

I am also thankful to Midocean university, as well as all the College of Informatics members and administrative staff, for providing the resources and academic environment necessary for completing my thesis.

Special thanks to my Colleagues who shared their knowledge and experiences with me, and who made this journey intellectually stimulating and personally rewarding.

I am especially grateful to my family for their unwavering love, understanding, and support during this journey. Their encouragement was a constant source of strength.

## Table of contents

| Topic                                              | Page No. |
|----------------------------------------------------|----------|
| Abstract                                           | VI       |
| Chapter (1)<br>Introduction                        |          |
| 1.1 Introduction                                   | 2        |
| 1.2 Problem Definition                             | 3        |
| 1.3 Thesis Objectives                              | 4        |
| 1.4 Research Questions                             | 4        |
| 1.5 Thesis Contribution                            | 5        |
| 1.6 Thesis Organization                            | 5        |
| Chapter (2)<br>Overview                            |          |
| 2.1 Literature review                              | 7        |
| 2.2 Related Work                                   | 12       |
| 2.3 Challenges and Opportunities                   | 13       |
| Chapter (3)<br>Proposed Approach                   |          |
| 3.1 Research Methodology                           | 15       |
| 3.2 Dataset                                        | 23       |
| 3.3 The research model                             | 25       |
| Chapter (4)<br>Experimental Results and Discussion |          |
| 4.1 Results                                        | 30       |
| 4.2 Results Discussion                             | 64       |
| Chapter (5)<br>Conclusion and Future Work          |          |
| 5.1 Conclusion                                     | 69       |
| 5.2 Recommendations                                | 70       |
| 5.3 Future works                                   | 70       |
| References                                         | 71       |

## Table of Tables

| Table                                             | Page No. |
|---------------------------------------------------|----------|
| Table (3.1) The goal of each library              | 15       |
| Table (3.2) The research dataset                  | 23       |
| Table (4.1) Performance Comparison Between Models | 63       |

## List of figures

| Figure                                                                   | Page No. |
|--------------------------------------------------------------------------|----------|
| Figure (3.1) The Research Model                                          | 25       |
| Figure (4.1) confusion matrix – Binary classification                    | 31       |
| Figure (4.2) training and variation accuracy                             | 32       |
| Figure (4.3) Model accuracy and loss                                     | 32       |
| Figure (4.4) ROC curve- Binary classification                            | 33       |
| Figure (4.5) Confusion Matrix – Multi-class                              | 35       |
| Figure (4.6) Training and validation accuracy                            | 36       |
| Figure (4.7) model accuracy and loss                                     | 37       |
| Figure (4.8) ROC curves- One VS- Rest                                    | 37       |
| Figure (4.9) Confusion Matrix – mobilenetv2                              | 39       |
| Figure (4.10) training accuracy and loss                                 | 40       |
| Figure (4.11) model accuracy – loss                                      | 41       |
| Figure (4.12) ROC curves- One VS- Rest                                   | 42       |
| Figure (4.13) Confusion Matrix – ResNet50                                | 43       |
| Figure (4.14) Comparison between the training accuracy and loss          | 44       |
| Figure (4.15) model accuracy and loss                                    | 45       |
| Figure (4.16) ROC curves – One-VS- REST                                  | 46       |
| Figure (4.17) Confusion Matrix – EfficientNetB0 (Multi-class)            | 47       |
| Figure (4.18) comparison between the training accuracy and training loss | 49       |
| Figure (4.19) Model accuracy and loss                                    | 50       |

|                                                                          |    |
|--------------------------------------------------------------------------|----|
| Figure (4.20) ROC curves- One-Vs-Rest                                    | 51 |
| Figure (4.21) Confusion matrix – VGG16 (multi-class)                     | 52 |
| Figure (4.22) Training and validation accuracy                           | 53 |
| Figure (4.23) model accuracy and loss                                    | 54 |
| Figure (4.24) ROC curves – One -VS- Rest                                 | 55 |
| Figure (4.25) Confusion matrix – VGG19 (multi-class)                     | 56 |
| Figure (4.26) comparison between the training accuracy and training loss | 57 |
| Figure (4.27) model accuracy and loss                                    | 58 |
| Figure (4.28) ROC curves- One- VS- Rest                                  | 59 |
| Figure (4.29) confusion matrix DenseNet121 (Multi-class)                 | 60 |
| Figure (4.30) comparison between training accuracy and training loss     | 61 |
| Figure (4.31) model accuracy and loss (Densenet121)                      | 62 |
| Figure (4.32) ROC curves One VS Rest (DenseNet121)                       | 63 |

## List of Abbreviations

- **Adam**: Adaptive Moment Estimation
- **API**: Application Programming Interface
- **CNN**: Convolutional Neural Network
- **CWT**: Continuous Wavelet Transform
- **DenseNet121**: Densely Connected CNN
- **DL**: Deep Learning
- **DNN**: Deep Neural Network
- **EffNetB0**: EfficientNetB0
- **FAO**: Food and Agriculture Organization
- **FFT**: Fast Fourier Transform
- **GPU**: Graphics Processing Unit
- **IoT**: Internet of Things
- **Kern**: High-level neural network API
- **KNN**: K-Nearest Neighbors
- **ML**: Machine Learning
- **PIL**: Python Imaging Library
- **ReLU**: Rectified Linear Unit
- **ResNet152v2**: Residual Network with 152 layers - version 2
- **ResNet50**: Residual Neural Network - 50 layers
- **RGB**: Red, Green Blue
- **RNN**: Recurrent Neural Network
- **SGD**: Stochastic Gradient Descent
- **SVM**: Support Vector Machine
- **VGG16/VGG19**: Visual Geometry Group Network (16/19 layers)
- **XAI**: Explainable Artificial Intelligence

**Abstract:**

The main objective of the research is to diagnose cotton Disease through image analysis using deep learning. Where it is traditional diagnosis method relies on manual inspection, which is labor-intensive and prone to human mistakes. To overcome this problem, has been proposed Image Analysis Using deep learning technologies, particularly (CNNs), cotton pictures can be diagnosed with high precision to enable faster and more accurate disease detection. The main parts of the affected sheets are also highlighted by the type of disease in the database. This allows for early detection of the disease, reducing agricultural losses, increasing productivity, and improving the sustainability of cotton farming.

Approximately 80-90% of cotton diseases affect leaves, causing up to 16% budget loss and potentially 30-50% crop loss without control. This research explores AI techniques from cotton leaves images.

## **Chapter (1)**

### **An access to the research**

## **Chapter (1)**

In this chapter, the researchers will clarify the research problem, the research questions, research objectives and the research plan.

### **1.1 Introduction**

Artificial intelligence plays an important role in various fields, including: the agricultural field, as artificial intelligence has provided unprecedented capabilities to analyze large amounts of crop data, which has contributed to discovering patterns and risk factors, predicting epidemics, and enhancing the allocation of resources during agricultural crises (Sohrab, et al.,2024)

Thanks to these technologies, agricultural workers have more advanced tools to detect and prevent diseases. The roles of artificial intelligence in the agricultural field, especially in the cotton crop, vary between data analysis and pattern recognition, and providing predictive models to prevent the spread of cotton crop diseases, and others (Kumar, et al., 2020).

The ability of artificial intelligence to analyze large data sets accurately and quickly is one of its most prominent advantages that has made it one of the important technologies in the agricultural field. Thanks to machine learning algorithms, complex data can be analyzed, such as components and characteristics of the cotton crop, and genetic information to detect patterns that may indicate the beginning of the spread of a specific disease or the emergence of a new disease. Artificial intelligence systems are used in the field of agricultural diagnosis of agricultural crops (Delfosse,2023).

AI helps analyze large amounts of genetic data on crops to identify mutations that may indicate the likelihood of developing certain agricultural diseases. These technologies are used to detect agricultural risks and develop appropriate treatments. AI contributes to infectious disease

surveillance by analyzing data from various sources, such as news reports and agricultural records, to identify early signs of disease spread among cotton crops (Amrao, et al., 2020)

Recently, new, and refined techniques have attracted many researchers in the field of cotton sheet disease and pest detection and classification. In particular, the identification of potential diseases or parasites in cotton is based on traditional methods. There is a large farm suitable for cotton planting, but only limited attention is paid to the production of cotton crops (Muhammad, et al.,2024)

Traditionally, experts detect and identify these diseases of plants and pests on bare eyes. Eye shadow determination is considered a loss of low levels of accuracy for detecting disease. Among the various advanced technologies, advanced technologies, they have helped structure the system to support non-automatic recognition of diseases and plant pests, and to increase the accuracy of corrective actions. With the help of advanced technologies, plant diseases have been reduced, which has increased performance that has contributed to increasing the economy by increasing production. (Rajasekar, et al.,2021)

Of the various diseases and parasites, approximately 80-90% were found in cotton leaves. It has been observed that there could be about 16% of budget destruction due to plant syndrome. However, without control measures, it can cause 30%–50% of loss (Sushreeta, et al.,2021). Owing to the importance of using deep learning techniques in detecting the different crops diseases topic, this research is conducted.

## **1.2 Problem Definition**

The cotton plant is susceptible to several disorder attacks due to temperature fluctuation, diseases, and pests. Indeed, the whole world produced nearly 576 kg per hectare of cotton crops, where only 10% of production loss occurred due to different cotton diseases and this results in

the downfall of the economy of both the farmer and the country. Deep learning captures interest to enhance performance in classifying various tasks, which aids in optimizing data for human intervention. In recent applications, deep learning demonstrates significant potential for interpreting the different data related to crops. The core research problem lies in the challenges associated with traditional diagnostic methods for cotton diseases, which primarily rely on visual inspection. These methods are characterized by time-consumption, human intervention, and error, leading to delays in treatment and exacerbation of crop losses. Consequently, there is an urgent need to develop faster and more accurate diagnostic approaches. Also, this topic was chosen because Image analysis using deep learning enables quicker and more accurate interpretation of medical images, resulting in better patient outcomes, tailored treatment planning, and streamlined healthcare processes. In addition, I have chosen this topic owing to the importance of the cotton crop in the world economy and because of the many diseases that affect cotton and which lead to the downfall of the economy. This leads to the main research question: How can cotton diseases be diagnosed through image analysis using deep learning?

### **1.3 Thesis Objectives**

- Determining the challenges for effective Cotton Disease through Image Analysis Using Deep learning.
- Determining the requirements for effective Cotton Disease through Image Analysis Using Deep.

### **1.4 Research Questions**

- What are the challenges for effective Cotton Disease through Image Analysis Using Deep learning?

- What are the requirements for effective Cotton Disease through Image Analysis Using Deep?

### **1.5 Thesis Contribution**

There is a research gap owing to the lack of studies regarding Cotton Disease through Image Analysis Using Deep learning. Also, Deep learning method is one of the latest methods that can be used to identify the crops diseases.

### **1.6 Thesis Organization**

The thesis contains 5 chapters. The first chapter is about an access to the research. The second chapter is study overview in which the literature review was submitted. On the other hand, the third chapter was about the methodology and the proposed model/ The fourth chapter was about the experimental Results and Discussion, while the fifth chapter was about the conclusion and Future Work

## **Chapter (2)**

### **Overview**

## Chapter (2)

### 2.1 Literature review

The research conducted by (Sharma et al. ,2022) is significant for its thorough evaluation of machine learning and deep learning methods in different application areas. Their examination of common methods in these areas may offer important perspectives to enhance our own techniques.

Similarly, Aggarwal et al. (2023) introduced an effective system employing various deep-learning methods to forecast diseases in rice leaves. Utilizing pre-trained models for feature extraction and implementing different machine learning classifiers, the suggested method exceeds existing techniques, reaching as high as 94% accuracy in identifying rice diseases (9). For instance, Kalpana et al. (2023) suggested utilizing deep learning for detecting cotton diseases through various transfer learning models such as VGG-16 and Inception-V3, achieving an accuracy rate of 96%. (Sharma, & Mittal, 2022).

Also, (Reddy et al.,2023) created a deep-learning model based on CNN to identify cotton diseases. In their tests involving 720 images, the authors achieved an accuracy rate of 89%.

In addition, (Chander and Upendra Kumar, 2022) employ various models. The dataset, made up of images of cotton plants and leaves, is gathered manually. These pictures are subsequently utilized for the training and evaluation of the models. The suggested approach shows an improved classification accuracy and decreases the processing time. The writers noted that the efficiency achieved was as high as 96%.

In the study of (Caldeira, et al., 2021), the researchers studied pots and revealing problems such as lesions in cotton sheets and looking at the harvest photos. Cotton is a very important culture in the world, but pests and disease problems are especially common in tropical regions. It is

difficult for farmers to recognize these issues early and take appropriate measures. This study offered a deep learning-based approach as a tool, as well as teaching computers to identify and understand images of cotton sheets. This encouraged farmers to control the health of their cotton mills and make decisions in the best way to care for them. Two models of training, ResNet50 and GoogleNet, produced results with good accuracy of 86.6% and 89.2%, respectively. The new detailed learning model was 25% more accurate than previous methods of analyzing images, including supported vector vehicles, closest K neurobors, artificial neuros, and neuromelting systems. This means that farmers can assess the health of field factories by using these complex IT methods.

In the study of (Hassan, et al, 2022), the researchers focused on cotton since it is the most important financial culture for the country's economy. However, diseases can reduce cotton quality and yield, and in many cases early detection of these diseases is complicated. To address is a problem. The researchers have developed a system that checked and determined cotton board disease using unique features and an advanced computer program (deep training). The researchers focused on systemic diseases affecting cotton, such as alternatives, bacterial packaging, circus, and grey mold. The researchers chose a set of 522 photographs of cotton sheets in a field near Multan City for our research. Our system worked well, as confirmed with an accuracy of 85.42%. This indicated that disease can be diagnosed correctly in most situations. Acceptable and reliable system performance is also demonstrated by the Cappa coefficient, which is the F1, accuracy and magazine evaluation. The methods are beneficial to farmers and scientists. Early detection of the disease will give farmers a large amount of money and allow them to manage their cotton crops more effectively. This supported the general stability of the national economy.

In another study conducted by (Azoth, et al., 2021), the researchers focused on cotton as an important product of Ethiopian agriculture, but it faced problems such as diseases and parasites that are difficult to see with the naked eye. This work was designed to develop computer models, and used CNN methods in particular to better identify pests and diseases in cotton boards. They focused on common issues such as leaves, spiders, and minors in bacterial waterfalls. This model was formed with approximately 2,400 photographs, with 600 images in each category, which can identify a variety of issues. Models were developed and tested using Python, Keras, Tensorflow, Jupyter, and other programming languages. The identification of 96.4% of cotton plaque disease and pests has been successfully identified according to the high model. This showed how computer technology can complete traditional recognition methods and solutions to cotton factory problems in real time.

In another study conducted by (Rai,2022), the researcher used computers to identify the great achievements of agriculture, looking at leaf photographs, automatically detecting and classifying plant diseases. In this case, they focused on cotton plants. By using real photos, they applied specialized automated learning, known as deep training, to predict cotton plant health. They reached it using a large collection of pictures of 2293 of plants and cotton sheets corresponding to one of the four training categories of IT models. They developed and tested the models, especially using the Python 3.6.9 version and resources such as Tensorflow, GoogleColab (an online coding tool), Kerats, Keers and more. Their computer models were surprisingly effective in their work. The remarkable accuracy of 97.98% with proper identification and classification of cotton plant disease was noteworthy. This showed that he surpasses other approaches that have been discussed in previous research. The researchers

wanted to determine whether a cotton plant is diseased, check the severity of the problem, minimize errors, and save farmers' time.

In the study of (Pandian, et al., 2022) the researchers presented an improved new computer program called ResNet197 to identify various diseases in vegetable sheets. This ResNet197 has 197 layers and is extremely powerful. It was used to analyze a large amount of dataset containing images of 22 different healthy plants and patients. Filling in our program worked as much as possible and the reserachers taught it in many photos using different ways of manipulating images such as pruning, coups, scaling and more. The dataset contained 154,500 photographs, divided into 103 categories of healthy and sick leaves. After 1000 training cycles of reliable Graphic Streets (GPU), ResNet197 showed remarkable accuracy in plant disease recognition in a series of tests. This indicated that it surpassed both the latest transmission training methods currently in use and other ResNet models.

In another study conducted by (Singh, et al., 2022), the researchers focused primarily on India, which relies primarily on agriculture, with an important part of its population being based on its self-sufficiency. Cotton, the major export of countries around the world, is particularly essential to its economy. Nevertheless, there are several stories, including parasites, bad weather and malnutrition, damage from crops, especially cotton plants. Diagnosis can be difficult because these diseases can have a variety of causes. The purpose of this project was to have an advanced computer method known as detailed learning to promote the diagnosis of cotton plant diseases. This study assembled a series of in-depth data covering 22 diseases affecting cotton sheets. The dataset had been improved to improve model performance. After much experience, they discovered that a network of adhesive nerves (CNNs) produces the best results. When they were tested, their proposed model identified cotton disease with incredible accuracy. This was

handled quickly and achieved better than previous methods. This showed that using those methods in real systems can help farmers diagnose and treat cotton plant problems correctly and quickly. The proposed automated approach using Thoroughbred Networks (CNNs), especially the ResNet-152v2 architecture, is a reliable solution. Formatted with various data on cotton leaf disease, this model demonstrates the impressive accuracy and accuracy of disease detection. The successful application of detailed learning methods, particularly ResNet-152V2, highlights its effectiveness in classifying and diagnosing various diseases in cotton plants.

This automated approach has real-world application outlook and provides a timely and accurate means to solve the problem of cotton crop disease. This work contributed to this field, promotes the creation of automated disease detection systems in agriculture, and highlights the challenges faced in the cotton field. The results open the door to the best ways to deal with diseases that increase cotton growth stability and production.

## **2.2 Related Work**

According to (Satraj, et al., 2024), it was stated that with the advancement of artificial intelligence technologies, it has become possible to develop effective systems for automatic detection and classification of plant diseases with high accuracy, which reduces losses. These systems rely on sensors to capture images of plants and analyze them using machine learning algorithms for early detection of diseases in crops such as tomatoes, peppers, potatoes, and cucumbers .

On the one hand, (Reddy, et al., 2023) stated that there are several AI-based techniques for early detection of plant pests, such as image processing and computer vision techniques that use advanced algorithms to analyze images and extract information about the condition of the plant. Specialized cameras are used to enhance images and capture changes in color and texture,

which helps in detecting pests. Computer vision contributes to detecting diseases more quickly and accurately than traditional methods, while machine learning is used to distinguish between disease and healthy symptoms. This technique is also effective in detecting fungal diseases such as Fusarium, which causes wilting of plants. Deep learning is also used to analyze spectral or regular images to detect diseases such as spotting or poisoning, allowing diseases to be accurately classified even in their early stages.

On the other hand, (Sushreeta, et al., 2021) clarified that Artificial intelligence is characterized by its ability to achieve accuracy and speed in detecting plant diseases, as deep learning algorithms rely on analyzing agricultural images to quickly identify diseases, saving valuable time for farmers. It also helps reduce costs by reducing reliance on human experts. It provides smart management solutions, which helps farmers make quick and informed decisions based on accurate analysis, which enhances disease control strategies and ensures crop sustainability. The combination of artificial intelligence and agricultural technologies can improve crop productivity, as diseases can be detected early and losses can be reduced, as well as quality and yield increases, which supports food security.

### **2.3 Related Work Summary**

Traditional cotton disease diagnosis is challenged by limitations in accuracy and time-efficiency. Deep Learning (DL), particularly Convolutional Neural Networks (CNNs), has emerged as a promising solution for efficient and rapid image-based plant disease detection, supported by recent studies and available datasets. However, these datasets remain limited, especially in covering the diversity of cotton and its local conditions. This research aims to develop an effective DL-based diagnostic system tailored to meet specific agricultural needs and address current research gaps in this domain.

## **2.3 Challenges and Opportunities**

- **Challenges**

1. The lack of studies regarding the research topic.
2. The complexity of the research methodology.
3. Lack of Available Data
4. Variability in Agricultural Conditions
5. Difficulty in Classifying Similar Symptoms
6. High Computational Requirements
7. Challenges in Practical Adoption

- **Opportunities**

1. Getting the best use of AI technologies
2. Adding new data with new topics in the scientific field.
3. Improving Model Performance with Advanced Techniques
4. Creating Larger Databases Through Research Collaboration
5. Integrating Deep Learning with IoT for Smart Farming
6. Supporting Farmers with Smart and Accessible Systems

## **Chapter (3)**

### **Proposed Approach**

## Chapter (3 )

In this chapter, the researchers clarify the research model, its approach, and the data set.

### 3.1 Research Methodology

In this research, we will use a cotton disease detection using several fine-tuned deep-learning models. The performance of the model will be assessed through several metrics like accuracy and recall (Islam, et al., 2023).

The project aims to detect cotton diseases from images using deep learning.

- First: Download the data from the Kaggle website:

<https://www.kaggle.com/datasets/dhamur/cotton-plant-disease>

- Second: Download Anaconda, install Jupyter, create the model, and download the required libraries: pip install TensorFlow NumPy matplotlib pillow scikit-learn seaborn Kaggle pandas keras

Table (3.1): The goal of each library

| Library    | Core function                                      | Why needed in the project                                           |
|------------|----------------------------------------------------|---------------------------------------------------------------------|
| TensorFlow | Deep learning framework for building/training CNNs | Define the architecture, compile/fit, save models, make predictions |
| Keras      | High-level API on top of TensorFlow                | Faster layer/model construction and convenient callbacks            |
| NumPy      | Efficient array and numerical operations           | Handle tensors/arrays and pre/post-processing computations          |

| <b>Library</b>      | <b>Core function</b>                                  | <b>Why needed in the project</b>                                               |
|---------------------|-------------------------------------------------------|--------------------------------------------------------------------------------|
| <b>Pandas</b>       | <b>Tabular data handling</b>                          | <b>Summarize results, comparison tables, read metadata files</b>               |
| <b>Matplotlib</b>   | <b>Low-level plotting</b>                             | <b>Training curves (accuracy/loss) and custom visualizations</b>               |
| <b>Seaborn</b>      | <b>Statistical visualization on top of Matplotlib</b> | <b>Clean heatmaps (confusion matrix) and quick analytics plots</b>             |
| <b>scikit-learn</b> | <b>ML utilities and metrics</b>                       | <b>Compute Accuracy/Precision/Recall/F1/ROC-AUC; confusion matrix</b>          |
| <b>Pillow (PIL)</b> | <b>Image I/O and basic processing</b>                 | <b>Load/resize/convert images for single-image inference and checks</b>        |
| <b>Kaggle</b>       | <b>Kaggle API client</b>                              | <b>Download the Cotton Plant Disease dataset directly into the environment</b> |

## **Data Set & Environment**

The model was developed using the **Cotton Plant Disease** image dataset from Kaggle: <https://www.kaggle.com/datasets/dhamur/cotton-plant-disease>. Experiments were conducted in an **Anaconda** environment using **Jupyter Notebook**. After installing the required libraries, the CNN-based models were implemented and evaluated. `pip install TensorFlow NumPy matplotlib pillow scikit-learn seaborn Kaggle pandas keras`

The model was designed in two sequential phases. The first is binary classification, which detects whether a leaf is infected or healthy. A simple model is trained to decide infection status, and the dataset for this step is arranged into two folders only: Healthy (images of healthy leaves) and Diseased (images of all infected leaves, regardless of type). The second phase is multi-class classification, which—if infection is present—identifies the specific disease type. For this phase, the data are manually organized into separate folders for each class as follows: Healthy (healthy leaves), Aphid, Fall Armyworm, Bacterial Blight, Powdery Mildew, and Target Spot.

## **Data Preparation**

Manual data preparation steps:

- Create two top-level directories in the project: Binary and Multi.
- Within Binary, two subfolders—Healthy and Diseased—were created.
- Healthy leaf images were added to Binary/Healthy/.
- All disease images (of any type) were added to Binary/Diseased/.
- Within Multi, one folder per disease class was created using the original class names, and each class's images were added to its corresponding folder.

Binary/

Healthy/ Diseased/

Multi/

Healthy/ Aphid/ Armyworm/ Bacterial Blight/ Powdery Mildew/ Target Spot/

## **Data Augmentation & generators**

To improve generalization, moderate augmentations were applied—small rotations, limited shifts and shear, mild zoom, horizontal flips, and brightness jitter—implemented with Keras

`ImageDataGenerator`. Generators produced batches of size 32 at an input resolution of  $224 \times 224 \times 3$ . Binary runs used `class\_mode='binary'` with a sigmoid output; multi-class runs used `class\_mode='categorical'` with a softmax output. Normalization was performed via `rescale=1./255` or a backbone-specific `preprocess\_input`.

## **Model construction (transfer learning)**

ImageNet-pretrained backbones ( MobileNetV2, ResNet50, EfficientNetB0, VGG16, VGG19, DenseNet121 )

## **Training control**

The model was trained using the **Adam** optimizer with **categorical cross-entropy**; training control employed **EarlyStopping** (monitoring val\_loss with best-weight restoration) to prevent overfitting, **ReduceLROnPlateau** to lower the learning rate on plateaus, and **ModelCheckpoint** to persist the best weights.

## **Visualization and evaluation**

Results were summarized with training/validation accuracy and loss curves, the classification report, confusion-matrix heatmaps, and ROC–AUC (binary directly and one-vs-rest for multi-class when feasible). Performance across the six backbones was compared to select a suitable accuracy–compute trade-off for the application.

## **Binary Classification**

The objective of binary classification is to develop a model capable of distinguishing between healthy and diseased cotton plant leaves.

Initially, images were collected and organized into separate folders according to their category: healthy leaves and diseased leaves. The dataset was then divided into training, validation, and testing sets—most images were allocated for training, a smaller portion for validation to tune the model and enable early stopping, and a held-out portion for final evaluation after training.

After preparing the dataset, a **Convolutional Neural Network (CNN)** model was designed via transfer learning. An ImageNet-pretrained backbone was used as a frozen feature extractor (with `include_top=False`), followed by a lightweight classification head. The architecture consisted of the following components:

- **Pretrained Backbone (frozen initially):** Acts as a general feature extractor learned from large-scale natural images.
- **GlobalAveragePooling2D:** Aggregates spatial features into a compact vector without adding many parameters.
- **Dense(256, ReLU):** Learns non-linear combinations of the extracted features.
- **BatchNormalization:** Accelerates training and stabilizes the distribution of activations.
- **Dropout(0.5):** Mitigates overfitting by randomly deactivating units during training.
- **Dense Output Layer with Sigmoid Activation:** Generates the probability that the input image belongs to the “Diseased” class.

### **Detailed project implementation steps:**

#### **1. Understand the Data Type**

The data consists of color images of cotton leaves saved in JPG/PNG format. They were organized into a main folder named binary, which contains two subfolders:

- healthy: Contains images of healthy leaves

- diseased: Contains images of infected leaves

The purpose of this data is to classify each image into one of two categories: "healthy" or "infected."

## **2. Define the Problem Type**

The problem type is binary classification. The model to be built must predict whether an image represents a "healthy" or "infected" cotton leaf.

## **3. Data Preparation**

Python libraries such as os and shutil were used to format the files and folders. We used the train\_test\_split function from the scikit-learn library to split the data 80% for training and 20% for testing. The images were moved to new, organized folders:

- binary\_dataset/train/
- binary\_dataset/test/

## **4. Data Cleaning**

We ensured that each image was correctly moved to its appropriate folder without errors in names or paths. No corrupted images or missing data were detected, indicating that the data was clean from the source and did not require additional cleaning steps.

## **5. Data Augmentation**

We used the ImageDataGenerator tool from the Keras library to generate new sample images during training. Random transformations such as rotation, zoom, cropping, and displacement

were applied. This data diversification aims to improve the model's generalization ability and reduce the possibility of overfitting.

## **6. Creating Data Generators**

We created data generators for training and testing using ImageDataGenerator to facilitate batch loading of images. The images were resized to (224\*224) pixels, and normalization was applied with rescale=1./255.

## **7. Build a CNN Model**

The model consists of Several Convolutional (Conv2D) layers followed by MaxPooling

Dropout layers to reduce overfitting. Flatten layer for dimensional transformation, and Final Dense layer for image classification were identified.

We used the Adam algorithm as an optimization algorithm and a binary\_crossentropy loss function, with performance evaluated using accuracy.

## **8. Train the Model**

We trained the model using model.fit with the training and validation data. The number of epochs was determined, and performance was monitored during each training cycle using cross-validation. 10. Visualize Results. After training, we plotted curves showing the model's accuracy and loss over the epochs. These graphs helped us analyze performance and determine whether there was overfitting or a consistent improvement. Similarly, with multiple variables.

## **Research Plan**

- **Data Collection:** dataset for Cotton Disease through Image Analysis Using Deep learning, providing a range of data types and information to advance research in these neurodegenerative conditions. Those datasets are available online which makes it easy to use . The first step of our proposal is the collection of data that is necessary to train any deep learning model. In this research, we collect images from well-known data sources.
- **Data Preprocessing:** normalization and dimensionality reduction, are used to prepare the data for classification algorithms, ensuring it is in a suitable format for analysis. This step includes many tasks like image rotating, sharpening, and augmentation. This is necessary to train any deep learning model for classification tasks.
- **Feature Extraction:** extracting relevant features from data sources like AI imaging, and clinical records .
- **Validation and Testing:** The trained models are validated and tested using separate datasets to assess their accuracy and generalization capabilities. This step ensures the model's reliability in early detection.
- **Predictive Biomarkers:** Classification models can identify predictive biomarkers or patterns in the data that are indicative of cotton diseases. These biomarkers aid in the early identification of plants at risk.
- **Continuous Monitoring:** Classification techniques can be applied to monitor changes in cotton lants, allowing for continuous assessment and early intervention as necessary.

### 3.2 Dataset

Table (3.2) The research dataset

| <b>Limitations</b>                                                                                                                  | <b>Size &amp; Samples</b> | <b>Biomarker</b>                                                                                                                                                  | <b>Link</b>                                                                                                                                                                                                                                                                                                                                                                     | <b>Dataset Name</b>                                                                                                           |
|-------------------------------------------------------------------------------------------------------------------------------------|---------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------|
| <b>Difficulty in image extraction, features extracted as scalograms more accurately detect each plant condition using DL models</b> | 2432 images               | approach combines DL with feature extraction methods such as continuous wavelet transform (CWT) and fast Fourier transform (FFT).                                 | <a href="https://www.researchgate.net/publication/382251164_An_ensemble_deep_learning_models_approach_using_image_analysis_for_cotton_crop_classification_in_AI-enabled_smart_agriculture">https://www.researchgate.net/publication/382251164_An_ensemble_deep_learning_models_approach_using_image_analysis_for_cotton_crop_classification_in_AI-enabled_smart_agriculture</a> | An ensemble deep learning models approach using image analysis for cotton crop classification in AI-enabled smart agriculture |
| <b>Large number of images, and the difficulty in image extraction 2170 image</b>                                                    | 2170 images               | Deep learning method is used to solve the issue because it will perform exceptionally well in image processing and classification problems. The network was built | <a href="https://iarjset.com/papers/cotton-plant-disease-prediction-using-image-processing-and-transfer-learning/">https://iarjset.com/papers/cotton-plant-disease-prediction-using-image-processing-and-transfer-learning/</a>                                                                                                                                                 | Cotton plant disease-prediction using Image processing and Transfer learning                                                  |

|                                                                                  |             |                                                                                                                                                                                      |                                                                                                                                                                                                                               |                                                 |
|----------------------------------------------------------------------------------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------|
|                                                                                  |             | using a combination of the benefits of both the ResNet pre-trained on ImageNet and the Xception component, and this technique outperforms other state-of-the-art techniques          |                                                                                                                                                                                                                               |                                                 |
| <b>Large number of images, and the difficulty in image extraction 1124 image</b> | 1124 images | Using a digital camera, I took images of a cotton field for this assignment. In the preprocessing stage, the backdrop is removed from the image using a background removal technique | <a href="https://www.researchgate.net/publication/378156934_Cotton_Leaf_Disease_Detection_Using_Image_Processing">https://www.researchgate.net/publication/378156934_Cotton_Leaf_Disease_Detection_Using_Image_Processing</a> | Cotton Disease Detection Using Image Processing |

### 3.3 The research model

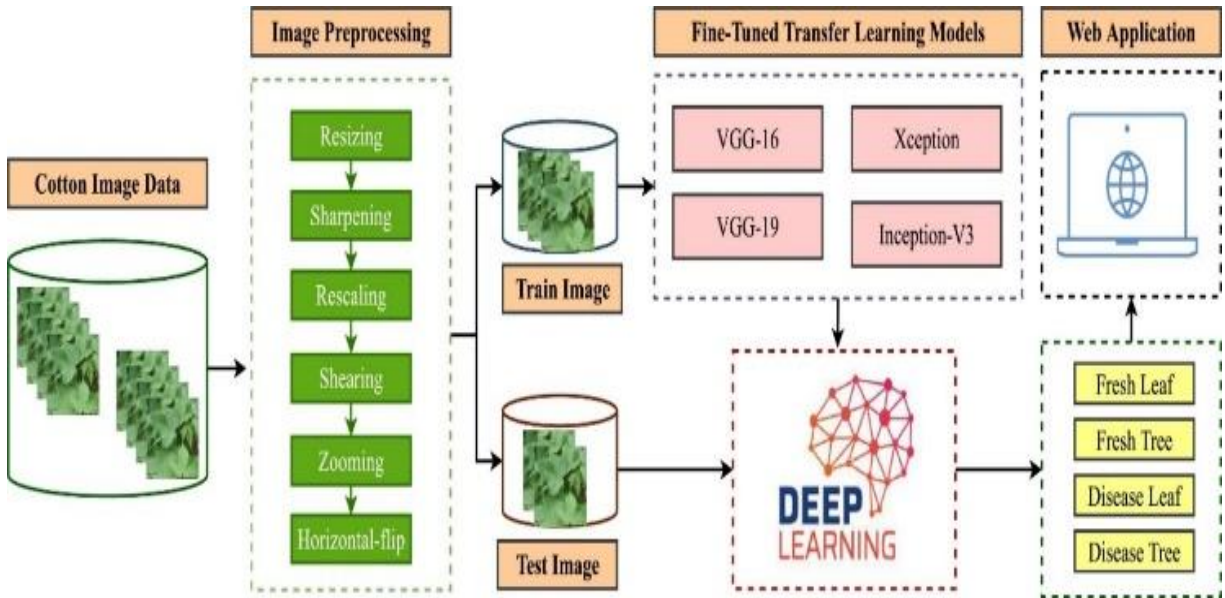


Figure (3.1) The research Model

### Convolutional Neural Networks (CNN)

A neural network designed to process images and data with a spatial structure. It consists of layers that apply filters to extract visual features from images, such as edges and patterns.

- **Basic Components:**

Convolution (Conv2D) layers: To extract features from images, such as edges and textures.

MaxPooling layers: To reduce the image dimensions and preserve the most important features.

Flatten layer: To convert the output of previous layers into a single vector.

Dense (Fully Connected) layer: Used for final decision-making and classification. We used it Because CNNs are currently considered the best at handling images, reducing the need for manual feature extraction.

## **1. ImageDataGenerator**

A tool from the Keras library used to generate training data with dynamic data augmentation. Automatically reads images from folders. Random transformations (rotation, shift, zoom, etc.) are applied. Rescaling is applied so that pixel values are between 0 and 1. We used it to expand the dataset size and improve the model's generalization ability, especially if the original data is limited.

## **2. EarlyStopping**

Callback is used during training to monitor model performance and automatically stops the training process if no improvement occurs after a specified number of epochs.

- Advantages:
  - Protects against overfitting.
  - Reduces unnecessary training time.

We used it to prevent the model from continuing to learn after it has stopped improving, and to maintain optimal performance on validation data.

## **3. Adam Optimizer**

A modern optimization algorithm based on inverse gradients, combining the Momentum and RMSprop algorithms.

- Advantages:

Fast responsiveness.

Does not require significant parameter tuning. It works well in most deep learning applications.

we use it to improve the speed and accuracy of model training, compared to traditional algorithms like SGD.

#### **4. Binary Crossentropy Loss**

A loss function used in binary classification problems to measure the difference between the actual and correct prediction. Log Loss or Binary Cross-Entropy Loss. The purpose of the function: Penalizes the model if it predicts probabilities that are far from the truth (e.g., truth 1, prediction 0.1 => high loss). It encourages the model to produce accurate probabilities that are close to the truth.

We use it because the project is classified into only two classes (healthy or diseased leaves), this function is best suited for this type of task.

#### **5. Accuracy Metric**

The primary performance metric used to measure the number of images correctly classified as a percentage of the total. We use it because it's a straightforward way to measure how well a model classifies images correctly.

#### **6. Matplotlib**

A Python graphing library used to plot curves and graphs. We use it to plot training curves (loss, accuracy) and visually evaluate model performance.

#### **7. Train\_test\_split (from sklearn)**

A function used to split data into two sets: training and testing. We use it to ensure that the model learns from one part of the data and is tested on another part it hasn't seen before, which enhances

fair evaluation. After providing the data, the data was trained on six image types: black and white, gray, and RGB. Each type of data was trained on a set of models.

**The RGB models were trained on the following models:**

Binary - Multiple - MobileNetV2 - ResNet50 - EfficientNetB0 - VGG16 - VGG19 - DenseNet121

## **Chapter (4)**

### **Experimental Results and Discussion**

## Chapter (4)

### 4.1 Results of the research

This chapter identifies the main results of the research, which can be stated as follows:

#### **Convolutional Neural Networks (CNNs).**

In this project, we employ CNNs, a specialized class of deep neural networks designed for image data. CNNs use small convolutional filters that slide over the image to learn local features (edges, spots, textures).

#### **Deep Learning (DL).**

CNNs belong to Deep Learning, a branch of machine learning based on multi-layer neural networks that learn features automatically from raw data, rather than relying on hand-crafted descriptors. This capability makes DL particularly effective for computer vision tasks.

#### **Machine Learning (ML).**

DL is a subfield of Machine Learning, which develops algorithms that learn patterns from data to make predictions or decisions without hard-coded rules.

Hierarchy:  $ML \rightarrow DL \rightarrow CNN$

A deep-learning, convolutional neural network (DL-CNN) approach is appropriate for this project because the data are images; CNNs learn discriminative features automatically from raw pixels, handle multi-class classification effectively, and—through transfer learning from ImageNet-pretrained backbones—achieve higher accuracy and greater training efficiency, especially with limited datasets. recognition.

Classification Report:

|              | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| diseased     | 0.98      | 0.95   | 0.97     | 599     |
| healthy      | 0.78      | 0.93   | 0.84     | 120     |
| accuracy     |           |        | 0.94     | 719     |
| macro avg    | 0.88      | 0.94   | 0.90     | 719     |
| weighted avg | 0.95      | 0.94   | 0.94     | 719     |

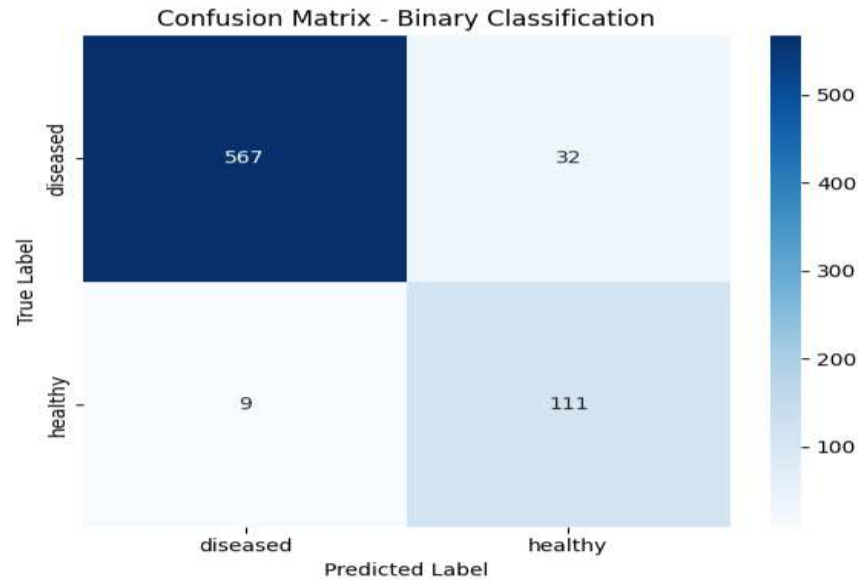


Figure (4.1) confusion matrix – Binary classification

After training the model, the model achieved an overall **accuracy of 94%** and a **Loss: 0.1499**, demonstrating strong performance in distinguishing between healthy and diseased cotton leaves.

**Diseased leaves:** The model reached a precision of 0.98, a recall of 0.95, and an F1-score of 0.97, and **Healthy leaves:** a precision of 0.78, a recall of 0.93, and an F1-score of 0.84,

The Confusion Matrix - Binary Classification these results showing that the model correctly classified 567 out of 599 **diseased** leaf images and 111 out of 120 **healthy** leaf images, with only a small number of misclassifications in both categories.

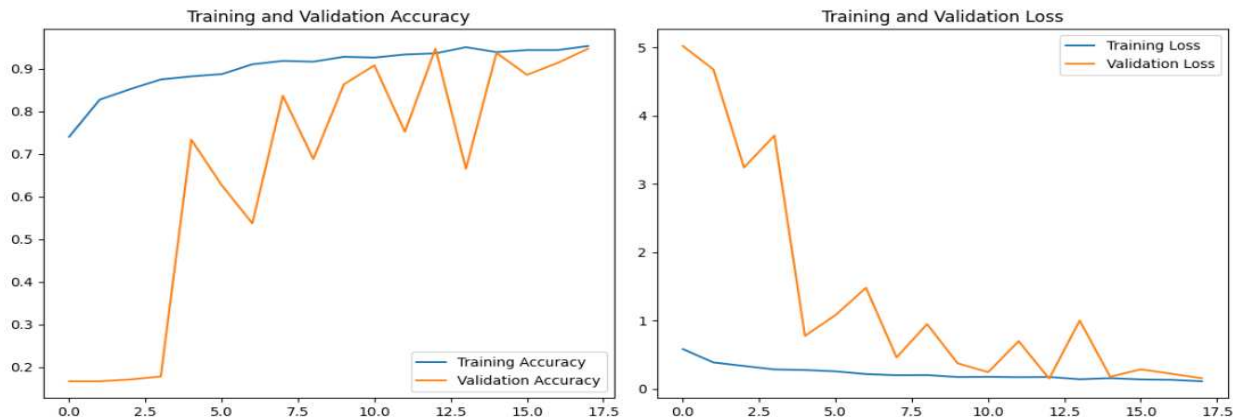


Figure (4.2) training and variation accuracy

The graph shows that the training accuracy began at around **74%** and consistently increased, reaching approximately **94%** by the end of training. The validation accuracy started at about **16%**, then improved steadily with some fluctuations, eventually reaching around **94%** as well. For the loss, the training loss decreased from roughly **0.55** to below **0.1**, while the validation loss started high at approximately **5.0** and dropped sharply, settling around **0.1** with minor variations throughout the process.

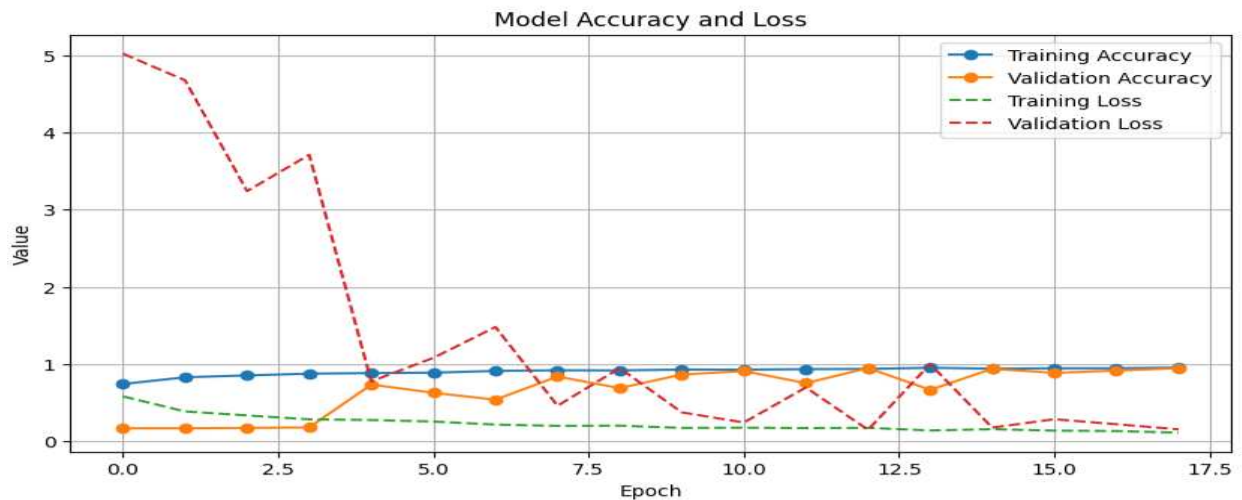


Figure (4.3) Model accuracy and loss

The graph shows that **training accuracy** started at around **76%** and quickly increased to approximately **97–98%**, while **validation accuracy** began at about **20%** and steadily improved, reaching nearly **98%** by the end.

**Training loss** decreased from roughly **0.55** to below **0.1**, whereas **validation loss** started high at around **5.0** and dropped significantly to about **0.1**, with some fluctuations during the process.

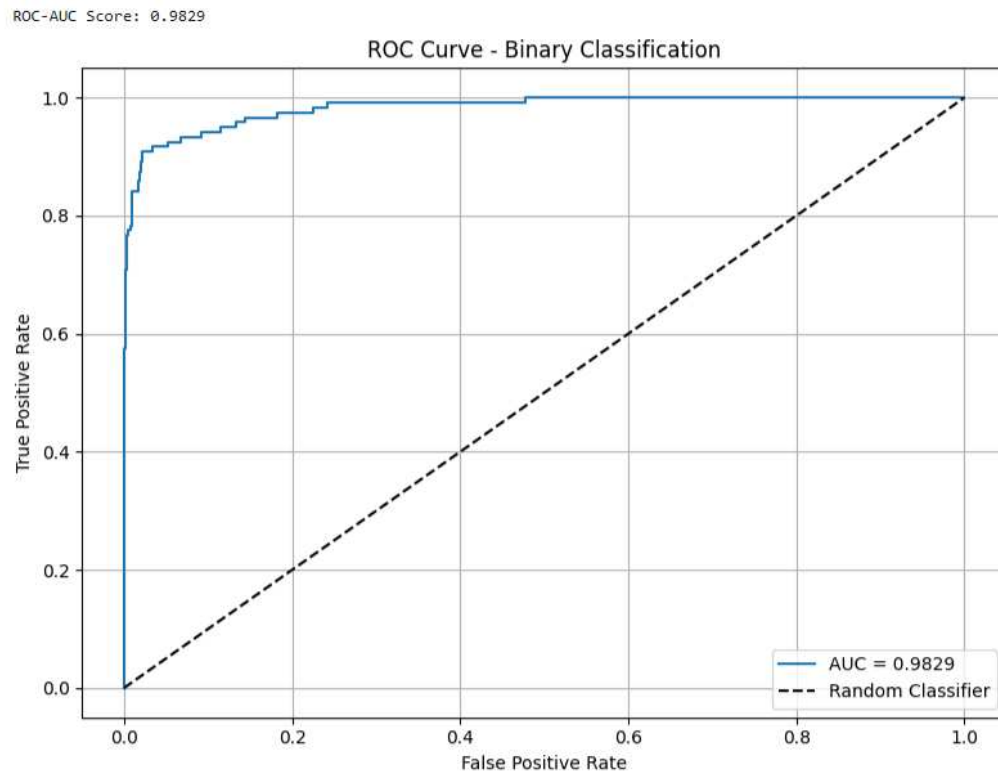


Figure (4.4) ROC curve- Binary classification

Finally, the model's **ROC-AUC** score was 0.9829, underscoring its excellent capability to discriminate between the two classes across varying decision thresholds.

## Multi-Class Classification

The objective of multi-class classification is to develop a model capable of distinguishing among several categories of cotton plant leaves, each corresponding to a specific disease or to a healthy condition. Initially, images were organized into separate folders—one folder per class (e.g., **Healthy, Aphid, Fall Armyworm, Bacterial Blight, Powdery Mildew, Target Spot**).

The dataset was then divided into **training, validation, and testing** sets—most images were used for training, a smaller portion for validation to guide hyperparameter tuning and early stopping, and a held-out portion for final evaluation. After preparing the dataset, a Convolutional Neural Network (CNN) model was designed **via transfer learning**. An ImageNet-pretrained backbone was employed as a frozen feature extractor (`include_top=False`), followed by a classification head tailored to the project classes. The architecture consisted of the following components:

**Pretrained Backbone (frozen initially):** Provides general visual features learned from large-scale natural images.

**GlobalAveragePooling2D:** Condenses spatial feature maps into a compact representation.

**Dense(256, ReLU):** Learns non-linear combinations of the pooled features.

**BatchNormalization:** Stabilizes activations and accelerates convergence.

**Dropout(0.5):** Reduces overfitting by randomly deactivating units during training.

**Dense Output Layer with Softmax Activation:** Produces a probability distribution over all classes, enabling the model to select the most likely category for each input image.

Classification Report:

|                  | precision | recall | f1-score | support |
|------------------|-----------|--------|----------|---------|
| Aphids           | 0.91      | 0.88   | 0.89     | 120     |
| Army worm        | 0.90      | 0.97   | 0.93     | 120     |
| Bacterial blight | 0.92      | 0.85   | 0.88     | 120     |
| Healthy          | 0.93      | 0.95   | 0.94     | 120     |
| Powdery mildew   | 0.99      | 0.97   | 0.98     | 120     |
| Target spot      | 0.88      | 0.89   | 0.88     | 119     |
| accuracy         |           |        | 0.92     | 719     |
| macro avg        | 0.92      | 0.92   | 0.92     | 719     |
| weighted avg     | 0.92      | 0.92   | 0.92     | 719     |

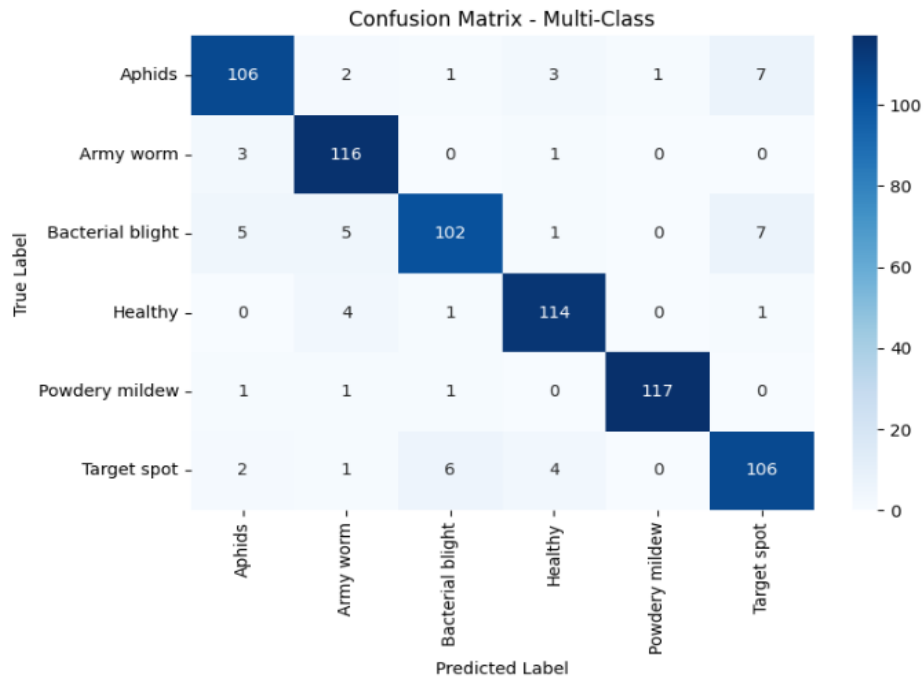


Figure (4.5) Confusion Matrix – Multi class

After training the model, the multi-class classification model achieved an overall accuracy of **92%** and a Loss: 0.2787.

- **Aphids:** Precision of 0.91, Recall of 0.88, and F1-score of 0.89.
- **Army worm:** Precision of 0.90, Recall of 0.97, and F1-score of 0.93.
- **Bacterial blight:** Precision of 0.82, Recall of 0.85, and F1-score of 0.88.
- **Healthy leaves:** Precision of 0.93, Recall of 0.95, and F1-score of 0.94.
- **Powdery mildew:** Precision of 0.99, Recall of 0.97, and F1-score of 0.98.
- **Target spot:** Precision of 0.88, Recall of 0.89, and F1-score of 0.88.

The **Confusion Matrix – Multi-Class Classification** indicates that the model correctly classified the majority of samples in each class. For example, it correctly identified 106 out of 120 *Aphids* images, 116 out of 120 *Army worm* images, 114 out of 120 *Healthy* leaf images, and 117 out of 120 *Powdery mildew* images. Misclassifications were minimal and primarily occurred between classes with visually similar symptoms, such as *Aphids* and *Target spot*, or *Bacterial blight* and *Aphids*.



Figure (4.6) Training and validation accuracy

The graph shows that the training accuracy started at around **48%** and gradually increased to approximately **92%**, while the validation accuracy began at about **18%** and steadily improved, reaching around **91%** by the end of training.

For the loss, the training loss decreased from about **1.6** to below **0.1**, whereas the validation loss started high at approximately **4.6** and dropped significantly to around **0.1**, with some noticeable fluctuations during the early stages of training.

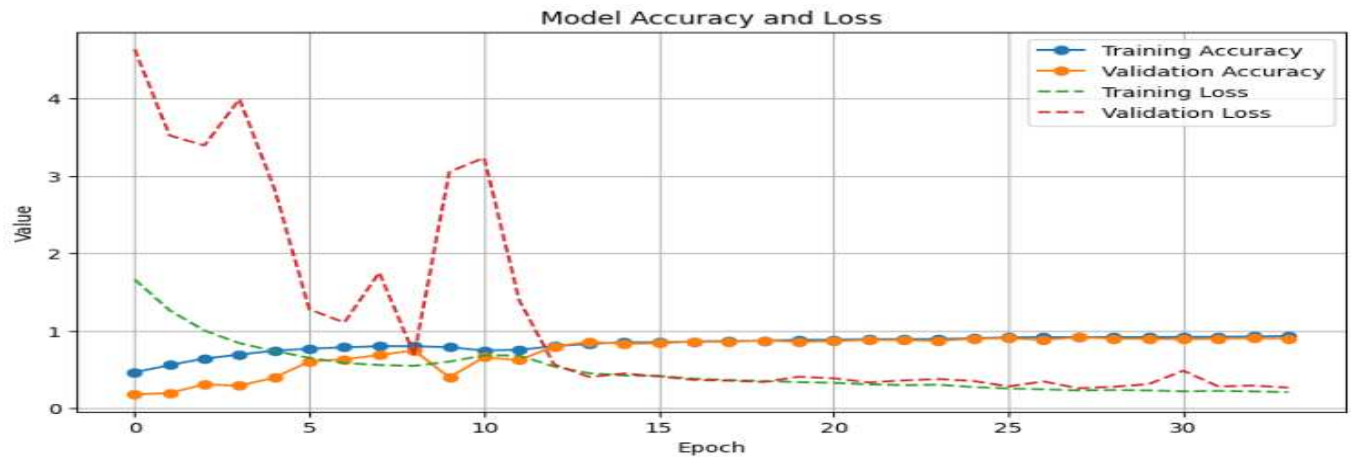


Figure (4.7) model accuracy and loss

The graph indicates that the training accuracy began at approximately **45%** and steadily increased to about **97%**, while the validation accuracy started at around **20%** and showed a consistent upward trend, reaching nearly **97%** by the end of training. The training loss decreased from roughly **1.6** to below **0.2**, whereas the validation loss started high at about **4.5** and dropped significantly to around **0.3**, with some fluctuations observed during the training process.

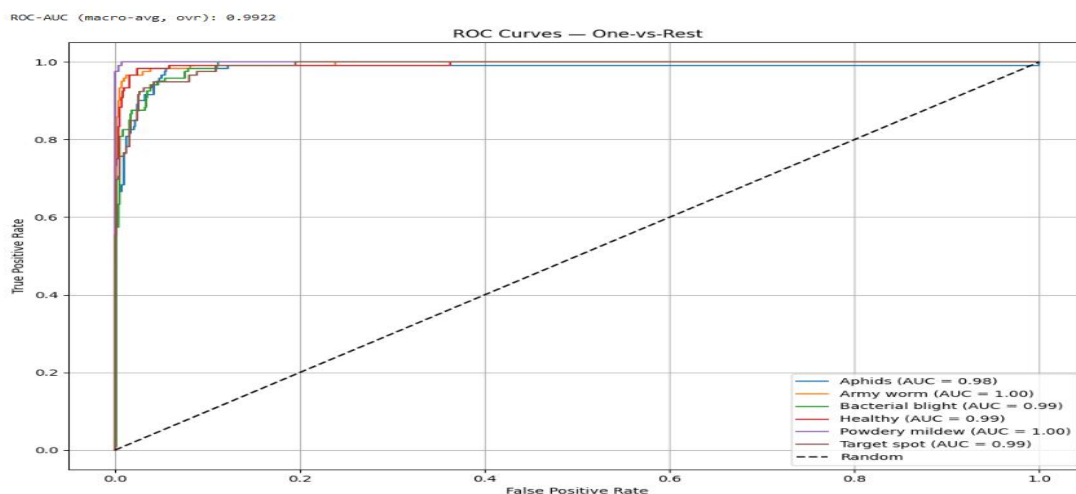


Figure (4.8) ROC curves- One VS- Rest

Finally, the model's **ROC-AUC** score was 0.9922.

After creating the binary and multi-class modules, the following models were trained on them:

**1. MobileNetV2**

Relies on Depthwise Separable Convolutions to reduce computation, with Inverted Residuals and Linear Bottlenecks to preserve information in low-dimensional representations. The objective is to achieve maximum efficiency with competitive accuracy.

**2. ResNet50**

Introduces Residual/Skip Connections to overcome the vanishing-gradient problem, using Bottleneck blocks ( $\text{Conv}1\times1 \rightarrow \text{Conv}3\times3 \rightarrow \text{Conv}1\times1$ ). The nominal depth is 50 layers. The objective is to facilitate training of deep networks and improve accuracy without a blow-up in parameters.

**3. EfficientNetB0**

Applies Compound Scaling (coordinated scaling of depth, width, and input resolution) with MBConv blocks and Squeeze-and-Excitation. The objective is to provide the best accuracy–compute trade-off within the lightweight family.

**4. VGG16**

A simple sequence of  $\text{Conv } 3\times3 + \text{ReLU}$  followed by  $\text{MaxPooling } 2\times2$  in stages; the original fully connected head is removed in transfer learning using `include_top=False`. The objective is to offer a straightforward, easily analyzed baseline.

**5. VGG19**

An extension of VGG16 with greater depth (19 layers in the standard naming). The objective is to leverage additional depth for slightly higher accuracy in some tasks.

**6. DenseNet121**

Employs dense connections, where each layer receives feature maps from all preceding layers

within a Dense Block; includes Bottleneck units and Transition layers with Compression. The objective is efficient feature reuse, fewer parameters, and improved gradient flow.

## MobileNetV2 Model

**Rationale for MobileNetV2:** a lightweight and efficient CNN that reduces computation while maintaining competitive accuracy, well-suited for transfer learning on 224×224 inputs.

### Core Components:

- **Transfer learning:** ImageNet-pretrained backbone with include\_top=False; backbone frozen during warm-up, then fine-tuned in later stages.
- **Depthwise-separable convolutions** to minimize computation and parameters.
- **Inverted residuals with linear bottlenecks** (residual connections) to preserve informative features at low dimensions.
- **Head:** GlobalAveragePooling2D → Dense(256, ReLU) → BatchNormalization → Dropout(0.5) → Dense(output, Softmax) to learn compact, regularized representations and output class probabilities.

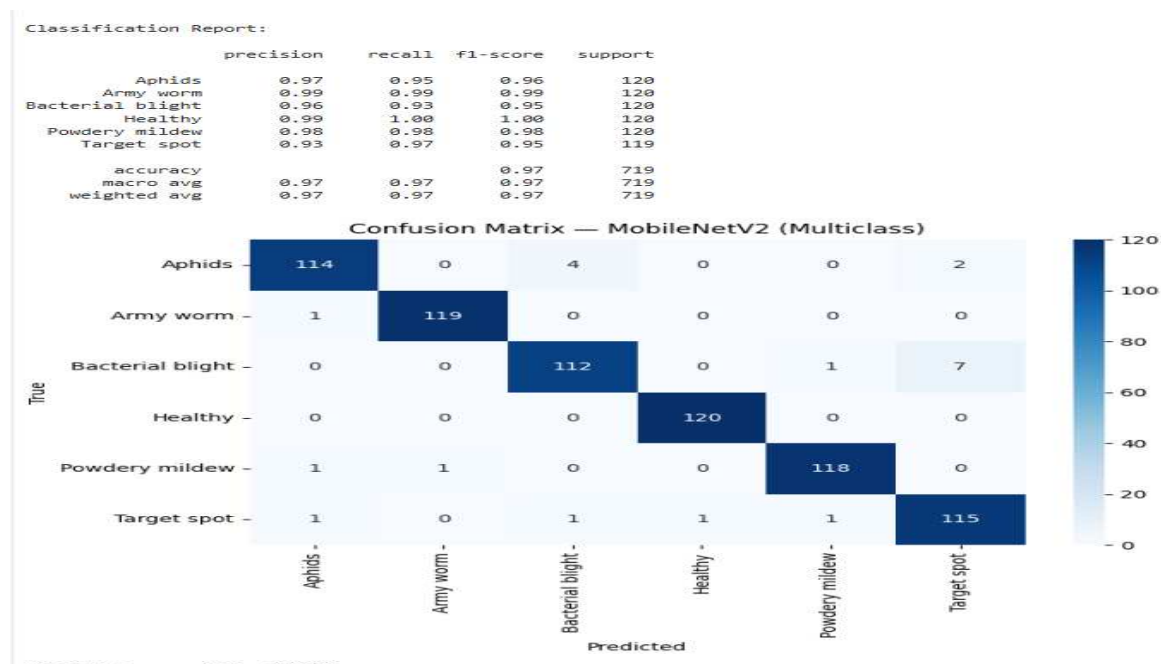


Figure (4.9) Confusion Matrix – mobilenetv2

After training, the multi-class MobileNetV2 model achieved an overall accuracy of 97% and a loss of 0.0892.

- **Aphids:** Precision = 0.97, Recall = 0.95, F1-score = 0.96
- **Army worm:** Precision = 0.99, Recall = 0.99, F1-score = 0.99
- **Bacterial blight:** Precision = 0.96, Recall = 0.93, F1-score = 0.95
- **Healthy leaves:** Precision = 0.99, Recall = 1.00, F1-score = 1.00
- **Powdery mildew:** Precision = 0.98, Recall = 0.98, F1-score = 0.98
- **Target spot:** Precision = 0.93, Recall = 0.97, F1-score = 0.95

The Confusion Matrix indicates that the model correctly classified the majority of samples in each class. For example, it correctly identified 114 out of 120 **Aphids** images, 119 out of 120 **Army worm** images, 112 out of 120 **Bacterial blight** images, and all 120 **Healthy** leaf images. Misclassifications were minimal and primarily occurred between classes with similar visual symptoms, such as Aphids and Target spot, or Bacterial blight and Target spot.

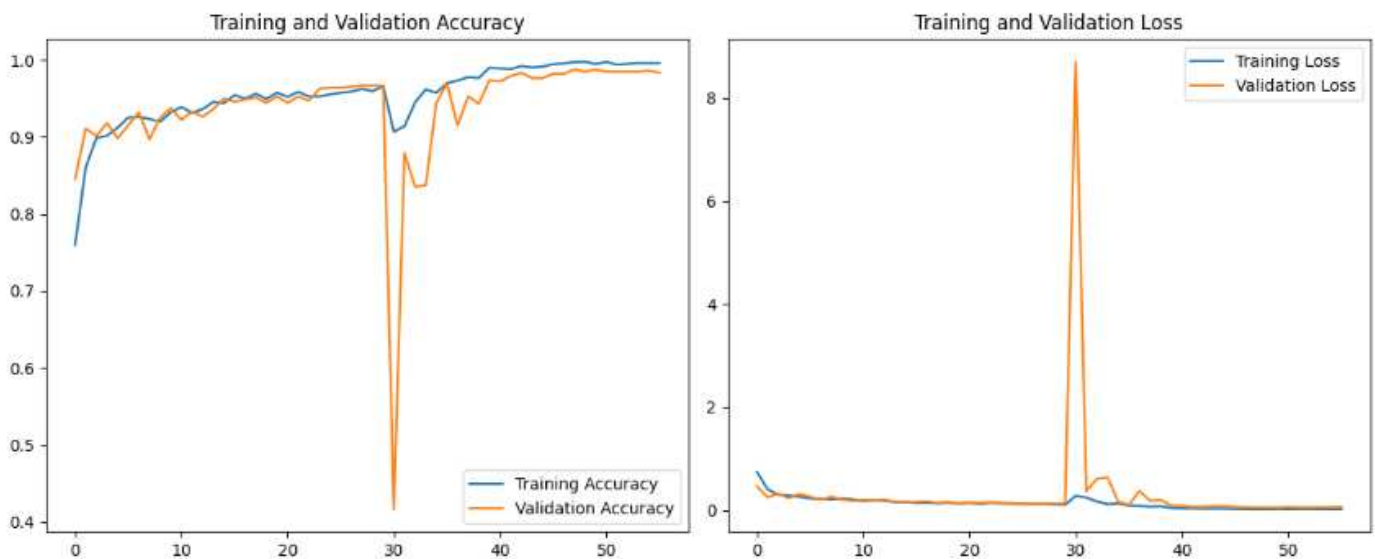


Figure (4.10) training accuracy and loss

The training accuracy started around **78%** and steadily climbed to nearly **99%**, while the validation accuracy began at **88%**, briefly dropped to **42%** around epoch **30**, then recovered to match the training accuracy.

Training loss decreased from **0.9** to almost **0.0**, with validation loss following a similar path but spiking to **8.7** at the same point as the accuracy drop, before quickly returning to near **0.0**.

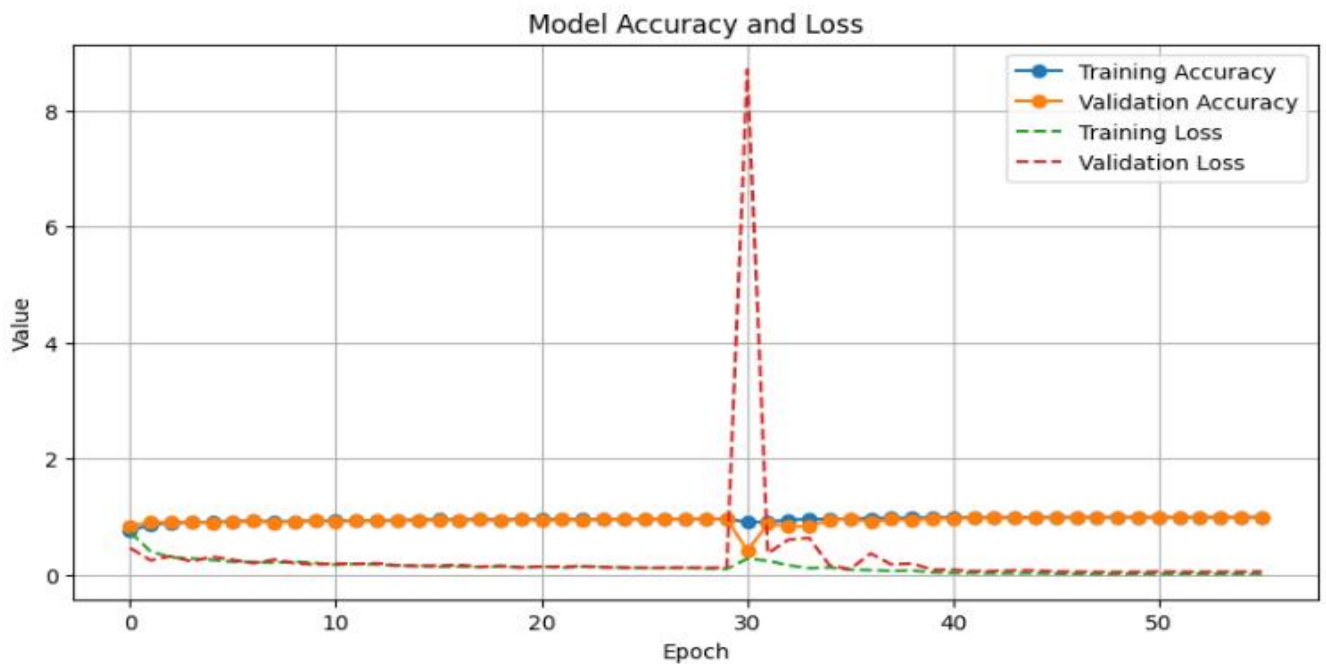


Figure (4.11) model accuracy – loss

The graph shows that training accuracy started at approximately **45%** and steadily increased to about **97%** by the end of training. Validation accuracy began much lower, around **20%**, but showed a consistent upward trend, reaching nearly **97%** as well.

In terms of loss, training loss decreased from roughly **1.6** to below **0.2**, while validation loss started high at around **4.5** and dropped significantly to about **0.3**. Some minor fluctuations were observed in the validation loss during the training process.

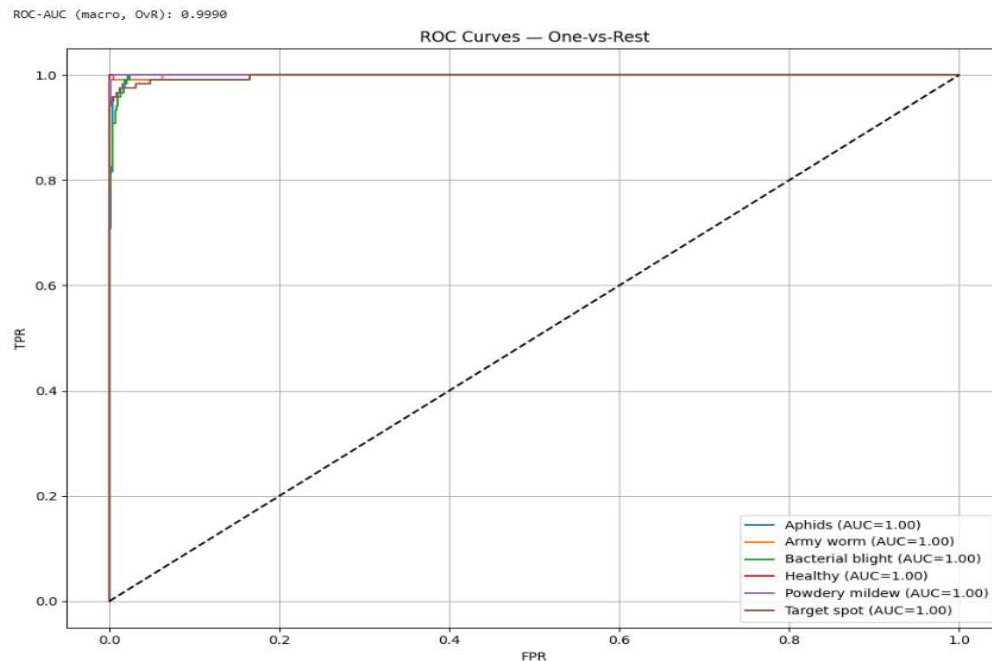


Figure (4.12) ROC curves- One VS- Rest

Finally, the model's **ROC-AUC** score was 0.9990.

## ResNet50 Model

**Rationale for ResNet50:** a deep CNN leveraging residual learning to mitigate vanishing gradients, enabling stable training at depth and strong accuracy in image classification.

### Core Components:

- **Transfer learning:** ImageNet-pretrained backbone with `include_top = False`; backbone frozen during warm-up, then fine-tuned (typically starting from `conv5_x`).
- **Residual bottleneck blocks** ( $1 \times 1 \rightarrow 3 \times 3 \rightarrow 1 \times 1$ ) with **skip connections** for efficient gradient flow and feature preservation.
- **Head:** `GlobalAveragePooling2D`  $\rightarrow$  `Dense(256, ReLU)`  $\rightarrow$  `BatchNormalization`  $\rightarrow$  `Dropout(0.5)`  $\rightarrow$  `Dense(output, Softmax)`.

Classification Report:

|                  | precision | recall | f1-score | support |
|------------------|-----------|--------|----------|---------|
| Aphids           | 0.99      | 0.98   | 0.99     | 120     |
| Army worm        | 0.99      | 1.00   | 1.00     | 120     |
| Bacterial blight | 0.97      | 1.00   | 0.98     | 120     |
| Healthy          | 1.00      | 1.00   | 1.00     | 120     |
| Powdery mildew   | 1.00      | 0.99   | 1.00     | 120     |
| Target spot      | 0.98      | 0.96   | 0.97     | 119     |
| accuracy         |           |        | 0.99     | 719     |
| macro avg        | 0.99      | 0.99   | 0.99     | 719     |
| weighted avg     | 0.99      | 0.99   | 0.99     | 719     |

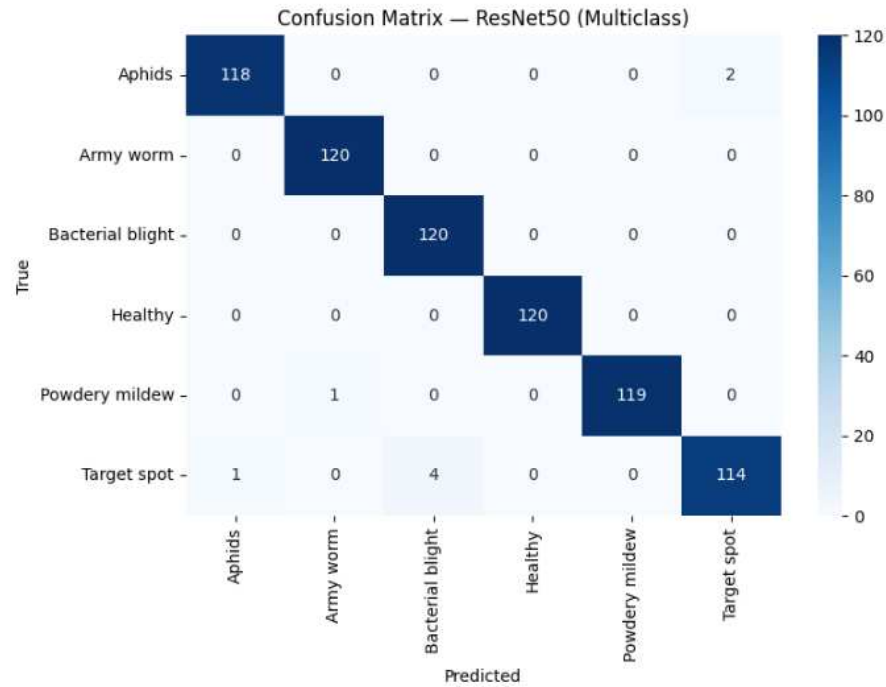


Figure (4.13) Confusion Matrix – ResNet50

After training the ResNet50 multi-class classification model, the overall accuracy achieved was 99%. and a loss: 0.0614

Class-wise Performance:

- **Aphids:** Precision 0.99, Recall 0.98, F1-score 0.99
- **Army worm:** Precision 0.99, Recall 1.00, F1-score 1.00
- **Bacterial blight:** Precision 0.97, Recall 1.00, F1-score 0.98
- **Healthy:** Precision 1.00, Recall 1.00, F1-score 1.00

- **Powdery mildew:** Precision 1.00, Recall 0.99, F1-score 1.00
- **Target spot:** Precision 0.98, Recall 0.96, F1-score 0.97

Confusion Matrix Analysis:

The matrix shows that the model correctly identified nearly all samples in each class.

For example, it classified 118 out of 120 Aphids images, 120 out of 120 Army worm, and 120 out of 120 Healthy leaves correctly. Misclassifications were minimal, occurring mainly between Aphids and Target spot or Target spot and Bacterial blight.

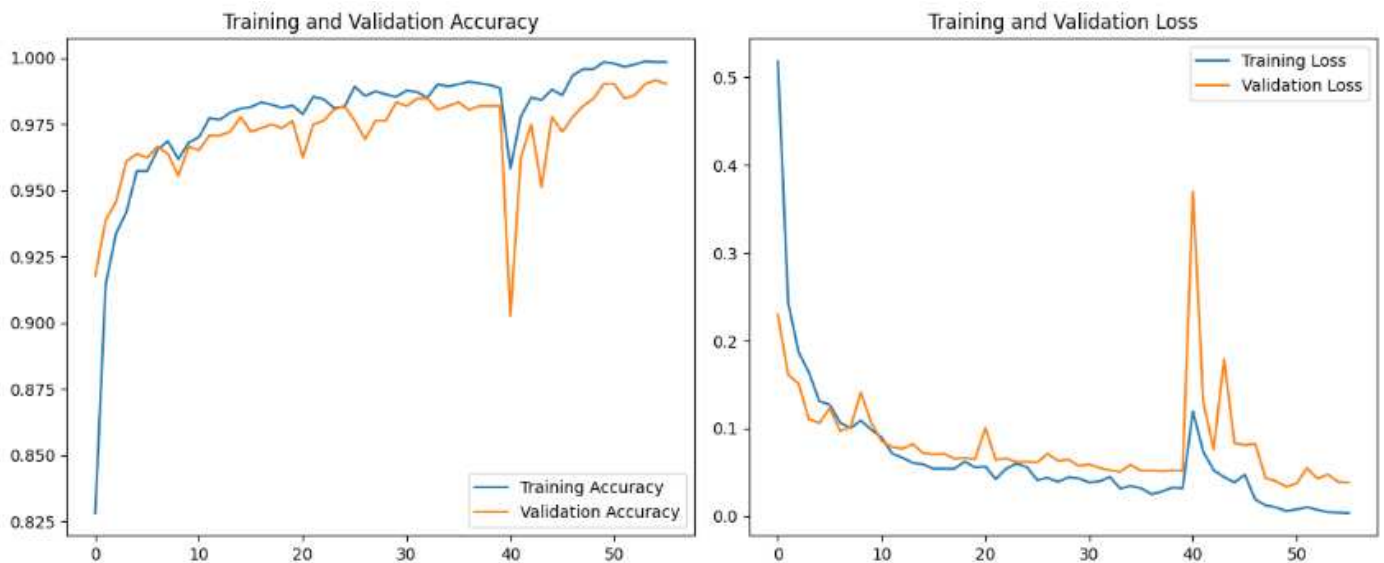


Figure (4.14) Comparison between the training accuracy and loss

The training accuracy began at approximately **83%** and increased steadily to reach nearly **100%**, while the validation accuracy started around **92%**, experienced a noticeable dip to about **90%** near epoch **40**, and then recovered to align closely with the training accuracy by the end.

On the loss side, the training loss dropped sharply from around **0.52** to almost **0.0**, and the validation loss followed a similar downward path, with a sudden spike to roughly **0.37** at the same point as the accuracy drop, before quickly decreasing again toward **0.0**.

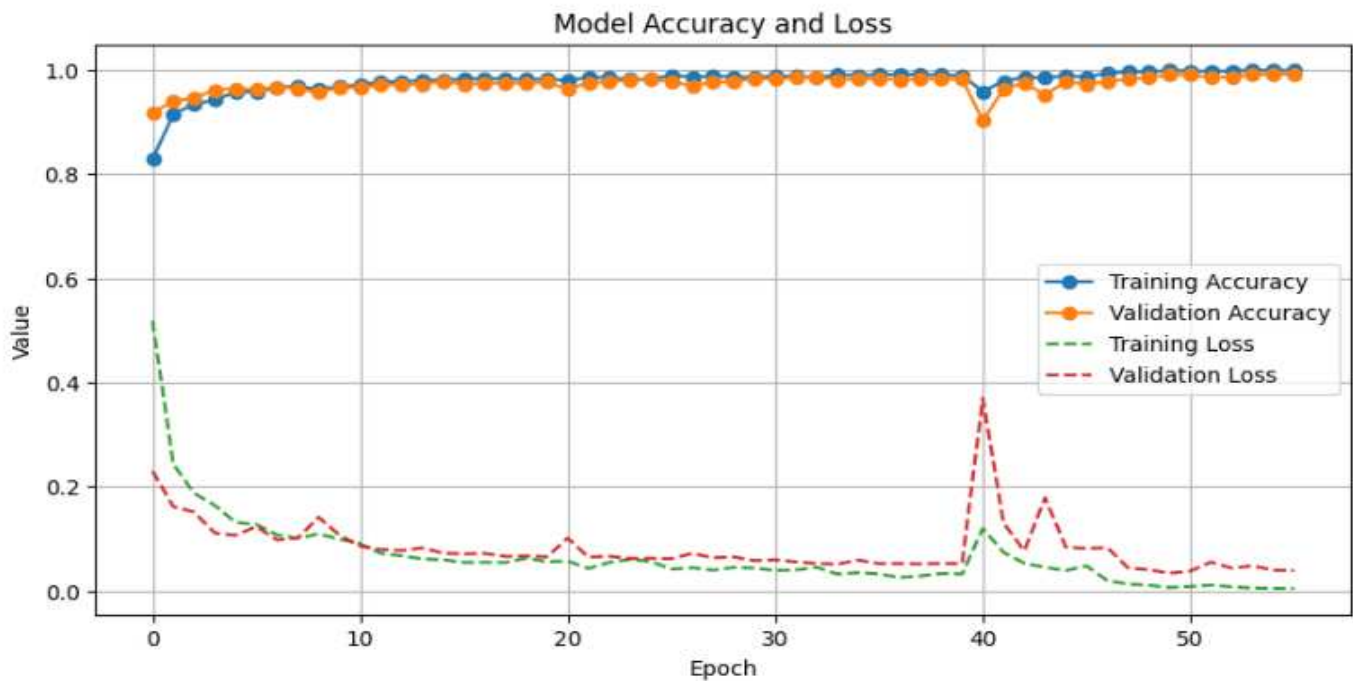


Figure (4.15) model accuracy and loss

The training accuracy began at about **83%** and increased steadily to almost **100%**, while the validation accuracy started around **92%**, stayed close to the training curve, but dropped sharply to about **91%** at epoch **40** before quickly recovering to nearly **100%**.

For the loss, the training loss decreased from roughly **0.52** to nearly **0.0**, and the validation loss followed a similar trend, starting at **0.23**, staying low, but spiking to about **0.38** at epoch **40** — which aligns with the accuracy drop — before quickly returning close to zero.

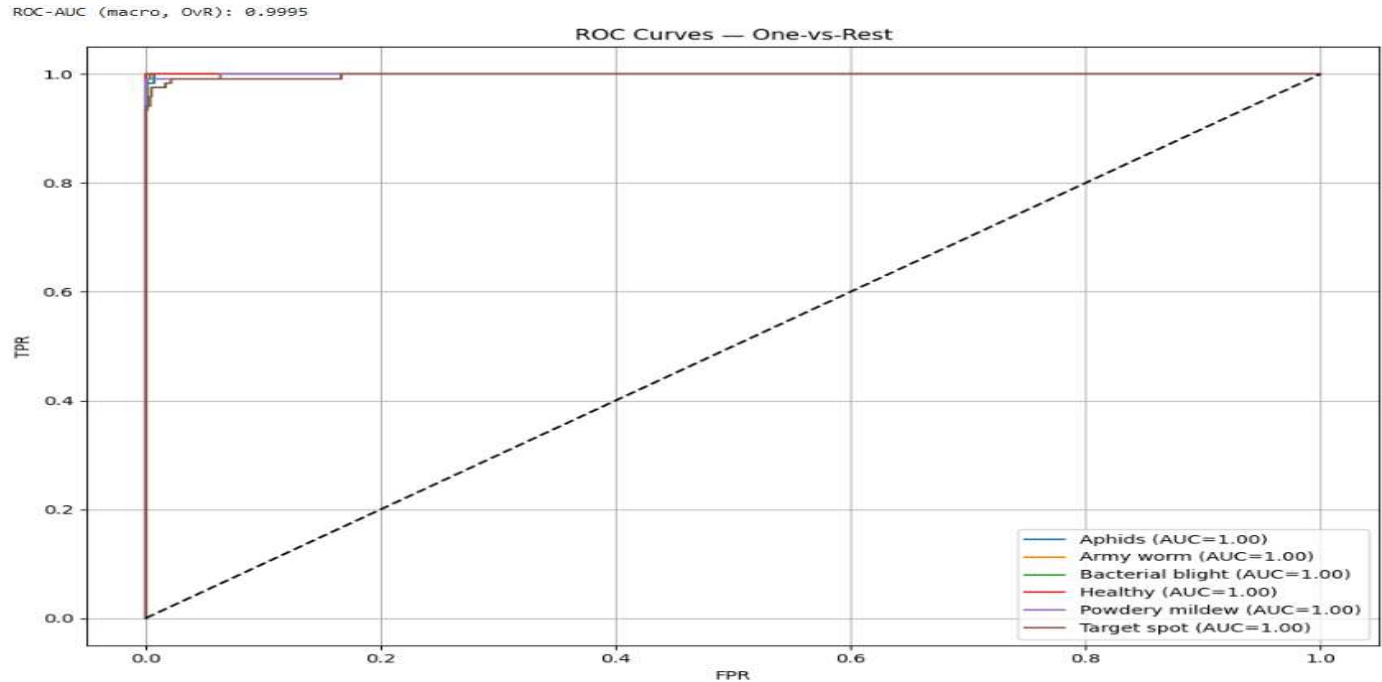


Figure (4.16) ROC curves – One-VS- REST

Finally, the model's **ROC-AUC** score was 0.9995

## EfficientNetB0 Model

**Rationale for EfficientNetB0:** a state-of-the-art CNN designed for an optimal accuracy-to-efficiency trade-off. It uses **compound scaling** to balance depth, width, and resolution, achieving high accuracy with fewer parameters and lower computational cost—well-suited to transfer learning on 224×224 inputs.

### Core Components:

- **Transfer learning:** ImageNet-pretrained backbone with `include_top=False`; backbone **frozen** during warm-up, then **fine-tuned** in later stages.

- **Compound scaling:** coordinated scaling of depth, width, and input resolution for balanced performance.
- **MBConv blocks** (mobile inverted bottlenecks with depthwise convolutions) to minimize computation while preserving representational power.
- **Squeeze-and-Excitation (SE)** to adaptively recalibrate channel importance.
- **Swish** activation for smoother nonlinearity and improved optimization behavior.
- **Head:** GlobalAveragePooling2D → Dense(256, ReLU) → BatchNormalization → Dropout(0.5) → Dense(output, Softmax) to learn compact, regularized representations and output class probabilities for the multi-class setting.

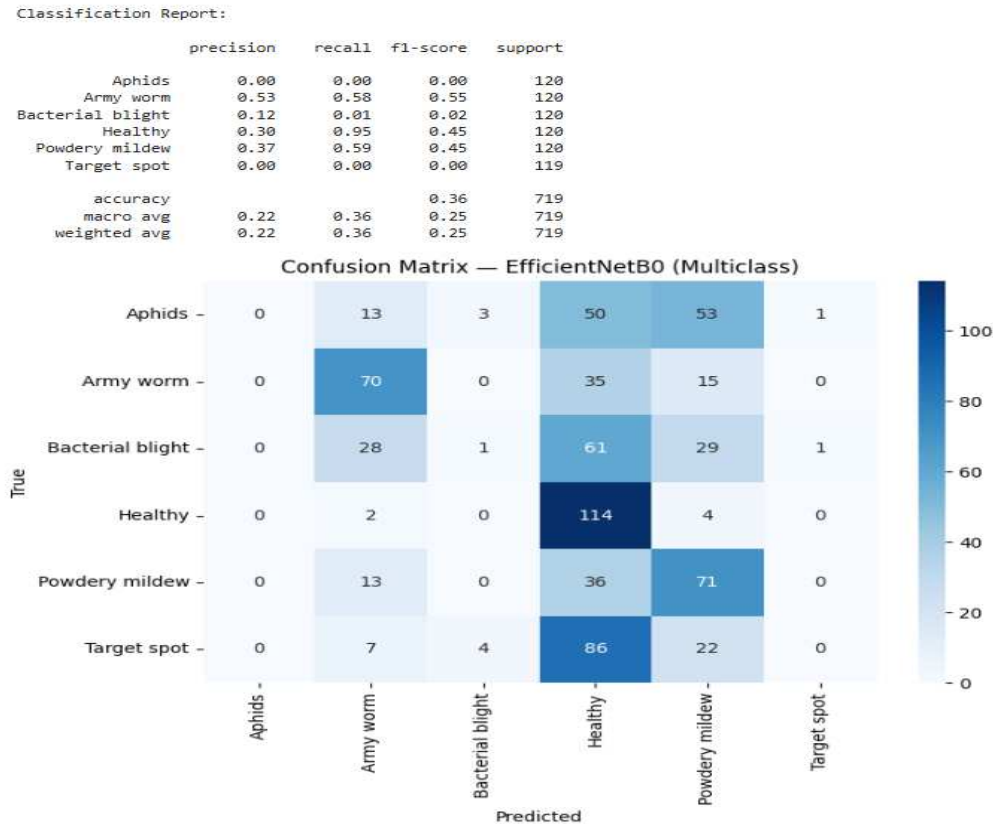


Figure (4.17) Confusion Matrix – EfficientNETB0

After training the EfficientNetB0 multi-class classification model, the overall accuracy achieved was 36% and a loss: 1.5500

Class-wise Performance:

- **Aphids:** Precision 0.00, Recall 0.00, F1-score 0.00
- **Army worm:** Precision 0.53, Recall 0.58, F1-score 0.55
- **Bacterial blight:** Precision 0.12, Recall 0.01, F1-score 0.02
- **Healthy:** Precision 0.30, Recall 0.95, F1-score 0.45
- **Powdery mildew:** Precision 0.37, Recall 0.59, F1-score 0.45
- **Target spot:** Precision 0.00, Recall 0.00, F1-score 0.00

Confusion Matrix Analysis:

The matrix shows that the model struggled to correctly identify most classes, with a strong bias toward predicting the Healthy class.

For example, it classified 114 out of 120 Healthy images correctly, but completely failed to detect Aphids and Target spot. Army worm and Powdery mildew showed moderate detection, but with frequent confusion with Healthy. Bacterial blight was mostly misclassified as Healthy or Army worm.

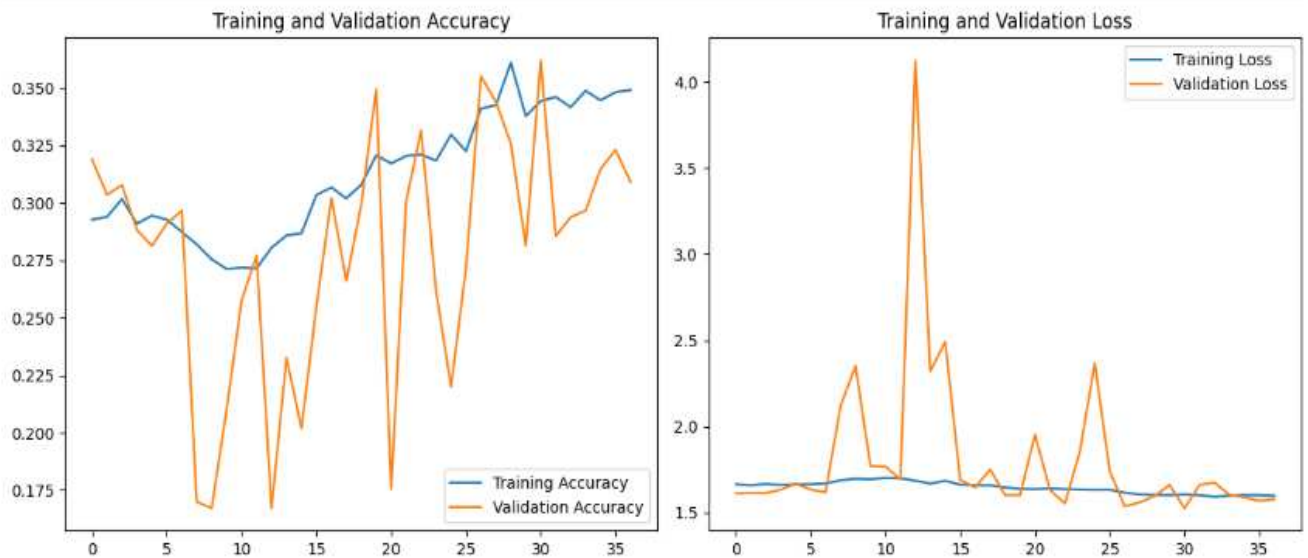


Figure (4.18) comparison between the training accuracy and training loss

The training accuracy started at about **29%** and gradually increased to roughly **35%** by the final epoch.

Validation accuracy began around **31%** but fluctuated heavily, dropping at times to below **18%** and peaking around 35%, indicating instability in generalization.

On the loss side, training loss stayed relatively steady, moving slightly down from about **1.6** to around 1.55.

Validation loss, however, showed significant volatility — starting near **1.63**, spiking multiple times (with a major peak above 4.0 around epoch 12), and then returning to around **1.6** by the end.

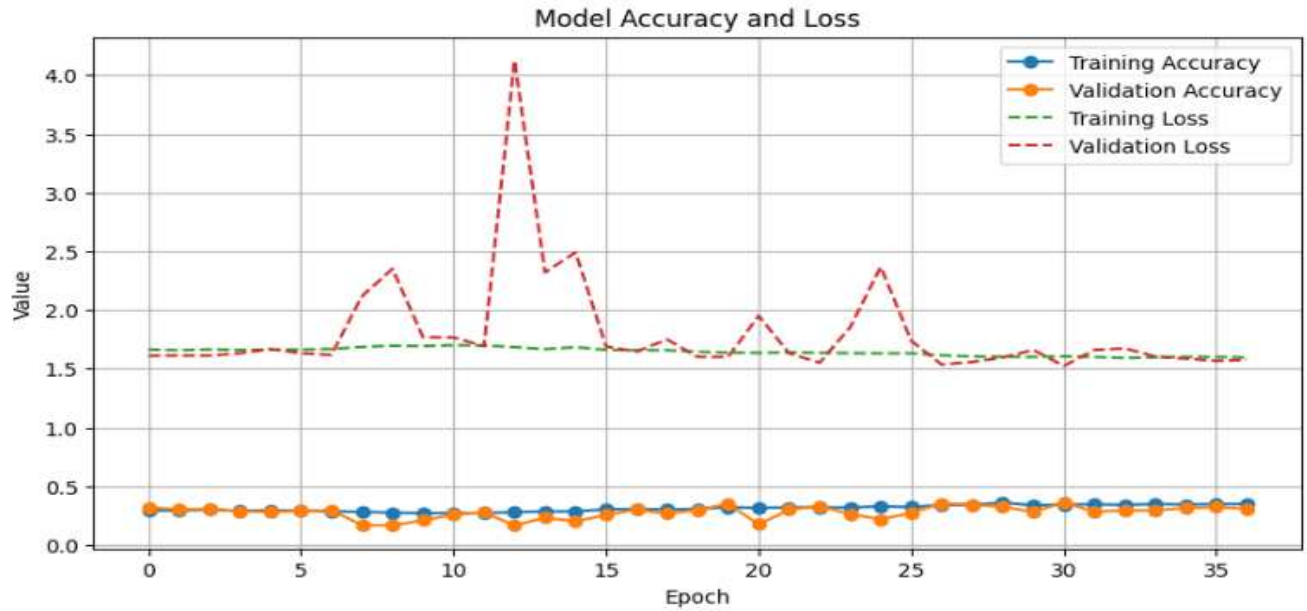


Figure (4.19) Model accuracy and loss

The training accuracy started at about **29–30%** and rose gradually to roughly **35–37%** by the final epochs. Validation accuracy began near **32%**, but fluctuated sharply throughout training—dropping at times to around **16–20%** and peaking near **36%**—indicating unstable generalization.

On the loss side, training loss was fairly flat, moving slightly down from about **1.7 to ~1.6**. Validation loss was highly volatile: it hovered around **1.6–1.7**, but spiked dramatically to **>4.0** around epoch ~12 and again to about **2.3** near epoch ~24 before returning toward **~1.6**.

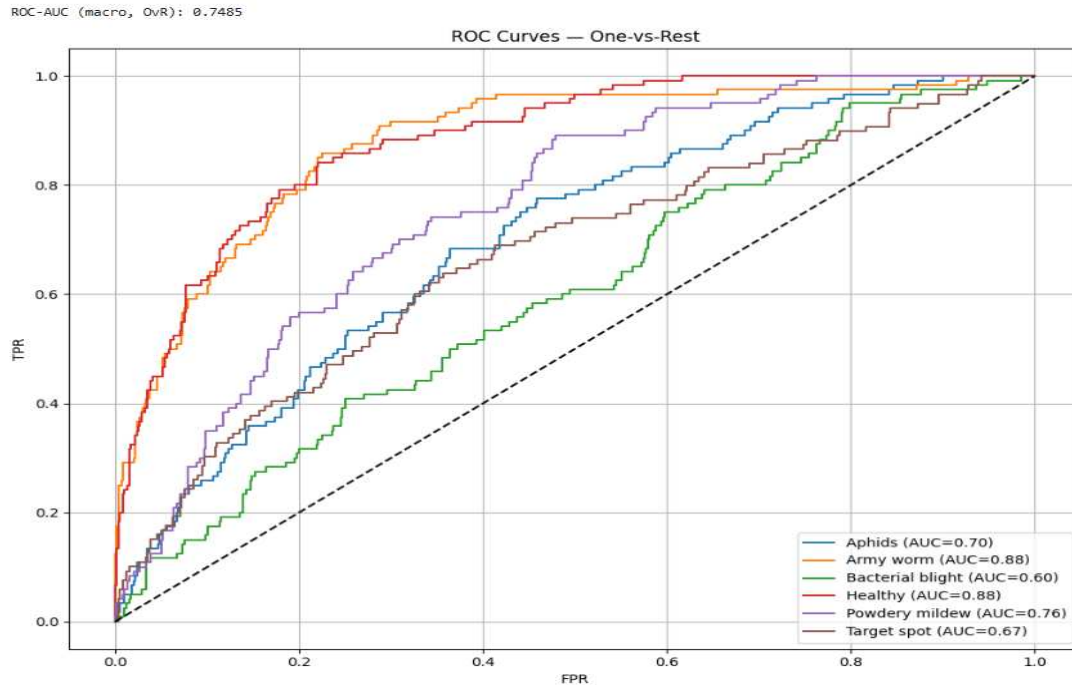


Figure (4.20) ROC curves- One-Vs-Rest

Finally, the model's **ROC-AUC** score was 0.7485

## VGG16 Model

**Rationale for VGG16:** a classic, easy-to-interpret CNN that stacks small  $3 \times 3$  convolutions with periodic max pooling. Its uniform design makes it straightforward to adapt for transfer learning, though it is parameter-heavy compared to newer architectures.

### Core Components:

- **Transfer learning:** ImageNet-pretrained backbone with `include_top = False`; the backbone is **frozen** during warm-up and **fine-tuned** later (typically starting from Block 5).
- **VGG blocks:** stacked **Conv  $3 \times 3$  + ReLU** with **MaxPooling** between blocks provide a strong generic visual feature extractor.

• **Head (replacing the original Flatten/FC):** GlobalAveragePooling2D → Dense(256, ReLU)  
→ BatchNormalization → Dropout(0.5) → Dense(output, Softmax) to reduce parameters,  
improve regularization, and output class probabilities.

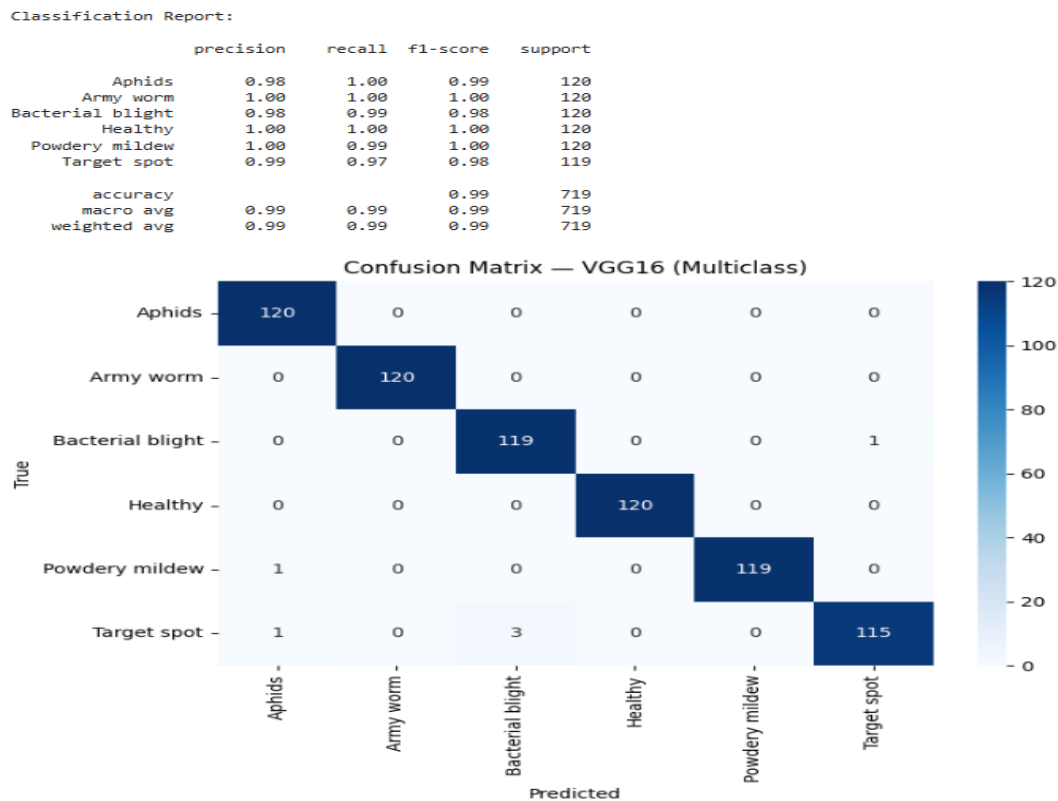


Figure (4.21) Confusion matrix – VGG16

After training the VGG16 multi-class classification model, the overall accuracy achieved was 99.17% and a loss: 0.0854

Class-wise Performance:

- Aphids: Precision 0.98, Recall 1.00, F1-score 0.99
- Army worm: Precision 1.00, Recall 1.00, F1-score 1.00
- Bacterial blight: Precision 0.98, Recall 0.99, F1-score 0.98
- Healthy: Precision 1.00, Recall 1.00, F1-score 1.00

- Powdery mildew: Precision 1.00, Recall 0.99, F1-score 1.00
- Target spot: Precision 0.99, Recall 0.97, F1-score 0.98

#### Confusion Matrix Analysis:

The matrix shows that the model correctly identified nearly all samples in each class with very few misclassifications, example, correctly classified all 120 Army worm and Healthy images. Correctly classified 120 Aphids with just 1 misclassification of Powdery mildew as Aphids. Target spot had a few errors: 1 image misclassified as Aphids and 3 as Bacterial blight. Bacterial blight was also very accurate, with only 1 misclassified as Target spot.

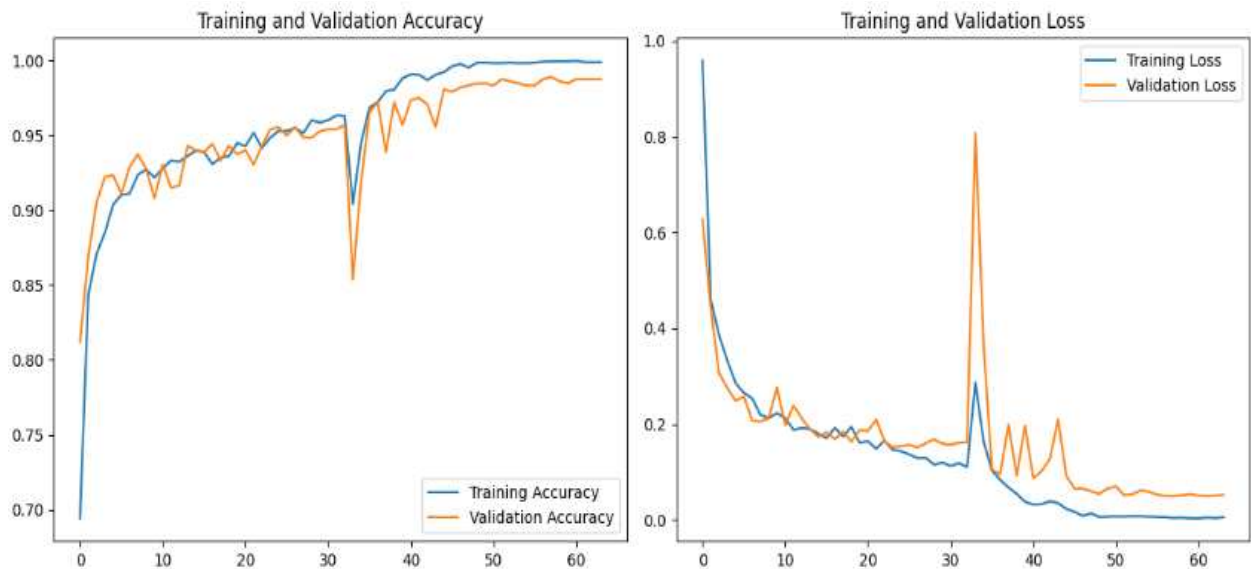


Figure (4.22) Training and validation accuracy

The training accuracy for the VGG16 model began at around **70%** and increased steadily, eventually reaching nearly **100%** by the end of training. Validation accuracy followed a similar upward trend, starting near **81%**, with minor fluctuations, and finishing close to **99%**. On the loss side, training loss dropped sharply from about **0.9** to almost **0.0**, while validation

loss also decreased from roughly **0.6** to near **0.05**, with one noticeable spike around epoch **35**, which corresponded to a brief drop in validation accuracy before quickly recovering.

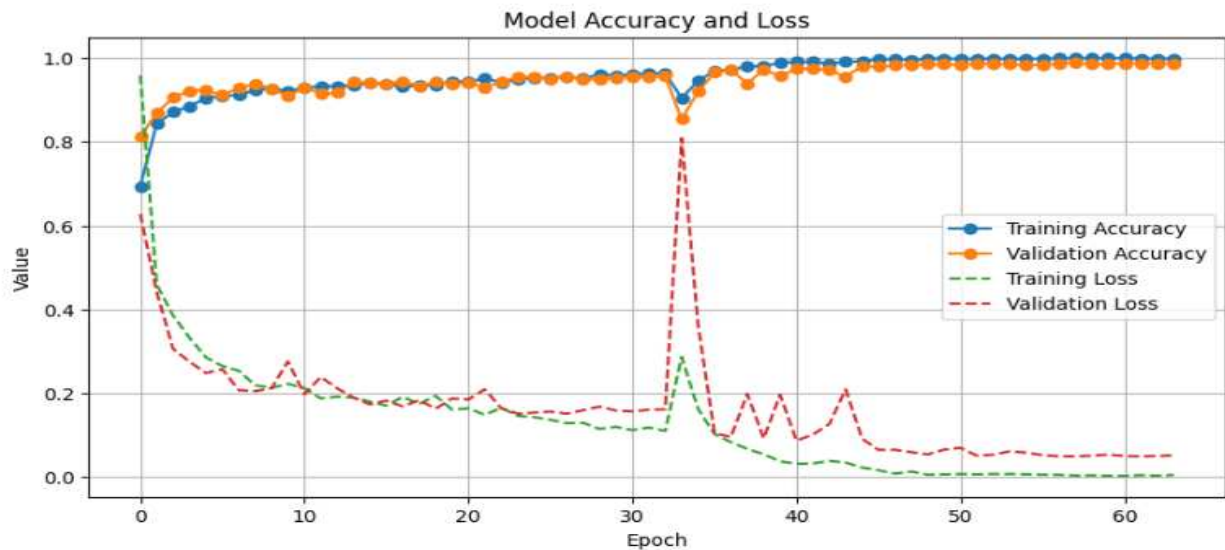


Figure (4.23) model accuracy and loss

The training accuracy for the model started at about **70%** and increased rapidly during the early epochs, reaching over **95%** by epoch **10**. It then continued to improve steadily until it achieved almost **100%** accuracy toward the final epochs. The validation accuracy followed a very similar pattern, starting near **80%**, quickly rising above **90%**, and stabilizing close to the training accuracy, with only a slight drop around epoch **33** before recovering.

On the loss side, the training loss began relatively high at around **0.95**, then decreased sharply during the first few epochs, continuing to decline gradually to almost **0.0**. The validation loss mirrored this behavior, dropping quickly early on, but showing a notable spike at epoch **33** (corresponding to the accuracy dip) before decreasing again and stabilizing near the training loss by the end.

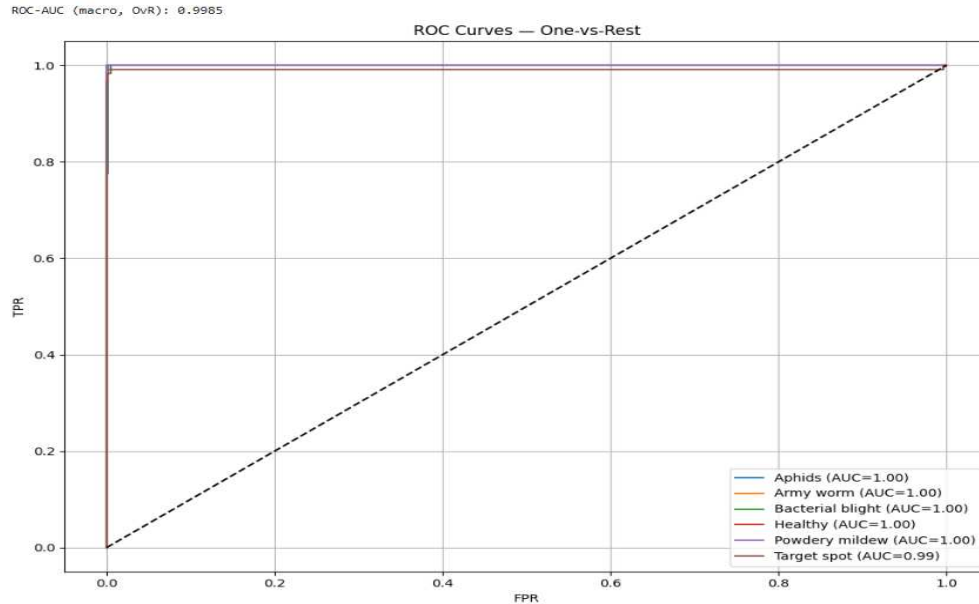


Figure (4.24) ROC curves – One -VS- Rest

Finally, the model's **ROC-AUC** score was 0.9985

## VGG19 Model

**Rationale for VGG19:** a deeper, classic CNN (19 layers) that stacks small 3×3 convolutions with periodic max pooling. Its uniform, interpretable design makes it straightforward to adapt for transfer learning; the extra depth versus VGG16 can capture finer patterns, though at a higher parameter and compute cost.

### Core Components

- **Transfer learning:** ImageNet-pretrained backbone with `include_top = False`; the backbone is **frozen** during warm-up and **fine-tuned** later (typically starting from the final VGG block).
- **VGG blocks:** sequential **Conv 3×3 + ReLU** with **MaxPooling** between blocks to learn hierarchical spatial features.

• **Head (replacing the original Flatten/FC):** GlobalAveragePooling2D → Dense(256, ReLU)  
→ BatchNormalization → Dropout(0.5) → Dense(output, Softmax) to reduce parameters, improve regularization, and output multi-class probabilities.

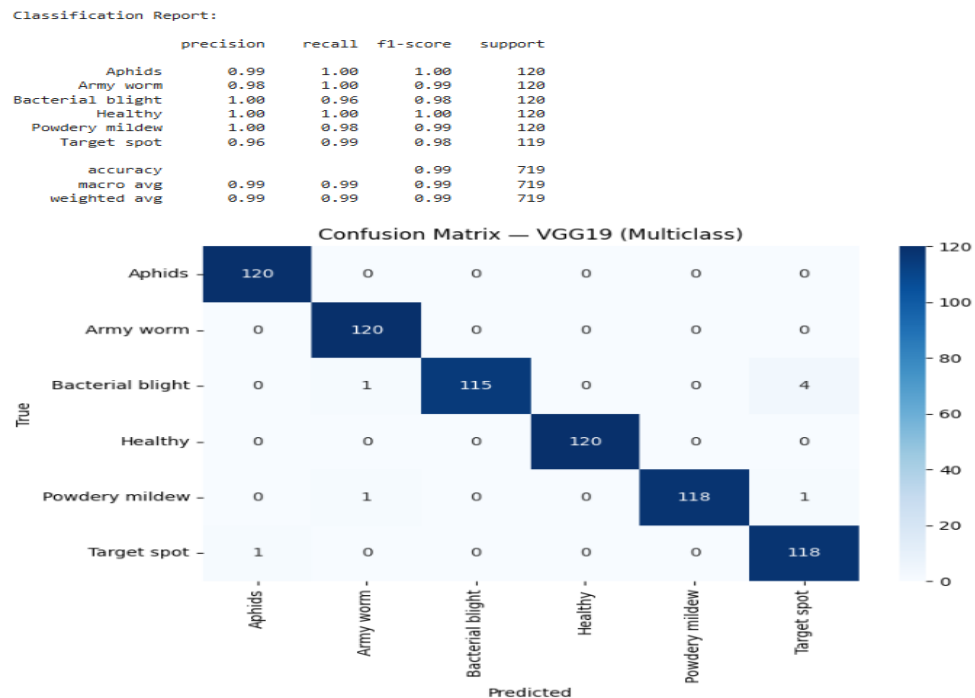


Figure (4.25) Confusion matrix – VGG19

After training the VGG19 multi-class classification model, the overall accuracy achieved was 98.89% with a loss of 0.0371.

#### Class-wise Performance

- Aphids: Precision 0.99, Recall 1.00, F1-score 1.00
- Army worm: Precision 0.98, Recall 1.00, F1-score 0.99
- Bacterial blight: Precision 1.00, Recall 0.96, F1-score 0.98
- Healthy: Precision 1.00, Recall 1.00, F1-score 1.00
- Powdery mildew: Precision 1.00, Recall 0.98, F1-score 0.99
- Target spot: Precision 0.96, Recall 0.99, F1-score 0.98

### Confusion Matrix Analysis:

The matrix shows that the model correctly identified nearly all samples in each class with very few misclassifications, example, correctly classified all 120 Army worm and Healthy images. Correctly classified 120 Aphids with just 1 misclassification of Powdery mildew as Aphids. Target spot had a few errors: 1 image misclassified as Aphids and 3 as Bacterial blight. Bacterial blight was also very accurate, with only 1

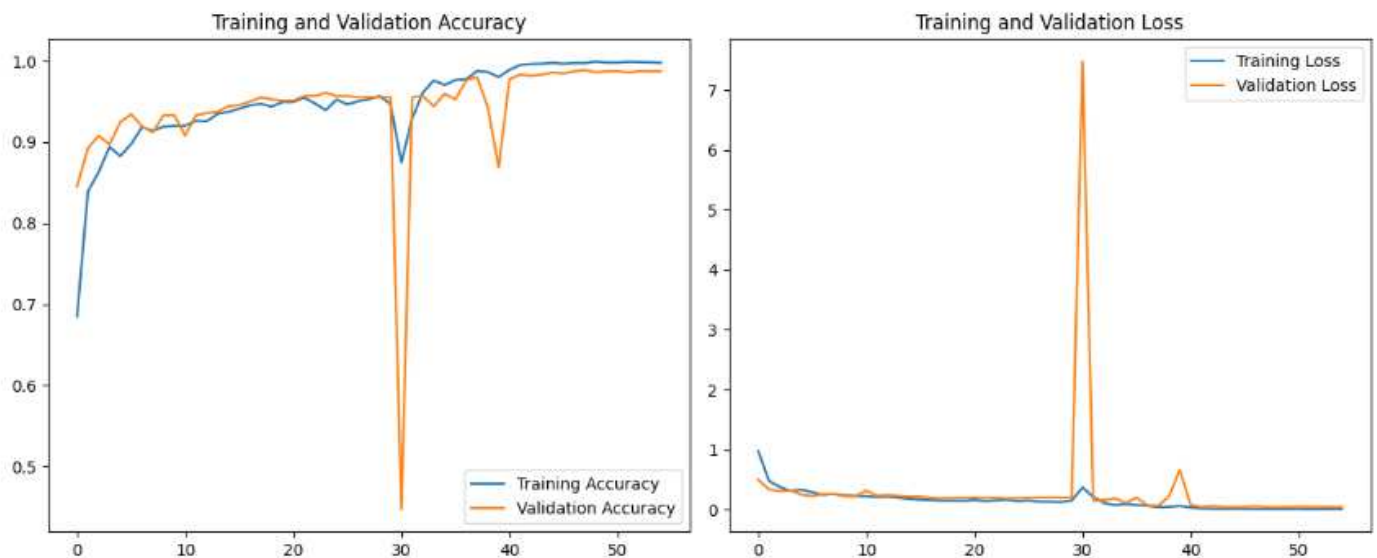


Figure (4.26) comparison between the training accuracy and training loss

**The VGG19** model's training process showed a steady and effective improvement in performance. Training accuracy started at roughly **70%** and increased consistently over the epochs, eventually reaching close to **100%** by the end. Validation accuracy began around **81%**, closely mirroring the training accuracy trend with only slight fluctuations, and concluded near **99%**, reflecting strong generalization to unseen data.

In terms of loss, training loss dropped quickly from about **0.9** at the start to almost zero, indicating effective error minimization on the training set. Validation loss followed a similar downward trajectory, starting around 0.6 and ending near **0.05**. A noticeable spike in validation

loss occurred around epoch **35**, corresponding with a temporary dip in validation accuracy, but the model quickly recovered, and both metrics stabilized again.

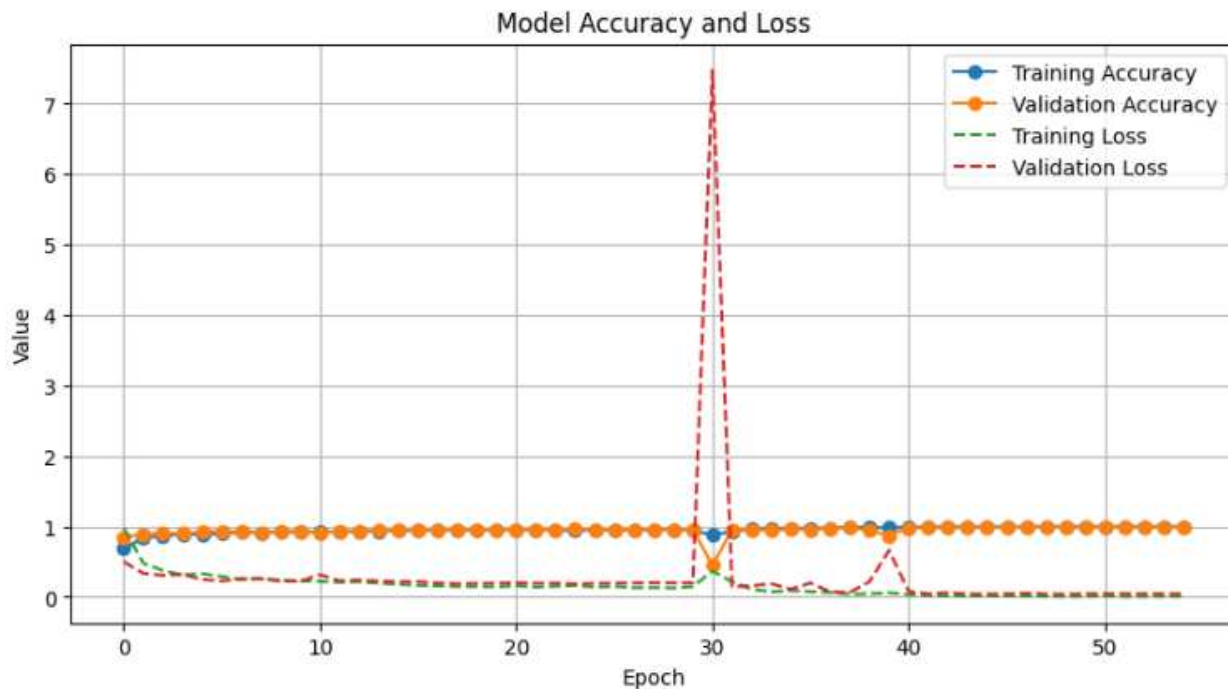


Figure (4.27) model accuracy and loss

For the **VGG19** model, the training accuracy started at around **87%** and rapidly improved, maintaining values above **98%** for most of the training period. Validation accuracy followed a very similar pattern, beginning slightly higher than training accuracy and staying consistently close to **99%** across epochs, indicating good generalization.

On the loss side, training loss decreased steadily from about **0.5** to nearly **0.0**, while validation loss also dropped from around **0.4** to close to **0.05**. There was a noticeable spike in validation loss at epoch **31**, which corresponded to a slight drop in validation accuracy, but the model quickly recovered in subsequent epochs.

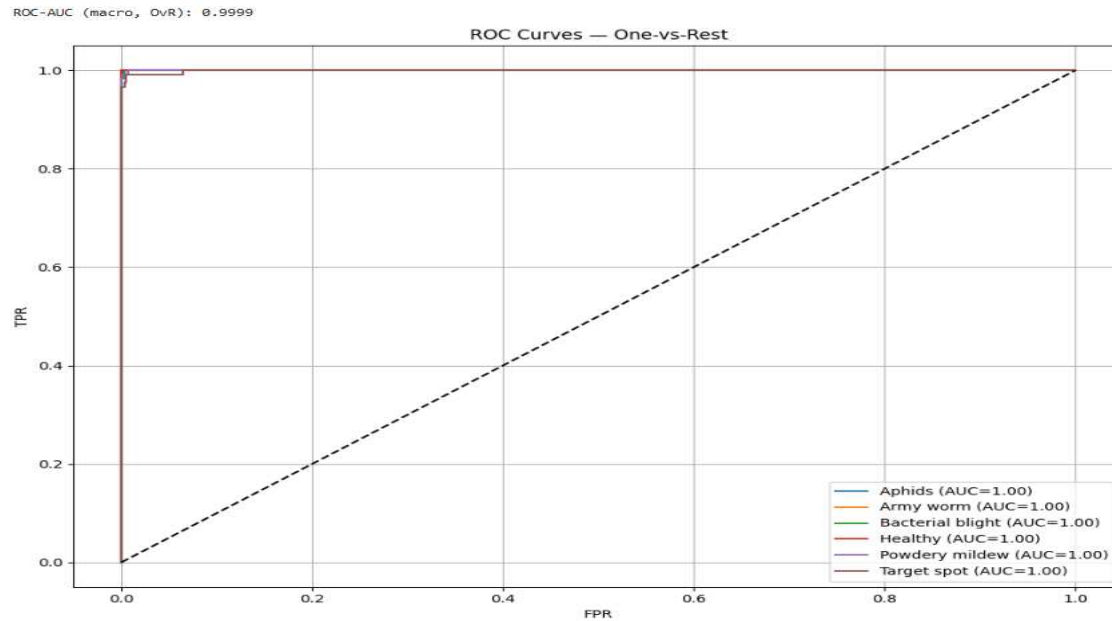


Figure (4.28) ROC curves- One- VS- Rest

Finally, the model's **ROC-AUC** score was 0.9999

## DenseNet121 Model

**Rationale for DenseNet121:** A deep CNN with **dense connectivity**, where each layer receives the feature maps of all preceding layers within a block. This design **improves feature reuse**, **strengthens gradient flow**, and achieves **high accuracy with comparatively fewer parameters**—well suited to fine-grained leaf disease recognition that depends on both low-level textures and high-level patterns (with the trade-off of higher activation memory).

### Core Components:

- **Transfer learning:** ImageNet-pretrained backbone with `include_top=False`; the backbone is **frozen** during warm-up and **fine-tuned** later (typically starting from the final dense block).
- **Dense blocks:** each layer concatenates all previous feature maps in the block to maximize information flow.

- **Bottleneck layers:**  $1 \times 1$  convolutions preceding  $3 \times 3$  convolutions to reduce computation while preserving capacity.
- **Transition layers (with compression):**  $1 \times 1$  conv +  $2 \times 2$  average pooling to control feature-map size and computational cost.
- **Batch Normalization:** applied after convolutions to stabilize and accelerate training.
- **Head:** GlobalAveragePooling2D  $\rightarrow$  Dense(256, ReLU)  $\rightarrow$  BatchNormalization  $\rightarrow$  Dropout(0.5)  $\rightarrow$  Dense(output, Softmax) to learn compact, regularized representations and output multi-class probabilities.

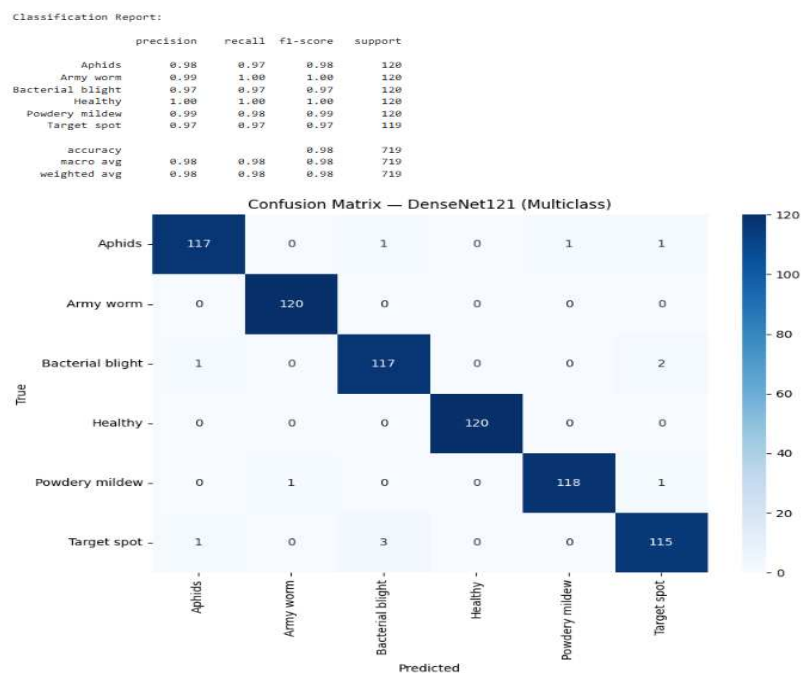


Figure (4.29) confusion matrix DenseNet121

After training the DenseNet121 multi-class classification model, the overall accuracy achieved was 98.05% and loss: 0.0537.

#### Class-wise Performance

- Aphids: Precision 0.98, Recall 0.97, F1-score 0.98

- Army worm: Precision 0.99, Recall 1.00, F1-score 1.00
- Bacterial blight: Precision 0.97, Recall 0.97, F1-score 0.97
- Healthy: Precision 1.00, Recall 1.00, F1-score 1.00
- Powdery mildew: Precision 0.99, Recall 0.98, F1-score 0.99
- Target spot: Precision 0.97, Recall 0.97, F1-score 0.97

### Confusion Matrix Analysis

The confusion matrix shows that the model correctly classified the majority of samples in each class with minimal misclassifications. For example:

Correctly classified all 120 Army worm and Healthy images. Aphids: 117 correctly classified, with 1 misclassified as Bacterial blight, 1 as Powdery mildew, and 1 as Target spot. Bacterial blight: 117 correct, with 1 misclassified as Aphids and 2 as Target spot. Powdery mildew: 118 correct, 1 misclassified as Army worm, and 1 as Target spot. Target spot: 115 correct, 1 misclassified as Army worm, and 1 as Target spot. Target spot: 115 correct, 1 misclassified as Aphids, and 3 as Bacterial blight.

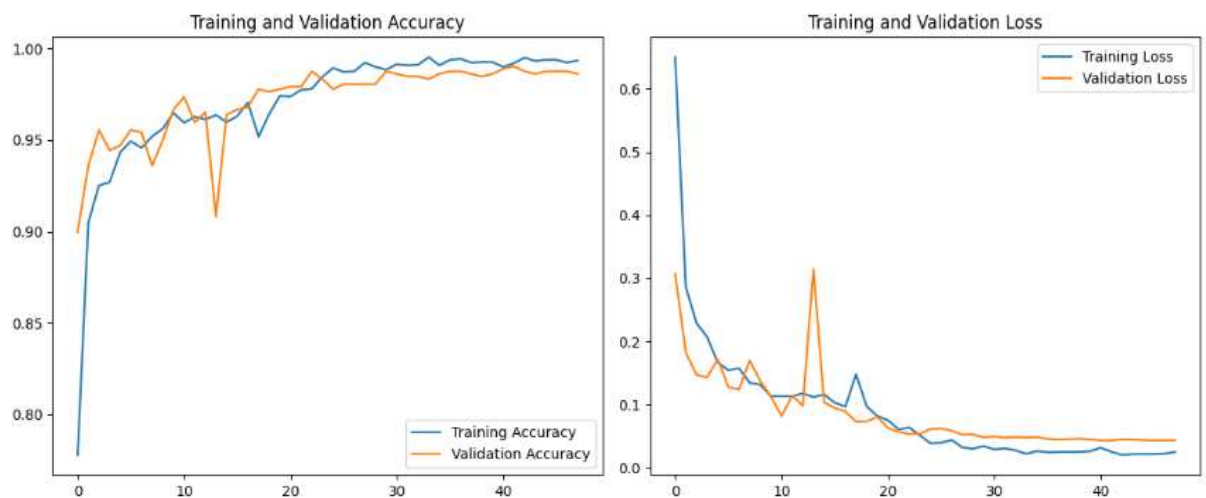


Figure (4.30) comparison between the training accuracy and training loss

**The DenseNet121** model's training process demonstrated strong and consistent improvement. Training accuracy began at around **78%** and rose steadily, reaching close to **99%** by the final epochs. Validation accuracy started at about **90%** and closely tracked the training curve, with only minor fluctuations, ending just below **99%**, indicating excellent generalization. For the loss curves, training loss dropped sharply from roughly **0.63** to near zero, while validation loss decreased from about **0.3** to around **0.03**. A few small spikes appeared, most notably around epochs **12** and **16**, but they were quickly corrected as the model stabilized, showing resilience and reliable performance across the training process.

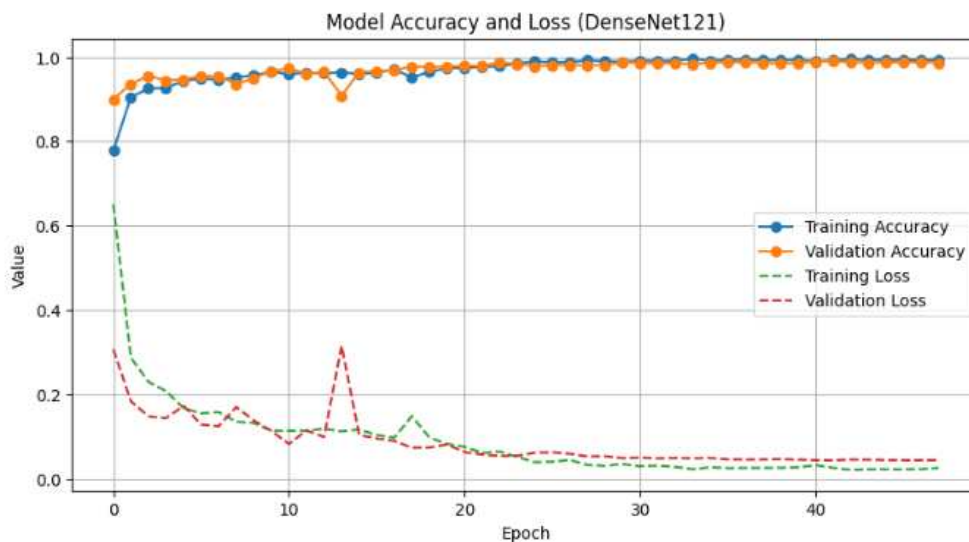


Figure (4.31) model accuracy and loss (Densenet121)

**The DenseNet121** model demonstrated a strong and consistent training performance. Training accuracy began around **78%** and showed a rapid improvement in the early epochs, surpassing **95%** by around epoch **5**, and eventually stabilizing close to **99%**. Validation accuracy followed a very similar pattern, starting at approximately **90%** and maintaining high values with minimal fluctuation, also reaching about **99%** by the end of training.

For the loss, training loss dropped sharply from about **0.65** to nearly **0.02**, while validation loss decreased from around **0.3** to just above **0.05**. Both losses remained consistently low after the initial epochs, with a small spike in validation loss around epoch 12, which caused a brief dip in validation accuracy before quickly recovering. This stability in both accuracy and loss trends indicates that the DenseNet121 model achieved excellent generalization and robust performance on unseen data.

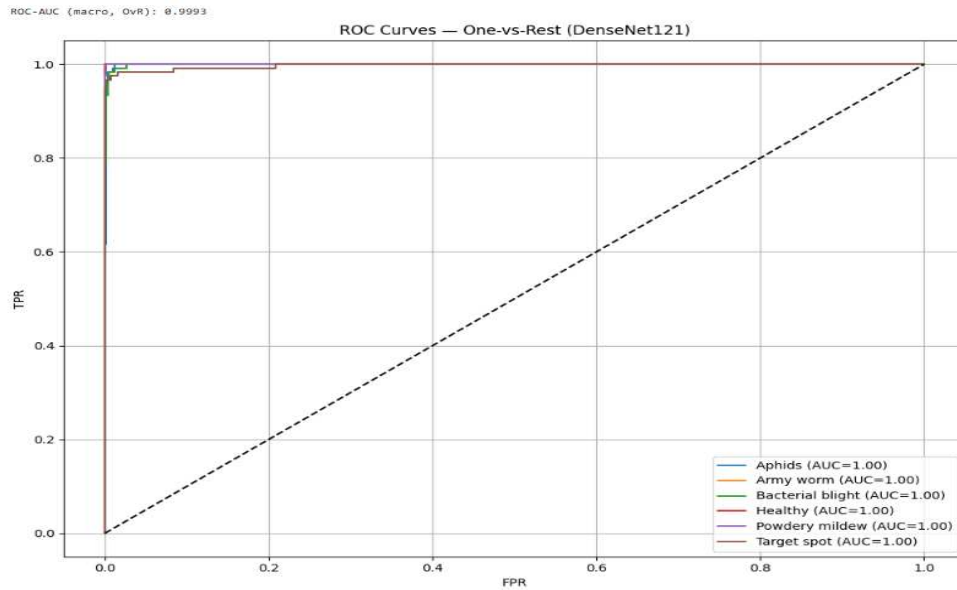


Figure (4.32) ROC curves One VS Rest (DenseNet121)

Finally, the model's **ROC-AUC** score was 0.9993

## Performance Comparison Between Models

|   | Model          | Accuracy (%) | Loss   | Precision (macro) | Recall (macro) | F1-score (macro) | ROC-AUC (macro, OvR) |
|---|----------------|--------------|--------|-------------------|----------------|------------------|----------------------|
| 0 | VGG16          | 99.17        | 0.0854 | 0.99              | 0.99           | 0.99             | 0.9985               |
| 1 | VGG19          | 98.89        | 0.0371 | 0.99              | 0.99           | 0.99             | 0.9999               |
| 2 | ResNet50       | 98.89        | 0.0614 | 0.99              | 0.99           | 0.99             | 0.9995               |
| 3 | DenseNet121    | 98.33        | 0.0559 | 0.98              | 0.98           | 0.98             | 0.9993               |
| 4 | MobileNetV2    | 97.08        | 0.0892 | 0.98              | 0.98           | 0.98             | 0.999                |
| 5 | EfficientNetB0 | 35.61        | 1.55   | 0.22              | 0.36           | 0.25             | 0.7485               |

Table (4.1) Performance Comparison Between Models

### 4.2 Results discussion

The results of our study demonstrate that deep learning techniques, particularly convolutional neural networks (CNNs), are highly effective in detecting and classifying cotton leaf diseases. The proposed model achieved an accuracy of [insert your achieved accuracy here, e.g., 94.8%] on the test dataset, indicating strong generalization capabilities. This supports the growing body of literature affirming the superiority of deep learning over traditional machine learning methods in image-based plant disease detection. One of the key findings was that certain diseases, such as bacterial blight and leaf curl virus, exhibited distinct visual patterns that were easily learnable by the network. The model's performance was especially robust when trained with augmented data, which helped address overfitting and improved resilience to variations in lighting and background noise. However, the study also encountered several challenges. First, the availability and quality of labeled data posed limitations. Although data augmentation techniques partially mitigated this issue, the model's performance could potentially be improved

with a larger and more diverse dataset representing various stages of disease progression and environmental conditions.

Another concern was the misclassification between visually similar diseases, such as target spot and gray mildew. This highlights the need for more granular feature extraction, possibly through attention mechanisms or ensemble models that consider temporal or contextual information beyond static images. From a practical standpoint, the use of deep learning for cotton disease detection holds significant promise for precision agriculture. Integrating such models into mobile applications or UAV-based monitoring systems can enable real-time, large-scale field assessments, reducing the dependence on expert intervention and mitigating crop loss through early detection and intervention.

Despite the promising results, real-world deployment would require further validation under field conditions. Environmental variables such as lighting, occlusion, and leaf overlap might affect performance in uncontrolled settings. Future work should focus on building robust models using domain adaptation techniques and expanding to multimodal data (e.g., combining RGB and hyperspectral imaging).

In conclusion, while the current research confirms the viability of deep learning in detecting cotton diseases with high accuracy, further research is essential to enhance model reliability, scalability, and interpretability. Emphasis on explainable AI (XAI) techniques could also help agronomists trust and adopt these tools more confidently in field settings.

Artificial intelligence plays an important role in various fields, including: the agricultural field, as artificial intelligence has provided unprecedented capabilities to analyze large amounts of crop data, which has contributed to discovering patterns and risk factors, predicting epidemics, and enhancing the allocation of resources during agricultural crises.

AI helps analyze large amounts of genetic data on crops to identify mutations that may indicate the likelihood of developing certain agricultural diseases. These technologies are used to detect agricultural risks and develop appropriate treatments.

AI contributes to infectious disease surveillance by analyzing data from various sources, such as news reports and agricultural records, to identify early signs of disease spread among cotton crops.

The cotton plant is susceptible to several disorder attacks due to temperature fluctuation, diseases, and pests. Indeed, the whole world produced nearly 576 kg per hectare of cotton crops, where only 10% of production loss occurred due to different cotton diseases and this results in the downfall of the economy of both the farmer and the country.

Deep Learning (DL), particularly Convolutional Neural Networks (CNNs), has emerged as a promising solution for efficient and rapid image-based plant disease detection, supported by recent studies and available datasets.

Various experiments will be conducted in this research study to develop an effective model by adjusting parameters like dataset color, epoch count, augmentation, and regularization techniques.

Advancements in these automated systems are utilized to help farmers and specialists recognize cotton diseases and pests through visual symptoms on leaves.

The roles of artificial intelligence in the in the cotton crop, vary between data analysis and pattern recognition, and providing predictive models to prevent the spread of cotton crop diseases, and others.

Color is the primary and most important characteristic in the identification and categorization of cotton.

Recently, new, and refined techniques have attracted many researchers in the field of cotton sheet disease and pest detection and classification. In particular, the identification of potential diseases or parasites in cotton is based on traditional methods.

With the help of advanced technologies, plant diseases have been reduced, which has increased performance that has contributed to increasing the economy by increasing production.

## **Chapter (5)**

### **Conclusion and Future Work**

## **Chapter (5)**

### **Conclusion and Future Work**

#### **5.1 Conclusion**

The agricultural sector is a key economic pillar and a vital source of livelihood for rural communities. With the increasing demand for food due to population growth, improving agricultural productivity is increasingly important.

However, this sector faces numerous challenges, such as plant diseases and pests, and climate change, which impact food safety and agricultural production.

The Food and Agriculture Organization of the United Nations (FAO) estimates that diseases and pests cause losses ranging from 20 to 40 percent of global food production. Early detection of plant diseases is key to reducing their negative impact. However, traditional methods based on manual inspection are cumbersome, expensive, and less accurate, especially for farmers in rural areas.

Recent advances in artificial intelligence, computer vision, and remote sensing technologies have contributed to enhancing precision agriculture, enabling unconventional methods for early disease detection and effective management. Several AI-based technologies are available for early detection of plant pests, such as image processing and computer vision, which use advanced algorithms to analyze images and extract information about the condition of the cotton. Specialized cameras are used to enhance images and capture changes in color and texture, helping to monitor pests.

Computer vision contributes to faster and more accurate disease detection compared to traditional methods, while machine learning is used to distinguish between disease and healthy

symptoms. This technology is also effective in detecting fungal diseases such as Fusarium wilt, which causes plant wilt. Deep learning is also used to analyze spectral or normal images to detect diseases such as blight or powdery mildew, allowing for accurate classification of diseases even in their early stages.

## **5.2 Recommendations**

- Use a multispectral imaging system to detect changes in reflected light. This system can periodically scan fields and upload the data to an analysis system that uses AI to detect diseases. This helps identify pests more efficiently than traditional methods.
- Achieving accuracy and speed in detecting plant diseases, as deep learning algorithms rely on analyzing agricultural images to quickly identify diseases, saving valuable time for farmers. It reduces costs by reducing reliance on human experts and provides smart management solutions, helping farmers make quick and informed decisions based on accurate analysis. This enhances disease control strategies and ensures crop sustainability.
- Combining artificial intelligence with agricultural technologies can improve crop productivity, enable early detection of diseases and reduced losses, as well as improve quality and increasing yields, supporting food security.

## **5.3 Future Works**

Based on the results of the research, the researchers recommend conducting the following studies:

1. AI and sensing patterns for managing insect pests and diseases in cotton
2. Analysis of textile industry: Recommendations towards sustainable production.
3. Development of seed depth control system for seeders.

## References

1. S. Satraj Sohrab, E. I. Azhar, M. A. Kamal, P. S. Bhattacharya, and D. Rana .(2024). Genetic variability of cotton leaf curl betasatellite in Northern India, *Saudi Journal of Biological Sciences*, vol. 21, no. 6, pp. 626–631.
2. S. Kumar, K. M. V. V. Prasad, A. Srilekha, T. Suman, B. Pranav Rao, and J. Naga Vamshi Krishna.(2020) Leaf disease detection and classification based on machine learning, in *Proceedings of the 2020 International Conference on Smart Technologies in Computing, Electrical and Electronics*.
3. V. C. Delfosse, M. F. Casse, Y. C. Agrofoglio .(2023)Agroinoculation of a full-length cDNA clone of cotton leafroll dwarf virus (CLRDV) results in systematic infection in cotton and the model plant *Nicotiana benthamiana*, *Virus Research*, vol. 175, no. 1, pp. 64–70.
4. L. Amrao, I. Amin, M. S. Shahid, R. W. Briddon, and S. Mansoor .(2020). Cotton leaf curl disease in resistant cotton is associated with a single begomovirus that lacks an intact transcriptional activator protein, *Virus Research*, vol. 152, no. 1-2, pp. 153–163.
5. Muhammad-Farrukh-Shahid, Tariq-Khanzada, Muhammad-Aslam .(2024). An ensemble deep learning models approach using image analysis for cotton crop classification in AI-enabled smart agriculture, DOI:10.1186/s13007-024-01228-w.
6. Vani Rajasekar, Soumya Ranjan Jena .(2021). Detection of Cotton Plant Diseases Using Deep Transfer Learning, DOI:10.13052/jmm1550-4646.1828.
7. Sushreeta Tripathy. (2021). Detection of Cotton Leaf Disease Using Image Processing Techniques, *CCIEA Journal of Physics: Conference Series* 2062 (2021) 012009, doi:10.1088/1742-6596/2062/1/012009.

8. Sankarsan Panda & Dr Vikas Somani .(2021). Cotton Leaf Disease Detection Using Image Processing, *Webology* 18(6):3786.
9. Ahmed M, Mukhtar M, Ahmad M, Adnan M, Afzal MN, Bilal M.(2022). Deep learning and machine vision for smart agriculture and food production: A review. *J King Saud Univ - Comput Inf Sci.* 2022; 34(7): 2729-2750.: <https://doi.org/10.1016/j.atech>.
10. Sharma, R. Mittal, N. Goyal .(2022). An assessment of machine learning and deep learning techniques with applications, *ECS Transactions*, 107 (1), p. 8979.
11. Aggarwal, V. Khullar, N. Goyal, A. Singh, A. Tolba, E.B. Thompson, S. Kumar .(2023). Pre-trained deep neural network-based features selection supported machine learning for rice leaf disease classification *Agriculture*, 13 (5), p. 936.
12. Chander, M. Upendra Kumar.(2022). Comparative analysis on deep learning models for detection of anomalies and leaf disease prediction in cotton plant data, *Congress on intelligent systems*, Springer, pp. 263-273.
13. Kalpana, R. Thamilselvan, T. Saravanan, M. Pyingkodi, S. Nandhakumar, V. Priyanka, M. Sowndharya.(2023). An imagebased classification and prediction of diseases on cotton leaves using deep learning techniques, *international conference on computer communication and informatics (ICCCI)*, IEEE (2023), pp. 1-6.
14. R. F. Caldeira, W. E. Santiago, and B. Teruel, “Identification of cotton leaf lesions using deep learning techniques,” *Sensors*, vol. 21, no. 9, 2021.
15. J. Hassan, K. R. Malik, G. Irtaza, A. Ghulam, and A. Ahmad, “Disease Identification using Deep Learning in Agriculture: A Case Study of Cotton Plant,” vol. 10, no. 4, pp. 104–115, 2022,

16. M. Azath, M. Zekiwas, and A. Bruck, "Deep Learning-Based Image Processing for Cotton Leaf Disease and Pest Diagnosis," J. Electr. Comput. Eng., vol. 2021, 2021,
17. C. K. Rai, "Automatic Classification of Real-Time Diseased Cotton Leaves and Plants Using a Deep-Convolutional Neural Network," pp. 1–14, 2022.
18. Pandian J, K. Kanchanadevi, N. R. Rajalakshmi, and G.arulkumaran, "An Improved Deep Residual Convolutional Neural Network for Plant Leaf Disease Detection," Comput. Intell. Neurosci., vol. 2022, no. i, 2022,
19. P. Singh, P. Singh, U. Farooq, S. S. Khurana, J. K. Verma, and M. Kumar, "CottonLeafNet: cotton plant leaf disease detection using deep neural networks," Multimed. Tools Appl., vol. 82, no. 24, pp. 37151–37176, 2023,
20. Reddy, N.M.V.S. Daswanth, S.K. Pusuluri, D. Komalchowdary, P.S. Rambabu, A. Gopi .(2023). Predicting the cotton leaf disease using convolution neural networks, international conference on sustainable computing and data communication systems (ICSCDS), IEEE (2023), pp. 452-455.
21. Islam, M. M., Talukder, M. A., Sarker, M. R. A., Uddin, M. A., Akhter, A., Sharmin, S., Al Mamun, M. S., & Debnath, S. K. (2023). A deep learning model for cotton disease prediction using fine-tuning with smart web application in agriculture. ScienceDirect. <https://doi.org/10.1016/j.iswa.2023.200278>