# AI-BASED PLANT MONITORING SYSTEM FOR IDENTIFICATION OF DISEASES USING CDPR

**ABDUL BASIT 02-133202-010**
**ANIQA FIRDOUS 02-133202-038**
**ABDUL AZIZ 02-133202-069**


**Electrical Engineering Department**
**Bahria University, Karachi Campus**

**2023-2024**

## DECLARATION

We hereby declare that this project report is based on our original work except for citations and quotations which have been duly acknowledged. We also declare that it has not been previously and concurrently submitted for any other degree or award at Bahria University or other institutions.

Signature : _____

Name : ABDUL BASIT

Reg No. : 70061

Signature : _____

Name : ANIQA FIRDOUS

Reg No. : 70432

Signature : _____

Name : ABDUL AZIZ

Reg No. : 70100

Date : _____

## APPROVAL FOR SUBMISSION

We certify that this project report entitled **"AI-BASED PLANT MONITORING SYSTEM FOR IDENTIFICATION OF DISEASES USING CDPR"** was prepared by **ABDUL BASIT, ANIQA FIRDOUS** and **ABDUL AZIZ** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of **Electrical Engineering** at Bahria University.

Approved by,

Signature : _____

Supervisor : Dr. Abdul Attayyab Khan

Date : _____

## ABSTRACT

The agriculture sector of Pakistan has been the largest source of employment for laborers and the greatest contributor to the GDP through the foreign exchange of crops over the years. Pakistan is an agro-based economy, but the sector's contribution has declined to 18.5% for numerous reasons. One of the principal issues is poor crop yield, which is associated with the late identification of plant diseases. To identify symptoms of diseased leaves manually is time-consuming and inefficient. Researchers have found a solution through inducing deep learning and machine learning in image processing. This detailed report introduces an innovative AI-based Plant Monitoring System designed to rapidly and accurately identify. In its early stages, manually placing the symptoms is time-consuming, depending entirely on farmers' visual power, making it labor-intensive. The system integrates advanced machine learning algorithms with high-resolution imaging to analyze plant health by enabling early detection of diseases. This report presents a comparative analysis of different Convolutional Neural Network architectures encompassing ResNet50, VGG16, VGG-19, MobileNet, and MobileNetV2 trained and validated on 18 classes of tomatoes, apples, potatoes and grapes. After this, their accuracy, precision, recall, f1-score, size, and inference time are compared to find the optimal algorithm.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF SYMBOLS / ABBREVIATIONS

| | |
|---|---|
| CDPR | Cable-Driven Parallel Robot |
| ResNet50 | Residual Network |
| VGG-19 | Visual Geometry Group |
| MobileNet | Mobile Network |
| MobileNetV2 | Mobile Network Version 2 |
| ReLU | Rectified Linear Unit |
| CNN | Convolutional Neural Network |
| GPU | Graphical Processing Unit |
| CPU | Central Processing Unit |

CHAPTER 1

# INTRODUCTION

## 1.1 Background

Agriculture is vital for stability in Pakistan’s economy however in recent years, plant diseases have been posing real threats to the productivity of agriculture. Conventional methods for inspecting crop diseases mean manual detection which is time-consuming and inefficient as they only depend on farmers' visual power. A farmer may find it hard to make coherent judgments on the type of disease inflicted upon the leaf due to similar symptoms leading to the wrong use of pesticides. All these drawbacks can result in reduced effectiveness in the management of diseases. Hence, with the development of Artificial Intelligence and technology in the field of machine learning and computer vision, automated disease detection systems are becoming more practical and precise with time. These systems can provide timely insight into the type of disease affecting the leaf in its early stages, allowing the farmer to take action before it spoils the whole plant.

Adopting an AI-based plant monitoring system will not only provide economic benefits by increasing crop yields and mitigating losses but it is also capable of reducing the impact on the environment due to agriculture. These systems make more environmentally friendly and sustainable agricultural methods possible by permitting focused interventions and lowering the need for impetuous pesticide usage.

## 1.2 Literature Review

Plant disease identification has been an object of discussion for a while now among researchers. This section deals with the various techniques that were worked on in the past. Each method has merits and demerits as discussed in the paper below.

The goal of [1] was to find possible solutions to deal with plant diseases that are to be held accountable for the losses in production. This survey discussed the existing gaps in research and the tools required to diagnose plant diseases through deep learning. This article shed light on the importance of moving on from an unreliable approach to a better one, like the use of image processing. This survey is based on logistics and facts, narrating the start of machine learning and its success over the years. Furthermore, this study also cited essential factors like availability of data, sensors, data collection, performance comparison and generalization of models to develop a robust and reliable system for the inspection of diseases. It was determined that plant disease identification accuracy of more than 94% has been attained by the use of deep learning-based picture classification.

Research paper [3] experiments on the dataset named “Apple Leaf Disease Dataset” using the combination of deep convolutional neural network and Apple Leaf Disease Classifier. The dataset is acquired through GoogleNet Inception structure and Rainbow concatenation for detecting leaf disease of apples in the real world. VGGNet-16 and their proposed model, INAR-SSD. In the end, it was found that their designed model was able to detect in real-time only 78.80% accurately, with the detection rate being 23.13 frames per second.

The authors of [4] proposed the use of a deep CNN model with Bayesian learning technique to classify and identify numerous diseases in crops specifically rust spores, black spot, powdery mildew, leaf spot, and botrytis blight. Testing was performed on GPU using Python by selecting 15 diverse classes of unhealthy and healthy pictures of tomatoes, pepper bells and potatoes from the dataset which is extracted from PlantVillage totaling 20,639 pictures. The proposed

approach achieved an accuracy of 98.9% with no signs of overfitting. However, Bayesian learning is a complex model and one which is not widely used. Moreover, this paper also failed to conduct a comparative analysis to validate their proposed technique.

A new approach was introduced by the publishers of [25] for the identification of crop diseases termed as a lightweight Convolutional Neural Network 'VGG-ICNN'. This approach was tested on three main crops: Apple, Maize and Rice, each of them having four, four and five categories, respectively. VGG-ICNN is a lightweight model that has seven layers connected to a global average pooling layer just before ending with a fully connected layer and the total size of parameters is equal to 6,039,108. This model was compared to deep CNN models and lightweight models. VGG-ICNN proved that this model is the best architecture to be used due to less computational power and high accuracy among all the models. However, one downside of this model is that it has a low FLOPS (floating point operations per second) as compared to other models.

By contrasting MobileNet's performance with that of other CNN models—ResNet152 and InceptionV3, in particular—authors in [18] show that MobileNet provides a precision and efficiency balance appropriate for mobile deployment. After that, it outlines the dataset that was put together by Chinese agricultural experts from Shaanxi Province containing pictures of diseased apple leaves. However only two apple leaf diseases—Alternaria leaf blotch and rust—are the subject of this study. It hasn't been proven to work well for diagnosing other illnesses, including ones that affect apple leaves. Additionally, the dataset used in this research is restricted to 334 original pictures. Although MobileNet has a lightweight architecture and requires fewer processing resources, it might not be able to capture complicated patterns as well as more advanced models.

A novel method integrating the Location-wise Soft Attention mechanism into the pre-trained MobileNet-V2 model was introduced by the authors of [31]. The dataset which includes about 1,000 photos of diseases affecting corn, paddy and cucumbers

was taken in actual farming settings. Though the research yields an impressive average accuracy of 99.71%,
and even under disordered surroundings, it reaches 99.13%. There are only about 1,000 photos in the main dataset, so it may be difficult to extend the model's applicability to a wider range of real-world situations. Moreover, the Location-wise Soft Attention mechanism's integration makes the model more complex and increases computational requirements, which could make deployment difficult on devices with limited resources that are frequently utilized in agriculture.

A deep convolutional neural network was trained on an easily accessible dataset garnering 9000 images of tomato leaves to detect five diseases in [32]. This experiment aimed to consolidate computer vision and deep learning to identify diseases in time through the assistance of smartphones. Although the results were promising—99.84% in full-color and 95.54% in grayscale, rescaling the image size to 150 by 150 might have resulted in the loss of necessary details for accurate disease detection. Moreover, the experiment was only focused on tomato leaves restricting the generalizability of the findings to other diseases. Lastly, the paper failed to expand on the requirements of computational power and the practicability of employing this model on low-power devices like smartphones which can limit its practical application for smallholder farmers.

In [33] a comprehensive comparative analysis between models—ResNet50, VGG16, AlexNet, and VGG19 has been performed on thirty-eight diverse classes to identify diseases. Graphs of accuracy and loss have been illustrated in the result section, showcasing that Resnet50 has the smoothest convergence along with a notable accuracy of 99.80%. ResNet50 is known for solving the vanishing gradient problem through its residual learning capability. In addition to ResNet50, the research investigates the performance of VGG16 and VGG19. The basic architecture of VGG16 and VGG19—both of which are well-known for their simplicity and depth—has made them popular choices for image classification problems. This architecture comprises of successive convolutional layers followed by fully connected layers. The 16-weight layer VGG16 and the 19-weight layer VGG19 offer a concession between computational efficiency and depth.

ResNet50's incorporation of residual connections gives a more sophisticated method of managing deeper networks. Despite the success of ResNet50, few restraints can be spotted in the paper. First off, although the study notes the possible use of mobile devices for real-time disease detection, it offers no comprehensive information about the computational performance and viability. Second, given the degree of shape, colour, and texture similarity among various plant species, the model's high accuracy of 99.80% implies that a tiny number of disease classes may still be incorrectly identified. This suggests that more improvement is required to differentiate between closely related species.

Smart farming also requires the use of heavy autonomous robots which are likely to have their pros and cons to deal with. The articles [9] [10] point out the fundamental challenges faced by an agricultural robot such as uncertain and harsh environments that cannot be known in advance, uneven and bumpy terrains, large working areas, and alignment of sensors and tools. These papers also mention the previous efforts to tackle the problems posed by cable robots hence a new idea was presented in the form of CDPR (cable-driven parallel robots), more commonly known as wire/mobile robots. They've been widely researched and it's found that they are easy to move, compact, fast-moving and can perform hefty and complex duties with great precision. However, the operation of CDPR outdoors brings out certain challenges for example the vibration in wires due to the wind, the workspace required and the degree of freedom the end-effector can be rotated.

Table 1.1: Literature Review

| Titles | Limitations | Algorithms | Accuracy/Results |
|---|---|---|---|
| **Comparison of Convolutional Neural Network Models for Mobile Devices** | – High power implementation<br>– Lack of framework | – DenseNet121<br>– DenseNet201<br>– MobileNetV2<br>– VGG16 | MobileNetV2 was the best model overall for running on a mobile device, with the best results in inference time, overall memory usage, and computational complexity. |
| **Leaf Disease Detection Using Support Vector Machine** | – Lack of comparison with other models this model was trained on one type of leaf disease which may not generalize well to other types or plant varieties. | – Support Vector Machine classifier (SVM)<br>– Random Forest | SVM produces better results |
| **VGG-ICNN: A Lightweight CNN model for crop disease identification** | – no real-time deployment | – lightweight Convolutional Neural Network 'VGG-ICNN' | 99.16% |

Table 1.2: Literature Review

| Titles | Limitations | Algorithms | Accuracy/Results |
|---|---|---|---|
| **Transfer Learning Based Plant Diseases Detection Using ResNet50** | – No real-time testing. | – ResNet50<br>– VGG16,<br>– VGG19,<br>– AlexNet | 99.80 % training accuracy |
| **A survey on using deep learning techniques for plant disease diagnosis and recommendations for the development of appropriate tools** | – limited number of images per class<br>– a limited number of bands in multispectral sensors pose a challenge in identifying diseases.<br>– high costs and memory requirements | – CNN | Achieved 94 percent accuracy in the identification of disease plants. |
| **An Autonomous Agricultural Robot for Plant Disease Detection** | | – VGG16 | 97% |
| **Identification of Plant-Leaf Diseases Using CNN and Transfer-Learning Approach** | – Training and testing are done on pictures captured in laboratory conditions only.<br>– As the model was tested in real-time, the performance depreciated by 30 percent. | – InceptionV3<br>– InceptionV2<br>– MobileNetV2<br>– EfficientNetB0 | EfficientNetB0 achieved 99.56% |

Table 1.3: Literature Review

| Titles | Limitations | Algorithms | Accuracy/Results |
|---|---|---|---|
| **Real-Time Plant Health Detection Using Deep Convolutional Neural Networks** | – instances of missing and incorrect detection.<br>– Precision could be improved.<br>– Use of a better-resolution lens | – EfficentDet<br>– FasterRCNN<br>– YOLOv5<br>– YOLOv6 | The accuracy achieved on both the datasets, private and public is 93% at a rate of 120 frames per second by the YOLOv5 model. |
| **Plant Disease Detection using CNN** | – the model was trained using a plain background and singular leaf<br>– Real-time accuracy was just above 44%<br>– Limited datasets | – ResNet34 | an accuracy of 97.2% |
| **ResNet-based approach for Detection and Classification of Plant Leaf Diseases** | – pre-trained models are typically trained on large datasets that may differ from the target domain | – Residual Network (pretrained ResNet34) | 99.40% accuracy |
| **Plant Disease Detection Using Machine Learning** | – Limited datasets<br>– When the models were compared it was found to be less than 70%. | – Machine learning with Random Forest | 70 % of accuracy. |

## 1.3 Problem Statements

According to statistics, crop loss due to poor management of plant diseases globally has been assessed to be over $200 billion, annually. Most of the countries depend on agriculture for their GDP. The points relayed below unfold the issues with the existing arrangement of farming that need to be addressed.

1) **Incorrect diagnosis of disease:** Visual inspection by farmers is prone to errors as they struggle to decipher the difference between a variety of diseases that may look similar to the eye, impeding the efficacious treatment.

2) **Delayed Inspection of Diseases:** The traditional approach leads to amplified crop losses and lower productivity due to delayed disease detection of plants.

3) **Farmers Have Restricted Access:** The widespread adoption of existing technologies is limited by their complexity and difficulty of use for farmers with different levels of technical skill.

**4) Insufficient Real-Time Insights:** Active disease management is hampered by current agricultural techniques' lack of a real-time monitoring system that offers ongoing insights into the health of crops.

## 1.4 Aims and Objectives

i) **Timely Disease Detection:**
To enable timely intervention and minimize the impact on crop yield by creating a plant monitoring system using Artificial Intelligence (AI) that can identify the early warning indicators of crop diseases.

ii) **Precise And Accurate Detection:**
Deploying CDPR with machine learning algorithms to achieve high levels of accuracy and precision in identifying diseases, reducing errors.

iii) **Cost-Efficient:**

Manufacture a cost-efficient system that provides farmers with a compelling solution that accommodates a definite return on investment by curtailing financial losses due to crop diseases.

## 1.5 Scope of Project

AI-based Plant Monitoring system for disease identification makes itself useful in the agriculture sector utilizing CDPR. It makes the detection of diseases easy for numerous crops and helps in cultivating the growth of plants and crops. It potentially replaces the traditional ways of analyzing the crops that were more time-consuming and less efficient. This project brings higher accuracy and better efficiency. Moreover, it neglects the environmental concerns as well that were always lurking with other robots such as drones or other land robots used in agriculture. The design of the robot allows it to inspect the plant from a safer distance without directly causing any harm to the crop making it much more reliable in tougher environments.

## 1.6 Challenges Of CDPR In Agriculture

CDPRs are easy to move, compact, fast-moving, and can perform hefty and complex tasks with great precision. However, the operation of CDPR outdoors brings out certain challenges. As listed below:

- Cable Tension Control: Exact link strain control is critical for the exact situating and control of CDPRs. The tension of cables can vary due to the applied load on the end effector hence it is necessary to balance the load and ensure equal distribution of force on the cables. For given trajectories, cable tensions should be calculated beforehand.
- Limited Workspace: The work area of a CDPR is restricted by the length of its links. This reduces the flexibility of the cable and makes it hard to even the distribution of forces. This can restrict the scope of utilizations where CDPRs can be utilized, especially in huge-scope applications.

- Calibration and Accuracy: CDPRs require exact adjustment to guarantee precise situating and control. However, aligning a CDPR can be tiresome because of the intricacy of the framework and the need to represent link versatility and wear. Under heavy load, cables can sag and lead to nonlinear behaviour which needs to be accurately modelled.

### 1.6.1 Advantages Of CDPR in Agriculture

The fundamental challenges faced by an agricultural robot are uncertain and harsh environments that cannot be known in advance, uneven and bumpy terrains, large working areas, and alignment of sensors and tools. To tackle the problems posed by robots, CDPR (cable-driven parallel robots), more commonly known as mobile robots was presented.

However, some would question CDPR's usefulness and its feasibility over drones. Drones and cable-driven parallel robots (CDPRs) each have their benefits, but when it comes to agricultural inspection and disease monitoring, CDPRs are superior to drones in several ways as mentioned below.

- CDPR is capable of functioning both indoors and outdoors. It can move around with ease and perform effective crop inspections.
- CDPR can provide continuous monitoring due to their ability to operate for long hours without the requirement of charging. Whereas drones have a certain battery life which restricts their operational time.
- Drones can be a hazard to crops as they can involuntarily damage them because of their landing and fans whereas a CDPR monitors from above without any physical touch.
- CDPR can offer a steady foundation for accurate observation. Wind and weather conditions can have an impact on drones, causing instability and inaccurate data collection. This accuracy is essential for seeing early symptoms of crop diseases.

– Lastly, CDPR is less affected by the weather than drones. CDPR also operates in complete silence compared to drones, not disturbing the environment or any wildlife or cattle.

Figure 1.1: Drones in Agriculture[35]

Figure 1.2: CDPR in the agriculture sector

### 1.6.2 Qualities Of CDPR That Aid Can Agriculture

1. The end of the actuator has a fast speed increase, particularly reasonable for quickly moving the end effector around the plants and capturing images.

2. Cable has the qualities of adaptability, high awareness, and great security, and is truly appropriate for harsh conditions where the wind is strong, reducing the possibility of mishaps or crop damage.

3. The quality of adaptability allows CDPR to move across various terrains and layouts of crops. Because of their cable-driven architecture, they may move in a variety of ways and precisely target particular areas of interest.

4. A high-quality load-bearing structure which is a lightweight robot yet constructed with strong materials. This quality allows CDPR to integrate additional components like a camera and Raspberry Pi module winded on the end effector.

5. Possessing a huge work area, particularly reasonable for covering a large area of plants to be monitored.

## 1.7 Sustainable Development Goals of Project

Goal 3: Certainty of wellness and healthy life for people of all ages.
Goal 9: To promote and develop sustainable industrialization and promote innovation.
Goal 15: Protect, restore, and promote sustainable use of the ecosystem.

## 1.8 Environmental Aspects of Project

- Sustainable Agriculture

Sustainable agriculture describes the uninterrupted production of crops to meet the continual community demand. Smart farming is the probable solution to carry out this challenge.

- Reduced Chemical Usage

Over usage of chemicals in crops is harmful to the water and soil if it directly reaches it. Using robots is the sustainable way for this as we can directly spray pesticides on the affected plants and trees.

- Increases Food Security

Automated agricultural robots can be designed to improve the production of crops. These robots perform repeated tasks that normally humans take more time to complete.

- Preservation of Beneficial Organisms

Controlled and per-determined usage of pesticides can be helpful in the preservation of organisms. Beneficial organisms contain bacteria and fungi that help in the growth of crops.

- Healthy Ecosystem

Secure production and security of crops form a healthy ecosystem. A healthy ecosystem describes the environment in which all needs of crops are fulfilled for the community and with improved quality.

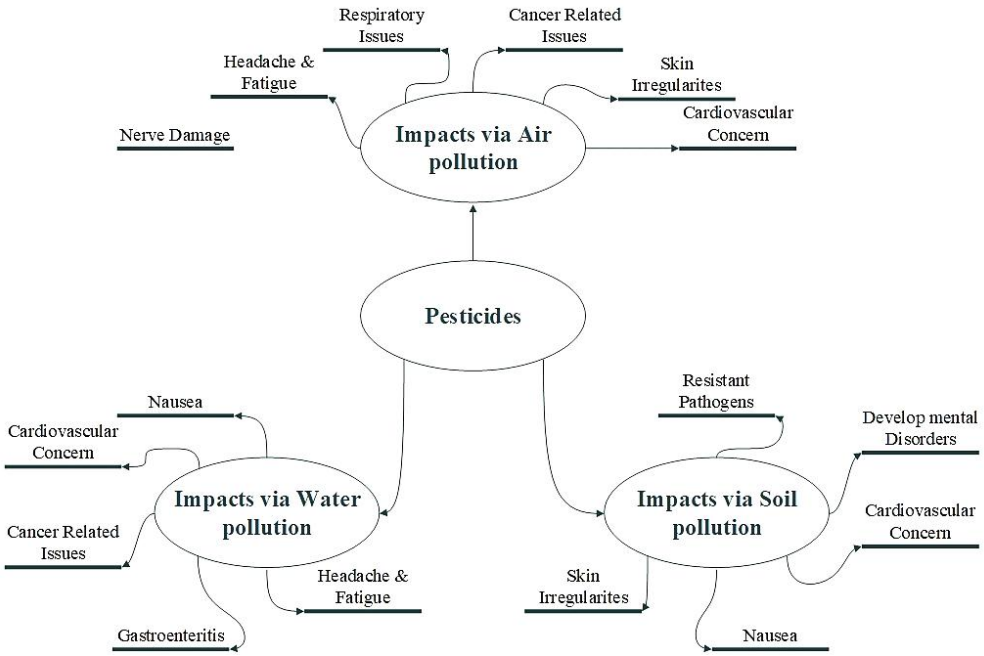


Figure 1.3: Environmental Impacts

CHAPTER 2

# DESIGN AND METHODOLOGY

## 2.1 Data Acquisition:

As shown in figure 2.1, The data we have gathered is of 4 different species of plant. The names of those plant species are given below:

I. Grape
II. Tomato
III. Potato
IV. Apple

In total, we have 18 classes. Each class contains 400 images and in total, our dataset contains 7,200 images from which 75 percent of the images are used in training, 15 percent of images are utilized in validation and 10 percent of the images are used in testing.

## 2.2 Image Pre-processing:

The second step is image pre-processing, a crucial step required to transform the input image into a suitable format for the algorithm to process it. It improves the image data and resizes it to the standard image size so that all the input images have the same dimension. In our pre-processing step, the resolution of all the input images is reduced to 224*224 pixels.

## 2.3 Data Augmentation:

Thirdly, random transformations such as flipping the images vertically and horizontally or rotating are applied to the images during training, which helps in generalizing the model better by subjecting it to the discrepancies in the input images. In this process, images were first rotated by 45º then by 90º and then by 180º.

After applying data augmentation our dataset size increased to a total of 28,800 images, each class containing 1600 images which were later divided into further

parts such as 75% for model training, 15% for model validating and 10% for model testing.

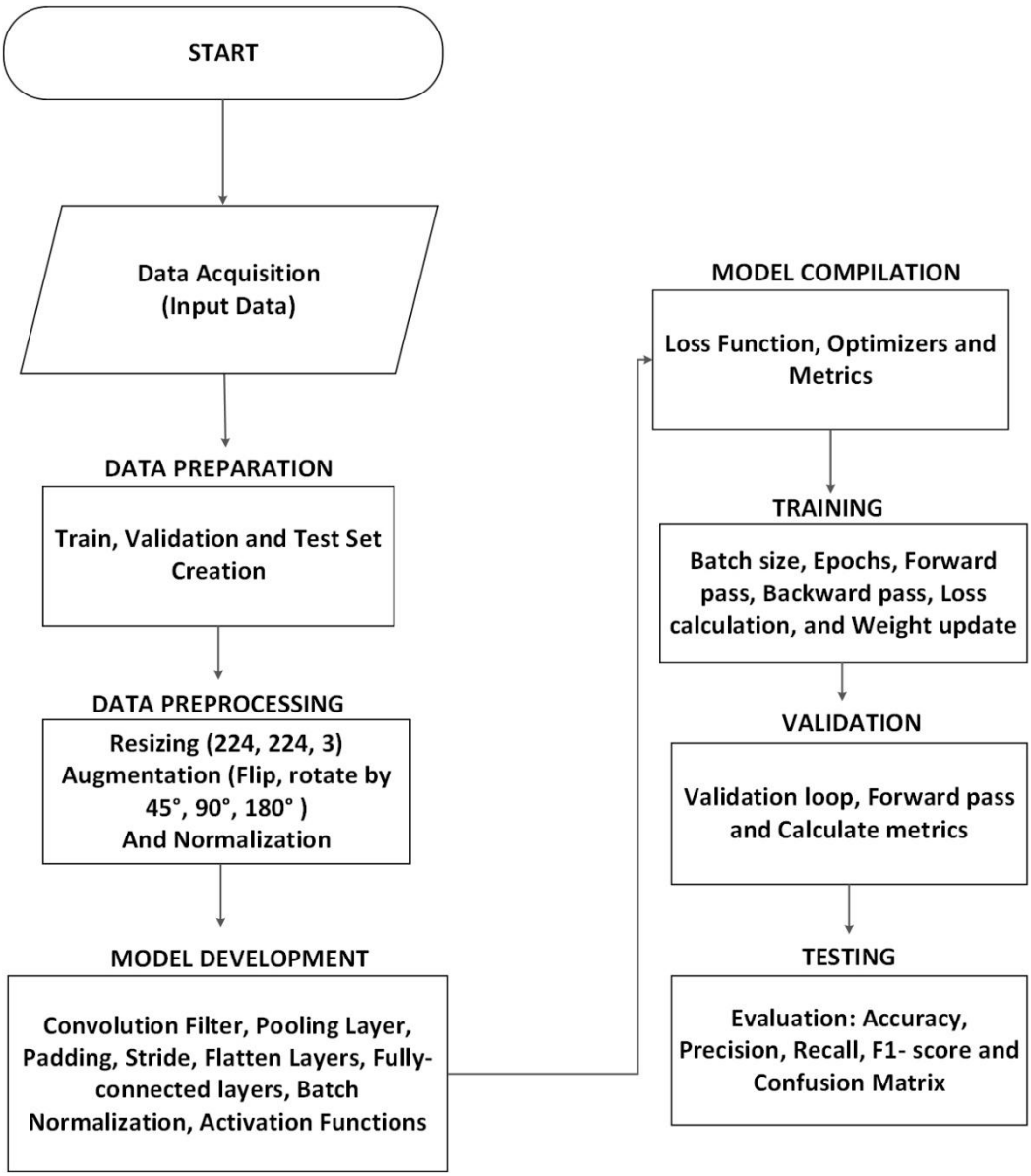


Figure 2.1: Flowchart

## 2.4 Finalized Dataset:

Below in figure 2.2, are the images of leaves we have trained and validated our model on, namely

1) Apple: scab, black rot, cedar rust, and a healthy leaf
2) Grape: black rot, isariopsis, esca, and a healthy leaf
3) Potato: bacterial spot, early blight, late blight, and a healthy leaf
4) Tomato: bacterial spot, early blight, mosaic virus, Septoria leaf spot, leaf mold, target spot and a healthy leaf

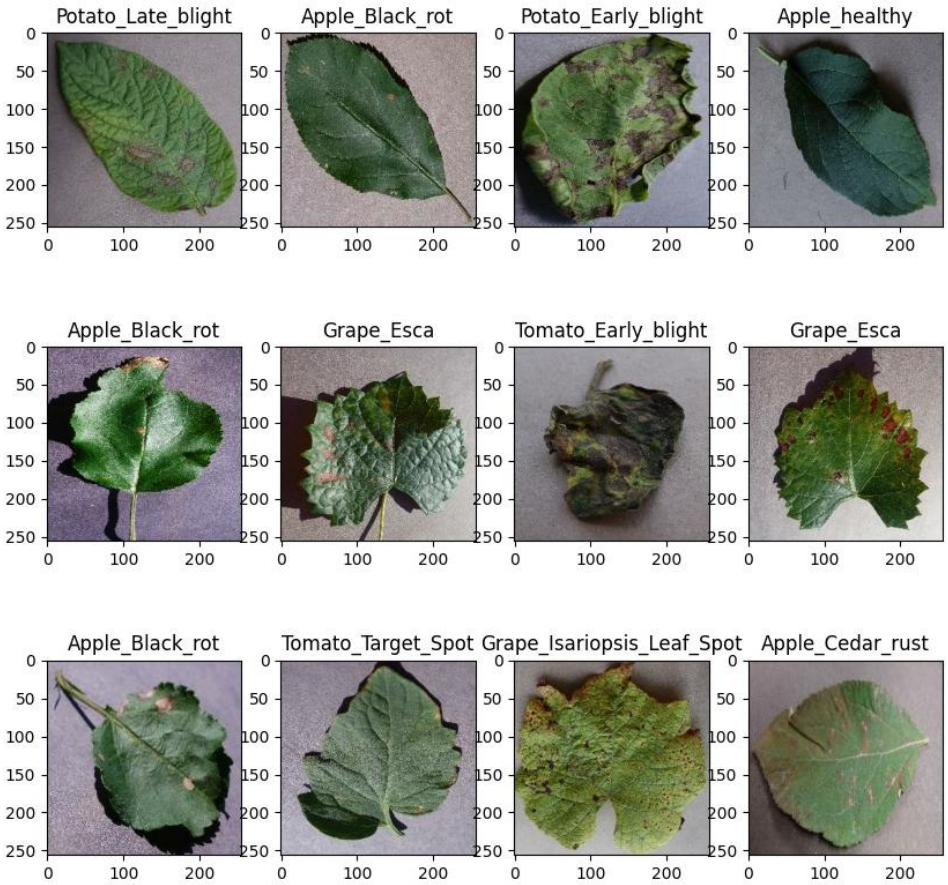


Figure 2.2: Some samples from the dataset.

## 2.5 Working Of CDPR with Raspberry Pi

Figure 2.3 shows the working of our system, where after training the model weights will be downloaded. The downloaded weights will then be used to predict the disease of plants. To do so we will load the weights of our trained model on Raspberry Pi and then according to those weights, our algorithm will run and make inference on plants. CDPR will move above plants and the end effector of CDPR will be carrying our end device (Raspberry Pi) and camera module. After reaching the aerial view of the plant, our classification process will begin and inference will occur. After which the results will be displayed on the screen predicting the class of plant and identifying the disease type if there is one.

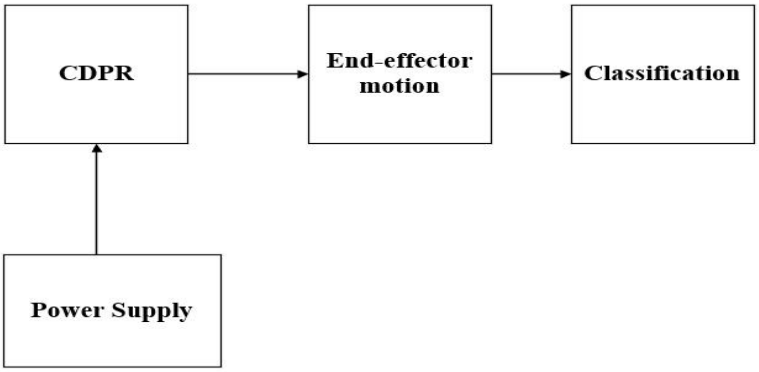


Figure 2.3: Block diagram of the system

## 2.6 Classification Process:

As the block diagram in Figure 2.4 illustrates, initially the acquired weights file of the model and text file containing the names of classes will be loaded into Raspberry Pi. The Camera will be initialized and the required mode, either video or pictures will be

selected by setting camera parameters such as FPS (frames per second) and resolution in megapixels. After setting these parameters, images will be continuously captured and stored, and this will be done with the help of Python packages namely sub-process, pi camera, and OpenCV. The incoming image frames will be pre-processed to match our model input shape (224 by 224), through computer vision Python frameworks such as OpenCV and Pillow. The preprocessed data will be used

for running inferences and making predictions on the preprocessed frames using the TensorFlow-lite framework of Python.

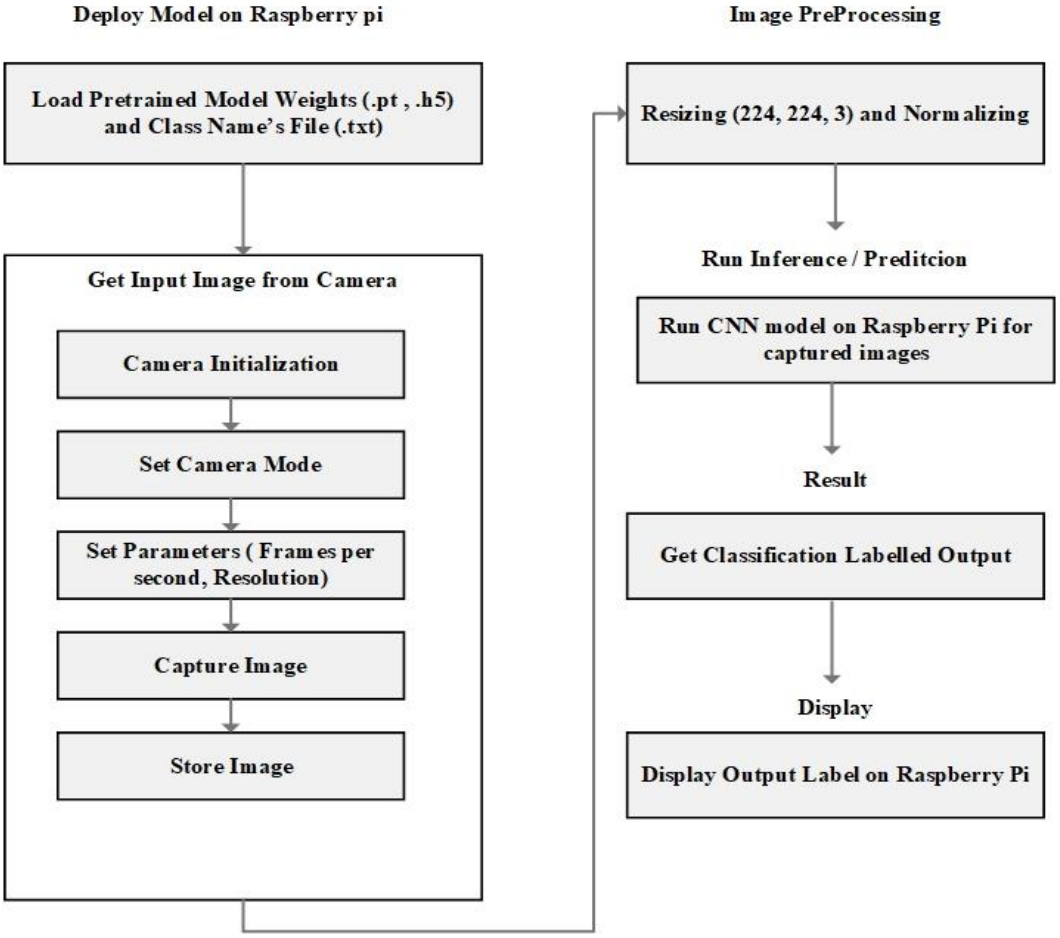


Figure 2.4: Block Diagram of the Classification process

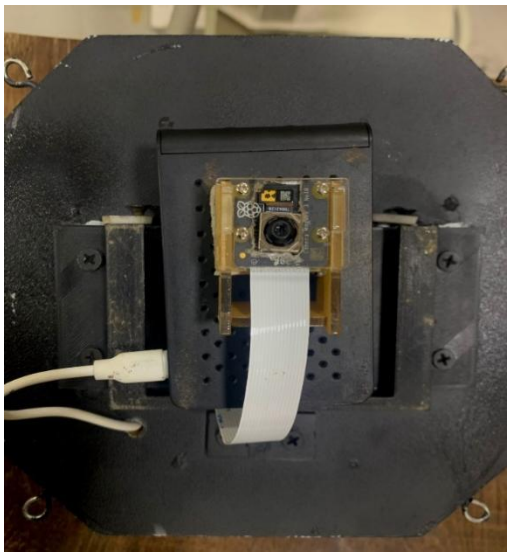

Figure 2.5: CDPR integrated with Raspberry Pi and Camera Module

CHAPTER 3

# DESIGN AND IMPLEMENTATION

## 3.1 CNN (Convolutional Neural Network):

Convolutional Neural Networks (CNNs) revolutionized image classification by imitating the human visual system's ability to recognize patterns and features in images. A CNN consists of multiple layers starting from the input layer that detects input data sent which is then passed through convolutional layers that pick features from it through a series of convolution operations with learnable filters and create feature maps. These filters are trained to detect specific patterns by adjusting their weights during the training process. After the raw image is passed through the convolutional layers, activation functions like ReLU (Rectified Linear Unit), and Sigmoid are applied to extract features. The next layers are pooling layers that discard the unnecessary information in the data which improves the model's computational efficiency, followed by fully connected layers, which are responsible for making predictions based on what it has learned for the features that have been detected in the image. As we are using five different CNN models, the three of our selected CNN models (ResNet50, VGG-16, VGG-19) are just normal typical models while the other two (MobileNet and M0bileNetV2) are Depth-wise CNN models. The characteristics of both types are as follows:

### A) Typical Convolutional Networks:

This type of convolution can autonomously extract features from images by running different kernels over images which results in the creation of feature maps that contain content about the image’s texture, shape, edges, and others. These maps are then down-sampled using pooling layers which also slide weighted kernels to reduce the spatial dimensions as well as consider the important features by considering the

maximum or average value which generally is the linear combination of the kernel's area.

There are a series of cascaded convolutional and pooling layers to effectively extract features from data. At the ending side of the model, we normally find flattened layers that are connected to convolutional and pooling layers they convert the previous layer outputs to a 1-dimensional array and feed that to Fully-connected layers where every neuron of the input layer and output layer are connected to learn the complex relationship of feature, this layer is followed by Activation functions which map the weights into class probabilities.

But the thing to be noted here is that while applying kernel filter in convolutional layers, the filter is applied to all channels at once which leads to the requirement of more computation and memory especially for larger datasets, and may suffer from overfitting when dealing with small datasets. These models also require high computation resources while deploying on applications to run inference and make predictions but usually offer good accuracy with too many parameters and high model size which make them perform poorly in real-time applications.

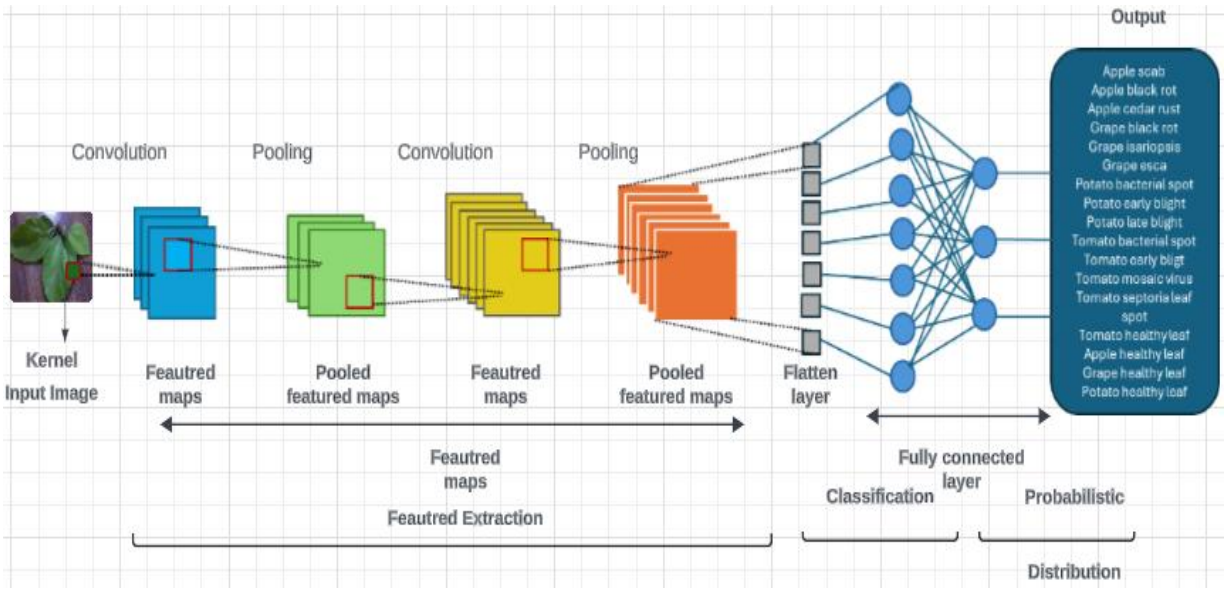


Figure 3.1: Typical CNN

**B) Depth-wise Convolutional Networks:**

The major difference between depth-wise and normal convolution is that it further breaks down convolution layers operation into two. In depth-wise convolution kernels are run over images but channel by channel, creating each channel feature map, and then using point-wise convolution the resultants are computed. The other layers' operations are almost the same such as pooling layers, activation functions, and others.

But this scheme offers a wide range of advantages. Firstly, it reduces the computational cost and improves generalization by preventing the model from exactly remembering the training data. Most importantly it reduces the model size which reduces memory requirement making it an optimal choice to use for embedded devices and in real-time applications due to its high inference speed. So, hence it might have a 5-10 percent trade-off on accuracy but can be used for end device deployment.

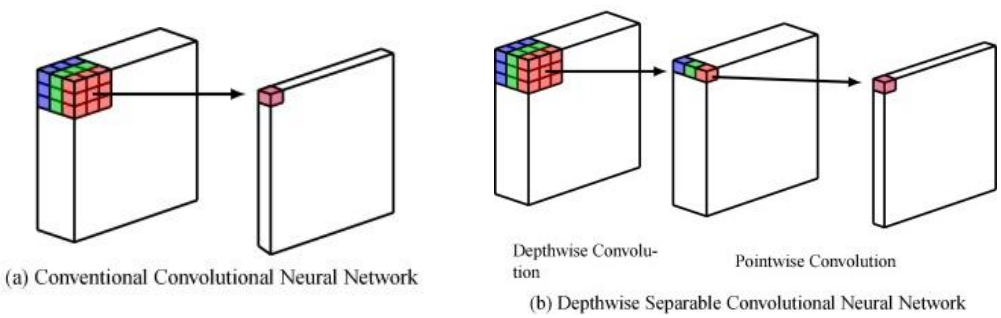


Figure 3.2: a) conventional CNN b) Depthwise Separable CNN [38]

## 3.2 Transfer Learning

In machine learning, there is a technique called transfer learning where pre-trained models are used as initial points for models to start, the idea is to use pre-learned features of pre-trained models and adapt them to new applications, rather than

training a whole model from scratch which helps in reducing training time and improved performance of models. Transfer learning has been used for better results on limited datasets especially in areas where data is not abundant such as medicine, agriculture underwater imaging, etc. In this technique, pre-trained model weights are loaded and then the last few layers are trained to adapt the model weights according to your classes, which allows the model to inherit knowledge from large datasets like COCO, ImageNet CIFAR etc.

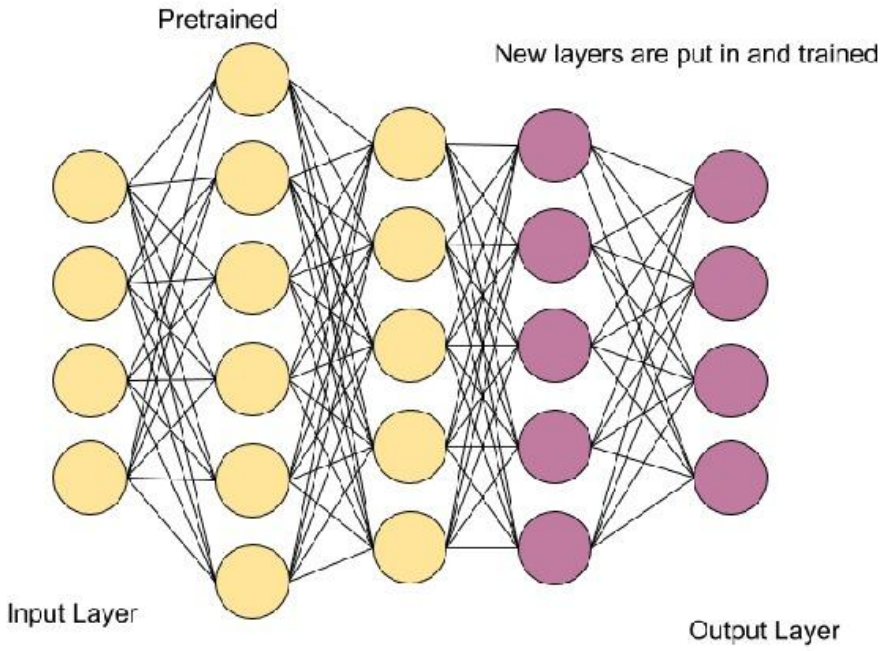


Figure 3.3: Model of Transfer Learning [37]

## 3.3 Resnet50

ResNet50 is one of the renowned CNN architectures as shown in Figure 3.1. This architecture comprises 50 layers and 16 residual blocks. Every residual block has several convolutional layers. What made Resnet50 the most notable architecture is its introduction of skip connections that address the problem of vanishing gradient which is a common occurrence in other CNN models, where the gradient becomes very small or sometimes vanishes while reaching the early layers of architecture while updating weight during back-prorogation which uses chain rule of derivatives, so while using this chain rule the repeated multiplications may make the weights

extremely small. ResNet50 introduced skip connections as solution which allows the information to move directly from input to output of model, skipping one or multiple layers that allow the model to learn the residual results that map the input to the required output rather than to learn all features from scratch. In this architecture, there are mainly two blocks, Identity block and Convolutional block. The identity block passes the input from multiple convolutional layers and then adds the input to the output directly using skip connection. On the other hand, convolutional block performs a 1x1 convolution before sending the input to the output, the reason to do this convolution is match the size. Furthermore, the model after taking the image performs 7x7 convolution and max pool it with stride 2 then feeds it to series of convolution and identity blocks. After that it average pools the image with stride to reduce spatial dimensions and uses fully connected layers with thousand neurons for learning complex pattern of image and classification followed by Softmax activation function. A pre-trained ResNet-50 model refers to a convolutional neural network (CNN) architecture that has worked and trained on larger datasets like ImageNet before allowed for public use. Frameworks like Keras and TensorFlow provide pre-trained ResNet-50 models that can easily be downloaded and used for your specific applications. Using a pre-trained model can preserve evident time utilized on training and in collecting data compared to training a DNN from scrape.

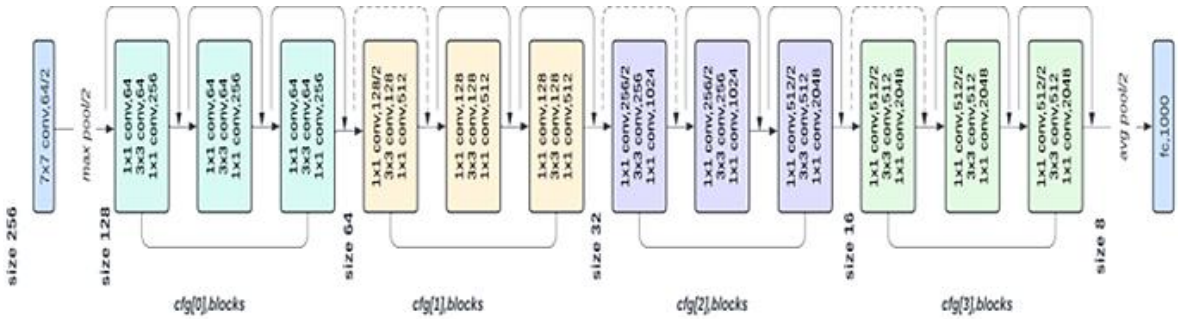


Figure 3.4: ResNet50 Architecture

## 3.4 VGG-16

VGG is another CNN architecture with multiple layers. Due to its success on ImageNet, VGG is frequently used for transfer learning on pre-trained models. Its pre-trained weights on ImageNet can be fine-tuned for specific image processing tasks, allowing effective use with smaller datasets. Reducing the convolutional kernel's size and increasing the depth of convolution layers made VGG stand out among other CNN models.

As shown in Figure 3.2, VGG-16 has 13 convolutional layers and 3 fully connected layers. The convolutional layers in VGG16 consist of a series of 3x3 kernels, a padding of 1, and a stride of 1. The system stacks multiple convolutional layers before downsampling the spatial resolution through max-pooling layers. After the convolutional layers, the structure is continued by 3 fully connected layers for classification by a final SoftMax layer.

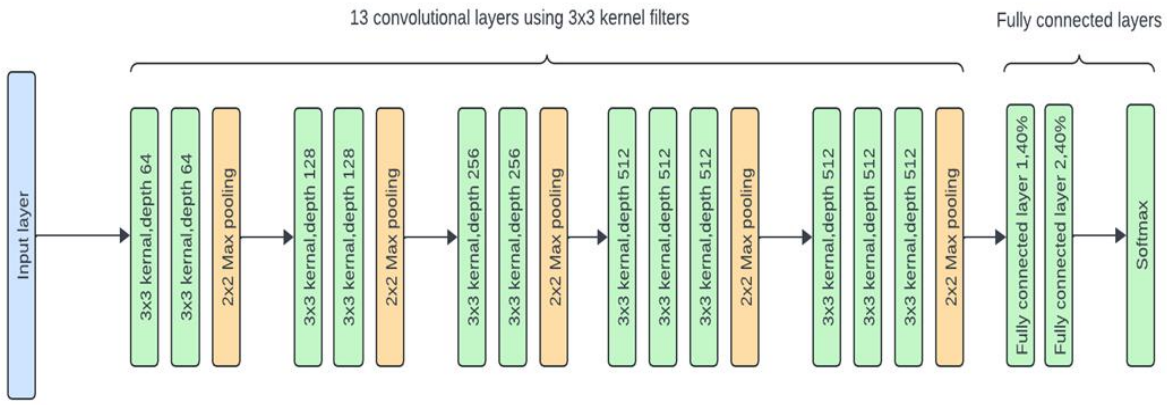


Figure 3.5: VGG-16 Architecture

## 3.5 VGG-19

As Figure 3.3 shows, VGG-19 has sixteen convolutional layers and three fully connected layers. VV-19 bears resemblance to VGG-16 but has more parameters and an increased depth which contributes to the model's capacity to learn complex

representations. Convolution layers operate by extracting features of the image data. These features are passed through the max pooling layer that minimizes them to a number which will prevent overfitting from occurring since it can impact the model's ability to generalize data negatively. In the end, these features assist the fully connected layers in classifying the leaf images.

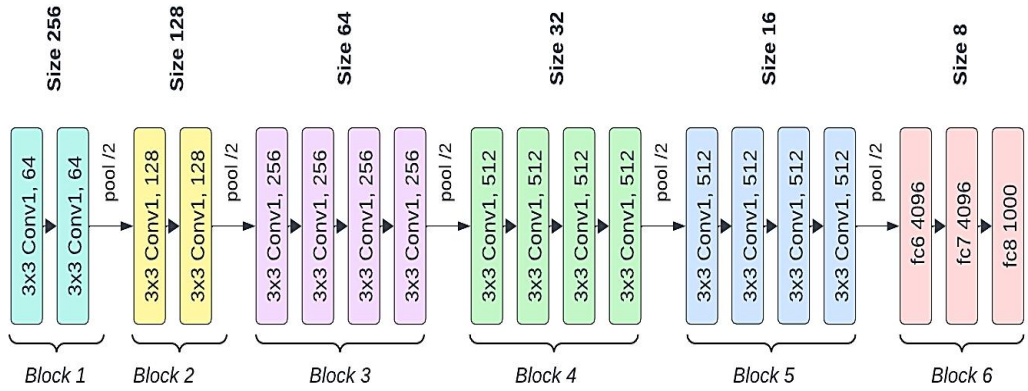


Figure 3.6: VGG-19 Architecture

## 3.6 MobileNet

MobileNet was first introduced in 2017 by Google. It is a class of lightweight deep CNNs that are faster and smaller than others which improves their computational efficiency and due to this very reason, these models are perfect for mobile and edge applications that need image processing and computer vision skills. They are specially designed to operate on low-power devices with constrained computing capabilities. MobileNet has limited size and its inference time is better than other CNN models both on CPU and GPU as well.

The architecture of MobileNet incorporates two layers. The first layer, known as a depthwise convolution, carries out lightweight filtering by applying a single convolutional filter to each input channel. Second layer is known as pointwise convolution for establishing new features by using input channels to calculate linear combinations and applying convolution of 1 by 1.

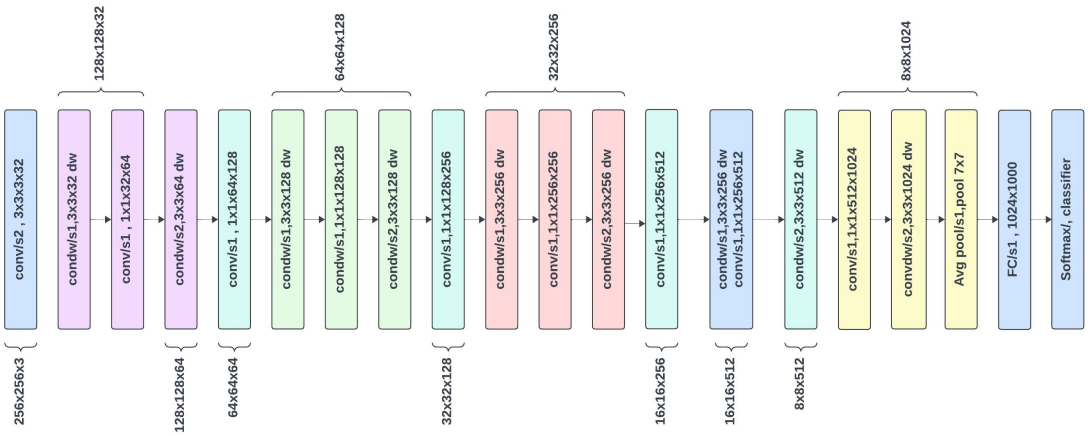


Figure 3.7: MobileNet Architecture

## 3.7 MobileNetV2

MobileNetV2 was introduced by Google in 2018, a revised, enhanced version of MobileNet. This versatile model offers a compelling solution for mobile and edge devices where computational efficiency is limited. MobileNetV2 can be implemented upon pre-trained models that have large datasets such as PlantVillage which can be fine-tuned on smaller plant disease datasets, leveraging the learned features for improved performance in disease detection tasks even with limited labelled data.

The architecture of MobileNetV2 introduces extra useful features for image classification through depthwise separable convolution optimization that are not found in the architecture of its predecessor. The architecture of MobileNetV2 is displayed in Figure 3.5. Features of this architecture are explained below:

- Depthwise separable convolutions optimization incorporates inverted residuals with linear bottlenecks.
- Inverted residuals feature involves widening the sum of channels in the input using a 1x1 convolution since the following 3x3 depthwise convolution reduces the parameters. Then squeezing the channels again by using another 1x1 convolution.

- Linear bottlenecks are crucial in learning the complicated representations of the data while maintaining the dimensions of feature map between the layers.

Another important feature is the induction of skip connections which links the residual block with wide layers that ease the flow of gradients during training, help in alleviating the vanishing gradient problem and enable faster convergence.

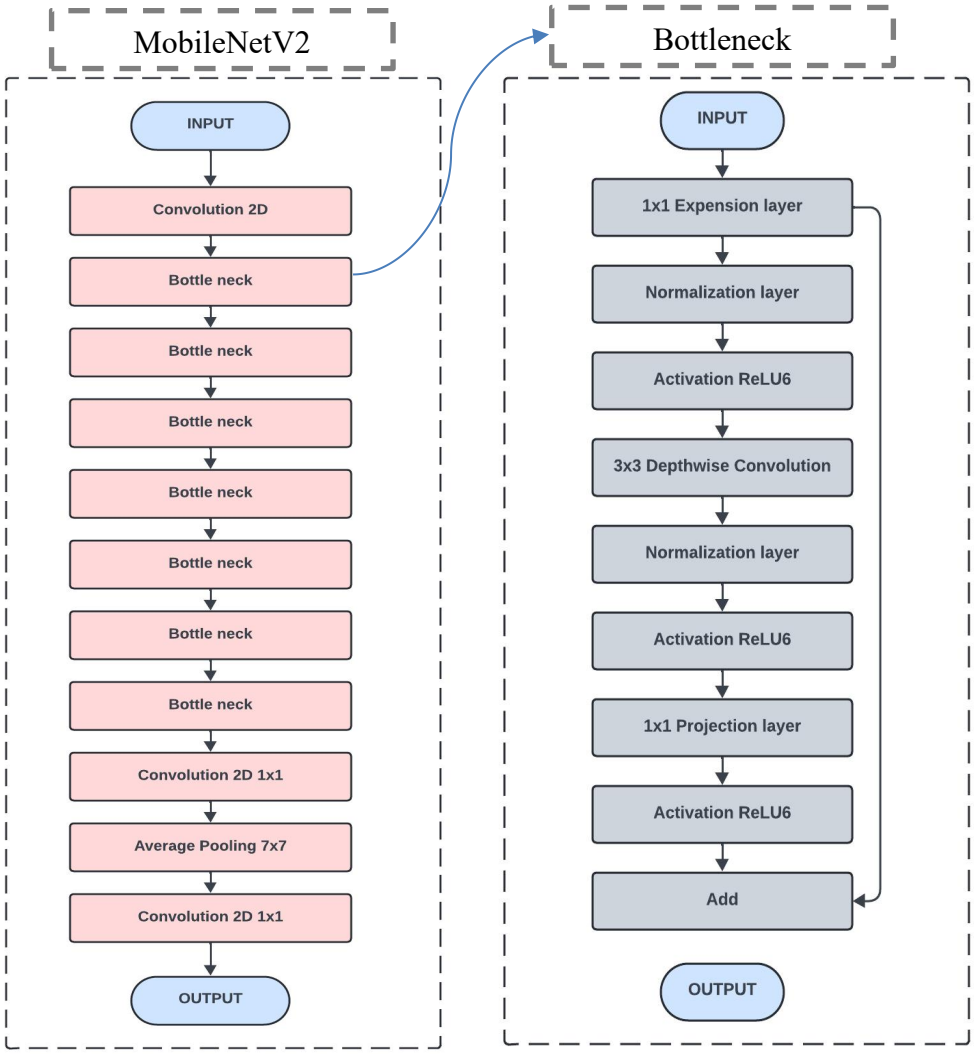


Figure 3.8: MobileNetV2 Architecture

## 3.8 Raspberry Pi 4 camera module Interfacing

The module of Raspberry Pi 4 shown in Figure 3.6 is interfaced with the camera module to collect offline real-time data. Initially, the Raspberry Pi is set and connected to the camera module which will be mounted on the CDPR and where it can be moved in three degrees of freedom linearly. Then we installed the necessary libraries such as OpenCV and Pillow to deal with images, and for CNN, the Python framework we are using is TensorFlow and TensorFlow Lite. It allows developers to create and run deep learning models without having to deal with all the complex details of neural networks such as mathematical operations. Then the camera is tested in different lighting to check the sensitivity of the camera to the light.

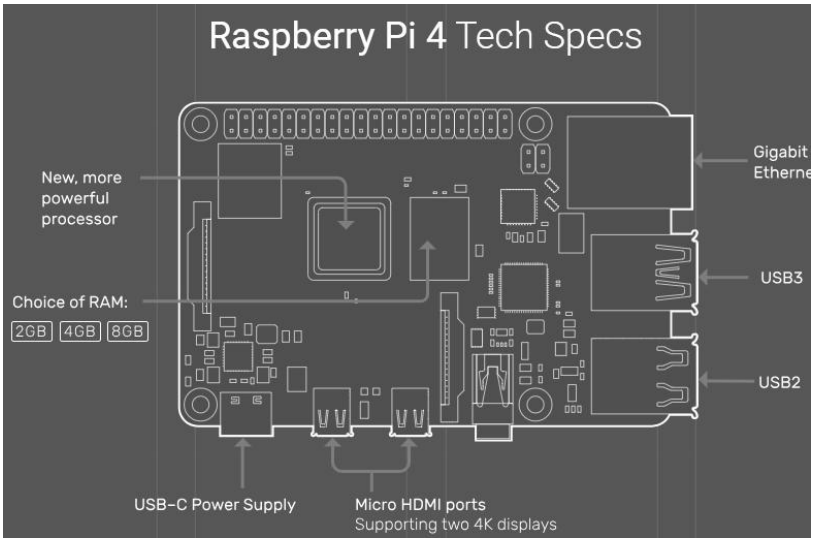


Figure 3.9: Testing of camera using Raspberry Pi [34]

**Specifications of Raspberry Pi:**

Raspberry Pi 4 has been launched in two models over the year. The latest one by the time this project was initiated was model B of Raspberry Pi 4. It entails a 64-bit

quad-core processor that has excellent performance distinguishing it from the previously launched processors performance-wise.

- Broadcom BCM2711 chip
- Cortex -A72 processor
- 8GB RAM
- Standard 40-pin GPIO header
- micro-HDMI ports (2)
- DSI display port.
- CSI camera port.
- Micro-SD card slot.
- GPIO Header: 5 Volts DC
- Power input (C type connector: 5 Volts DC)
- PoE enabled (Power over Ethernet)
- Ambient Temperatures to operate: 0ºC-50ºC

### 3.8.1 Details of Camera:

Raspberry has launched a series of cameras from 2012 onwards. In 2023, camera module 3 was released, a High-Quality (HQ) camera with a 12-megapixel resolution. HQ camera has no infrared versions. Camera Module 3 has the added advantage of a wide field of view for each of its variants. All Raspberry Pi cameras are capable of taking high-resolution photos, along with full HD 1080p video, and can be controlled through a program.

Figure 3.10: Picamera3 module used in this project [39]

**Specifications of Camera:**

- A stacked CMOS 12-megapixel Sony IMX708 with back illumination image sensor.
- Camera Module 3 NoIR Wide
- High signal-to-noise ratio (SNR)
- Enables HDR mode
- CSI serial output port (2x)
- Resolution: 11.9 megapixels
- Size of sensor: 7.4mm
- Pixel size: 1.4µm × 1.4µm
- Field of view: 120 degrees
- Common video modes: 1080p50, 720p100, 480p120
- Dimensions: 25 × 24 × 11.5mm

Figure 3.11: Quality of picture tested using Raspberry pi

# CHAPTER 4

# RESULTS AND DISCUSSIONS

## 4.1 Training And Validation:

Initially, all the models are trained in batch sizes of 32 through 50 epochs. The number of samples utilized in one repetition of the training process is known as the batch size before the model's parameters are updated. Choosing a high number for batch size may not learn the data of the model as proficiently as possible while using a smaller number may consume a lot of time while training but ensure better learning. Larger batch sizes require more memory to store hence smaller the batch size better the generalized performance. The optimizing algorithm we have used in our model is Adam (Adaptive Moment Estimation). This algorithm helps in learning the data efficiently and allows fast convergence to occur as the learning rate is adjusted automatically by analyzing the historical gradients of each parameter. Adam has shown remarkable performance on large datasets with a huge number of parameters

making it suitable for tasks like computer vision and object detection. All models were trained with the help of online available GPUs of Kaggle.

- **Accuracy and Loss:**

In general, one distinguishes between training and validation loss. The training graph (plotted in blue) indicates if the model is learning, while the validation graph (plotted in orange) indicates if the model can generalize well on new data. One would like to observe that the error loss decreases steadily over time, while accuracy rises. This suggests that the model is still learning from the training data and has not yet overfitted. When the loss graph plateaus, it suggests that the model's learning potential has been fully realized from the training data. The graphs followed by this text demonstrate the respective model's performance on training and validation data in terms of accuracy and loss.

- **ResNet50**

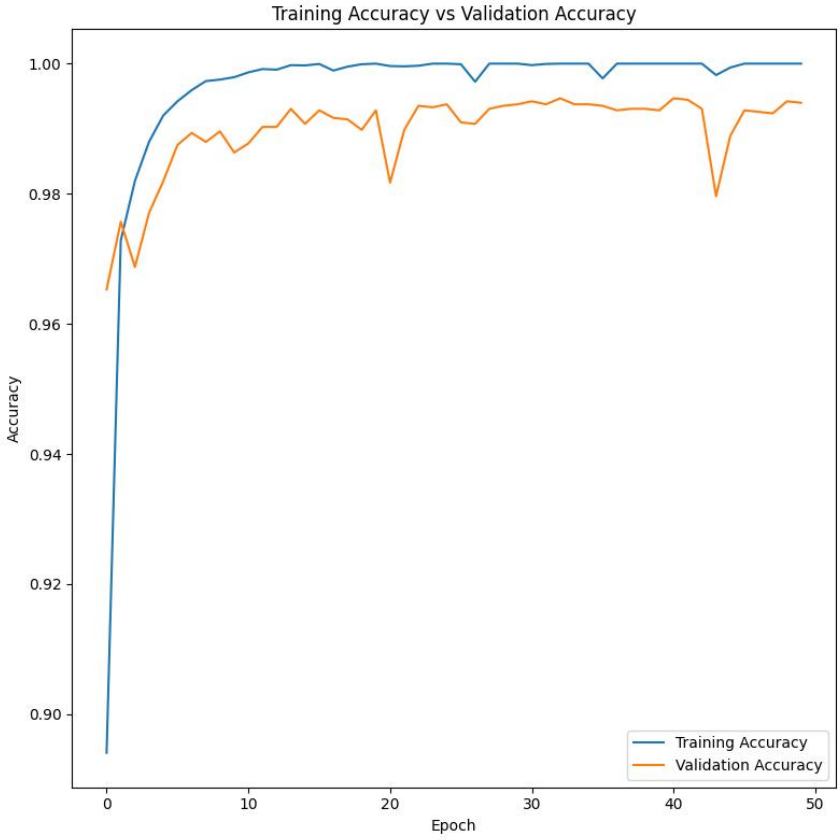


Figure 4.1: Graph of training and validation accuracy of ResNet50

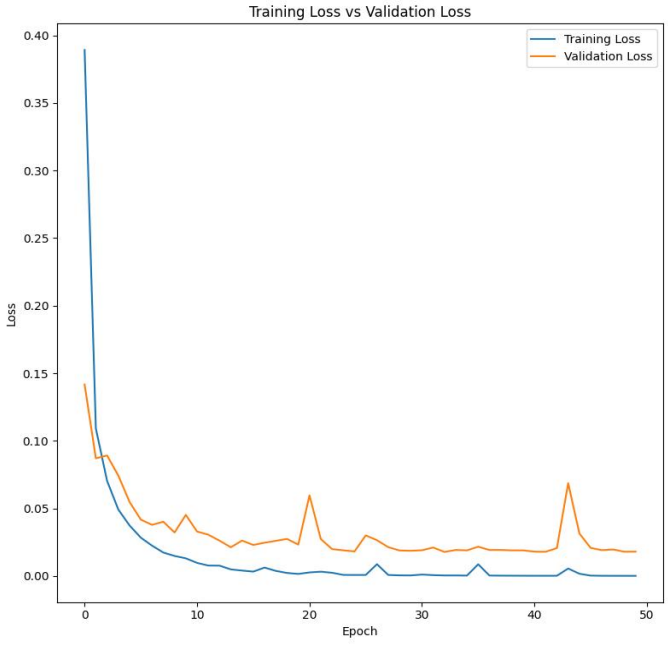


Figure 4.2: Graph of training and validation loss of ResNet50

**ResNet50 graph explanation:**

- Accuracy:

  In figure 4.1, the blue line represents the training accuracy for each epoch. It starts low and quickly rises, stabilizing near 1.0 accuracy after a few epochs. This rapid rise indicates that the model is learning and adapting well to the training data.

  The orange line represents the validation accuracy across each epoch.

  It begins lower and increases, but stabilizes somewhat lower than the training accuracy, reflecting how effectively the model generalizes to new data.

- Loss:

  In figure 4.2, the training loss decreases rapidly in the first few epochs before stabilizing at a relatively low value, demonstrating efficient learning and fitting to the training data. Accordingly, the validation loss begins higher and lowers, but eventually stabilizes at a larger value than the training loss, showing the model's performance on unknown data. There are a few spikes in the validation loss at various intervals, most notably between epochs 20 and 40. These spikes show that the model's performance on the validation set varies, which could be due to noise in the data.

➢ **VGG-16**

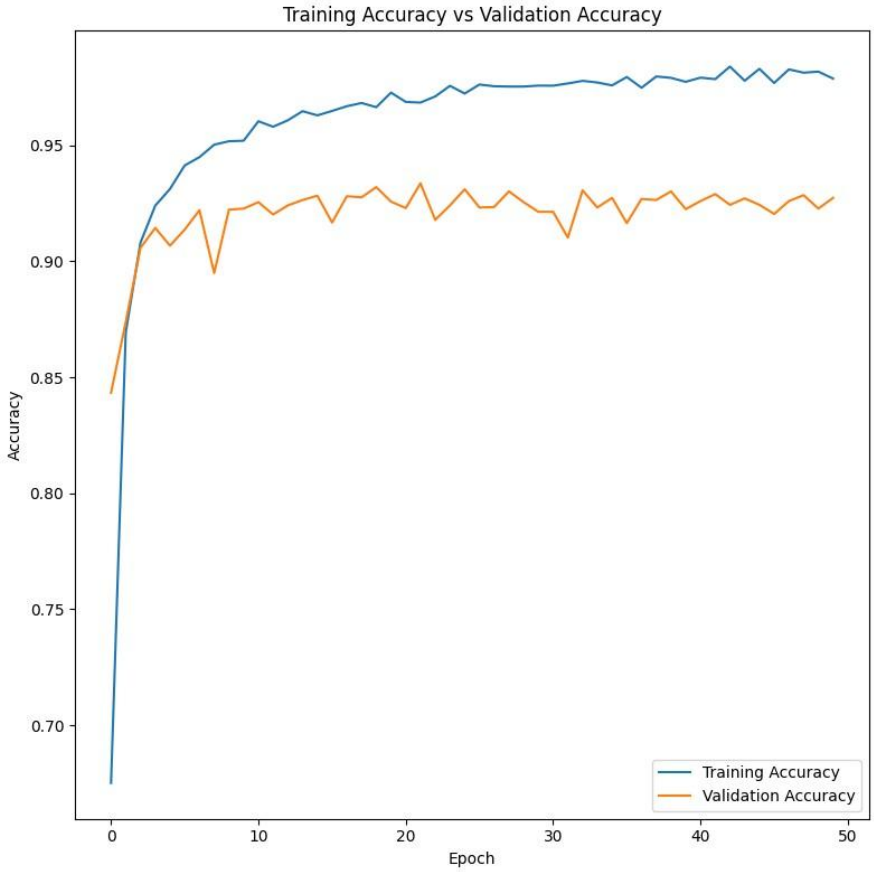


Figure 4.3: Graph of training and validation accuracy of VGG-16


Figure 4.4: Graph of training and validation loss of VGG-16

**VGG-16 graph explanation:**

- Accuracy:

  In figure 4.3, the blue line begins low and gradually increases, eventually stabilizing near 0.96 (or 96%) accuracy after about 10 epochs. This rapid rise indicates that the model is learning and adapting well to the training data.

  The orange line follows a similar pattern as the training accuracy but settles at 0.91 (or 91%), which is somewhat lower than the training accuracy.

  However, the continuous gap between training and validation accuracy is owing to the models' exceptional ability to learn from training data, which can lead to overfitting.

- Loss:

  In figure 4.4, at the start (epoch 0), both the training and validation losses are significant, indicating that the model is not performing effectively.

  The training loss decreases significantly in the first few epochs, demonstrating that the model is rapidly learning and improving its performance on the training dataset. Following the first few epochs, the training loss gradually decreases, indicating that the model is continuing to learn and improve.

- **VGG-19**

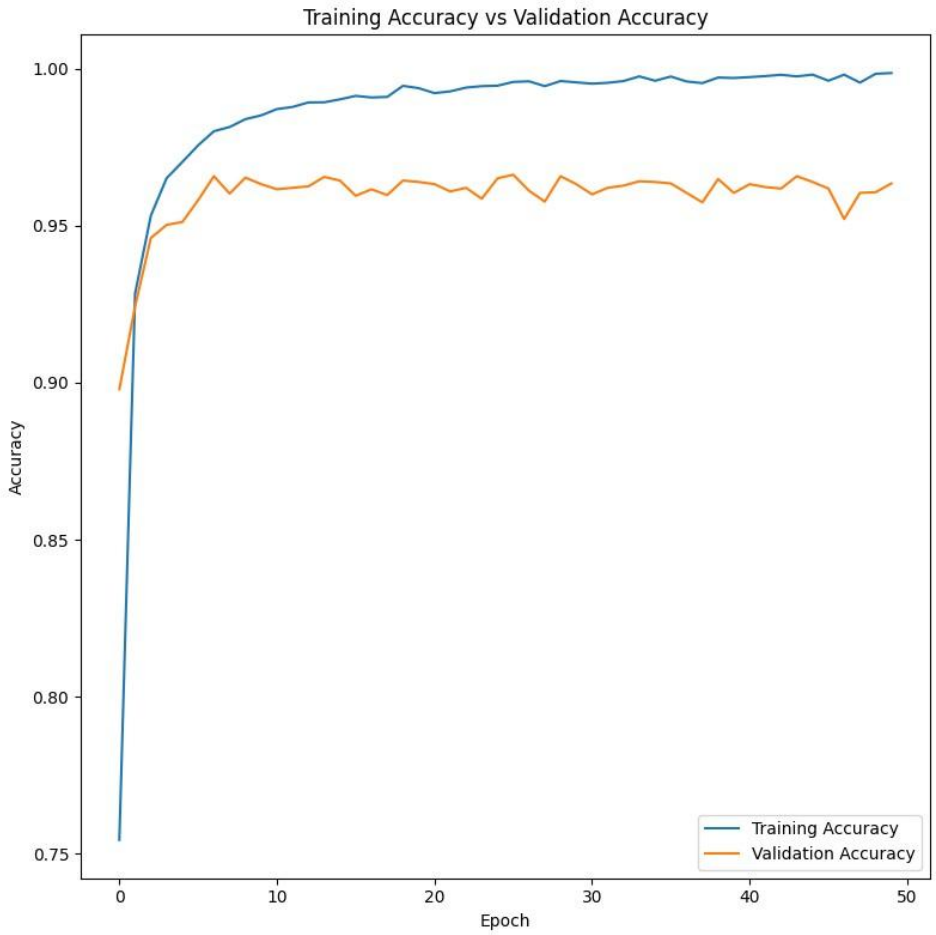


Figure 4.5: Graph of training and validation accuracy of VGG-19

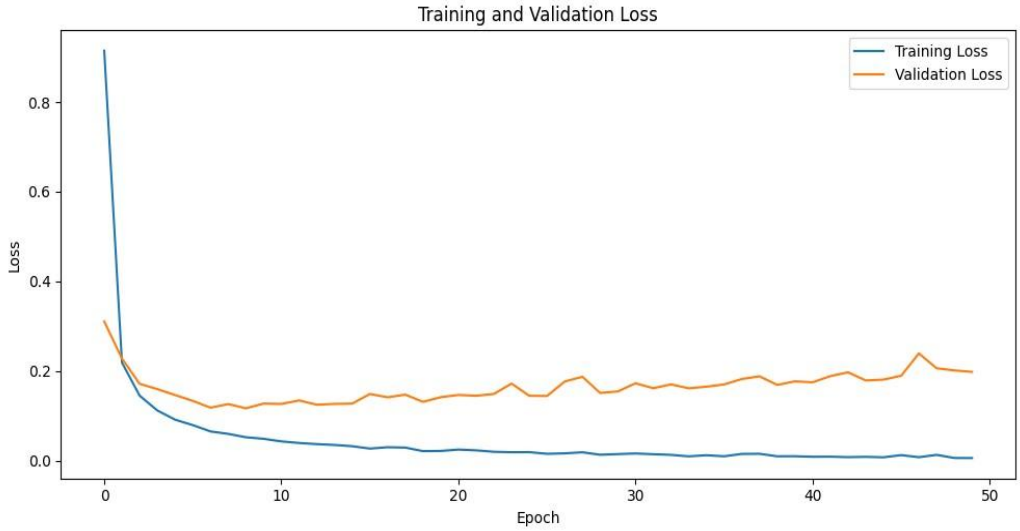


Figure 4.6: Graph of training and validation loss of VGG-19

**VGG-19 graph explanation:**

- Accuracy:

  The training accuracy normally exceeds the validation accuracy. This is a typical phenomenon in machine learning known as the training-validation gap. It suggests that the model might be overfitting the training data.

  VGG19 has a sophisticated architecture with multiple layers. If not adequately regularized, the model might grow too complex, conforming to noise in the training data rather than the underlying patterns.

- Loss:

  Both the training and validation losses appear to be reducing over epochs, indicating that the model is learning from training data. The training loss curve (blue) appears smoother than the validation loss curve (orange). This is to be expected given that the model is optimized for the training data, and the loss on the training set is less unpredictable.

➢ **MobileNet**

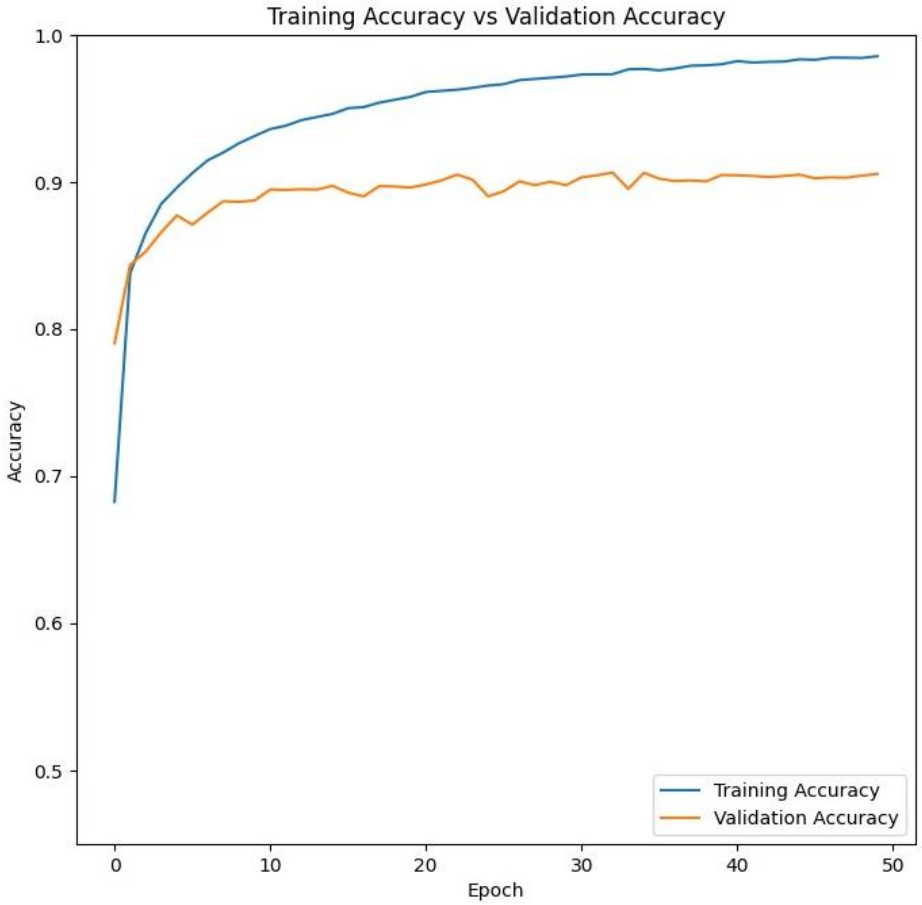


Figure 4.7: Graph of training and validation accuracy of MobileNet

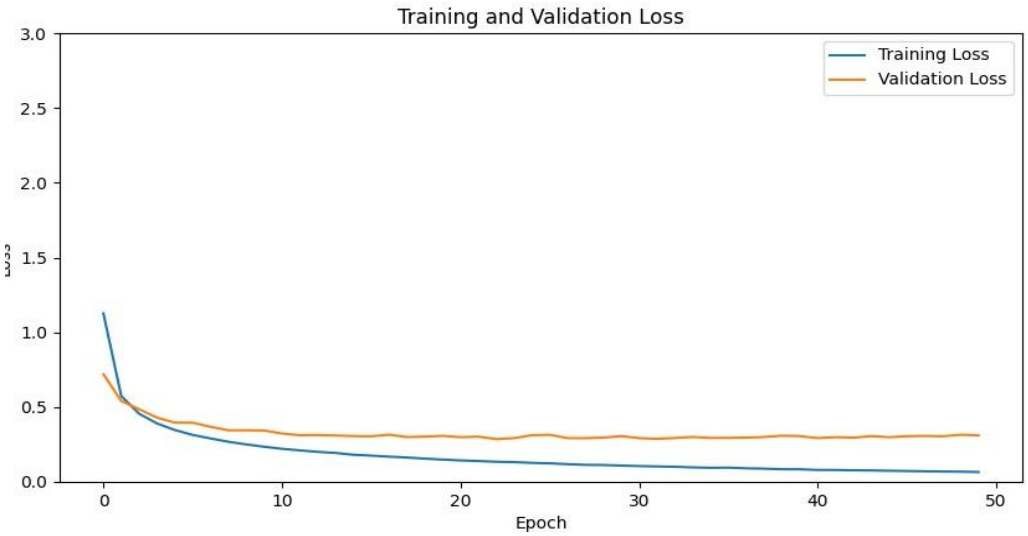


Figure 4.8: Graph of training and validation loss of MobileNet

**MobileNet graph explanation:**

- Accuracy:

  In figure 4.7, after approximately 10-15 epochs, the training accuracy continues to grow and approaches 1, while the validation accuracy stabilizes and hovers between 0.9 and 0.92. This suggests that the model is overfitting: it performs extraordinarily well on training data but does not generalize effectively to unseen validation data.

  This could be due to the architecture; while efficient, the design may limit the model's ability to capture complicated patterns.

  Since MobileNet is shallower than other deep neural networks, it can lead to faster convergence during training.

- Loss:

  In figure 4.8, the training loss decreases progressively, showing that the model is still learning and improving its performance on the training dataset.

  The validation loss diminishes and finally stabilizes, with minimal oscillations around a specific number.

  After the first few epochs, the difference between the training and validation losses is minimal, indicating that the model is not considerably overfitting. the model generalizes effectively to the validation data, which is a good sign for its performance.

➢ **MobileNetV2**


Figure 4.9: Graph of training and validation accuracy of MobileNetV2


Figure 4.10: Graph of training and validation accuracy of MobileNetV2

**MobileNetV2 graph explanation:**

- Accuracy:

  In figure 4.9, the training accuracy(blue) approaches 1 whereas the validation accuracy(orange) hangs between 0.9 and 0.92. The reason for the divergence between the graph of training and validation may very well be due to overfitting. MobileNetV2 depends significantly on depthwise separable convolutions, which are computationally efficient.

  While these convolutions are useful for low power devices, they may hinder the model's ability to learn complicated feature representations.

  MobileNetV2 uses linear bottlenecks in its building pieces to minimize the number of parameters, which may limit the model's capacity to understand non-linear relationships from the data. This constraint in learning complicated nonlinearities may cause the overfitting of the training data.

- Loss:

  In figure 4.10, Both the training loss and the validation loss appear to be reducing over time, indicating that the model is learning from the training data.

  The blue training loss curve appears smoother than the orange validation loss curve. This is expected because the model is optimized for the training data, making the loss on the training set less unpredictable.

**Conclusion:**

Among all the models deployed, ResNet50 model exhibits remarkable accuracy on both training and validation data, signifying exceptional performance. However, the negligible difference between training and validation accuracy shows some overfitting, but not much, as validation accuracy remains high and steady.

- **Classification Results**

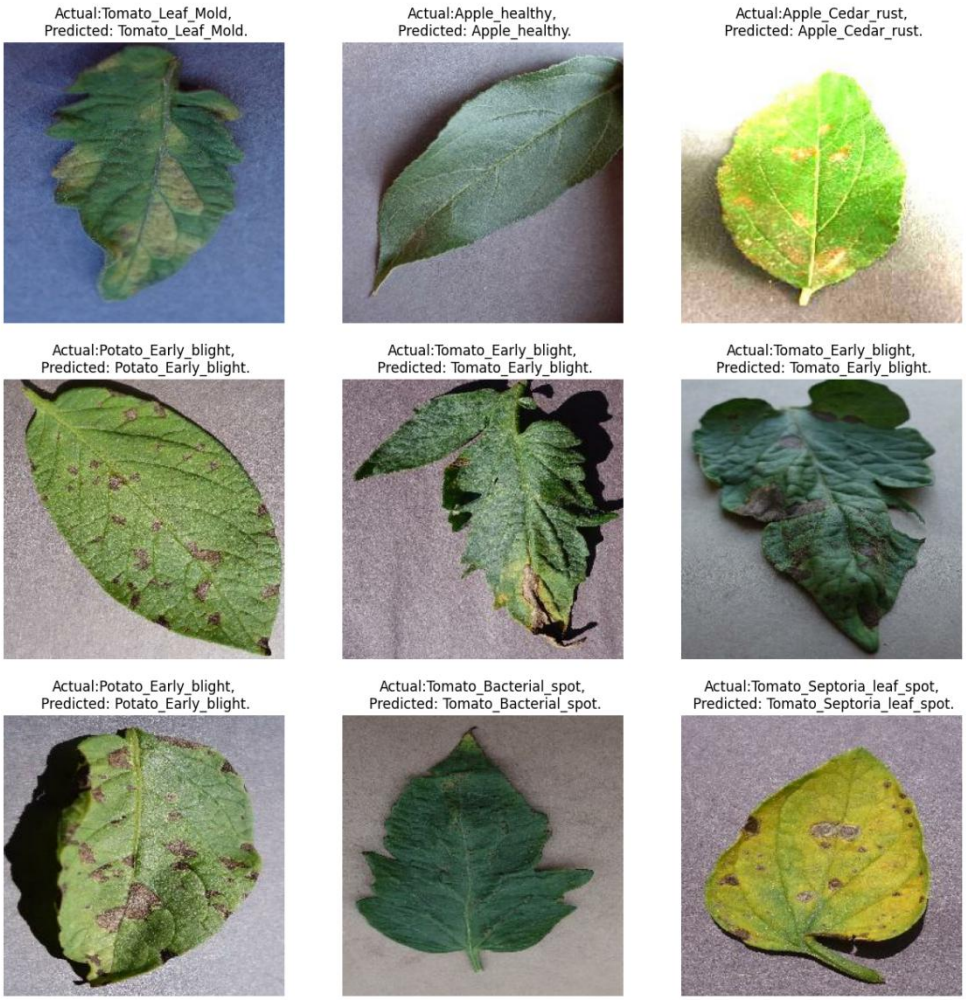


Figure 4.11: Classification results of ResNet50

The figure above demonstrates the classification results of 9 randomly taken samples from the test data that **ResNet50** was able to predict accurately. The accuracy of the results demonstrates the ability of our trained model to detect diseases. Testing accuracy achieved by ResNet50 is 98.61%.

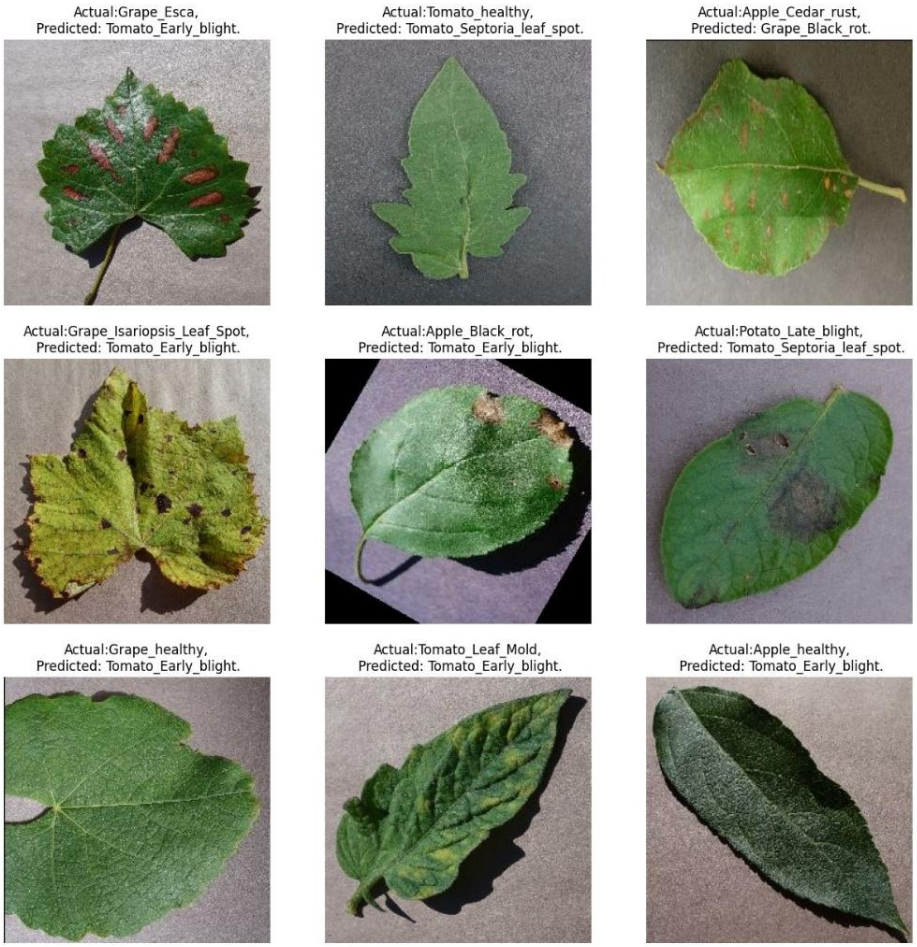


Figure 4.12: Classification results of VGG-16

The figure above demonstrates the classification results of 9 randomly taken samples from the test data that **VGG-16** was able to predict accurately. The accuracy of the results demonstrates the ability of our trained model to detect diseases. Testing accuracy achieved by VGG-16 is 95.73%.

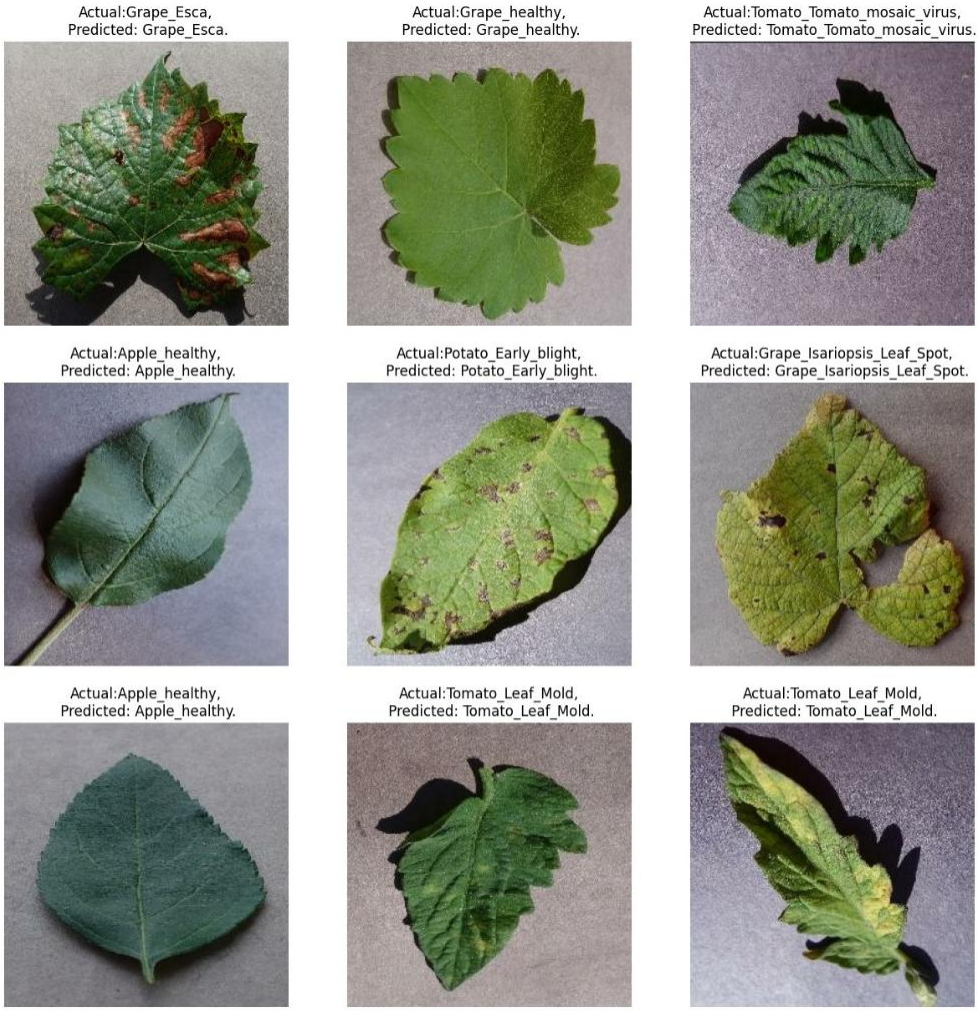


Figure 4.13: Classification results of VGG-19

The figure above demonstrates the classification results of 9 randomly taken samples from the test data that **VGG-19** was able to predict accurately. The accuracy of the results demonstrates the ability of our trained model to detect diseases. Testing accuracy achieved by VGG-19 is 91.67%.

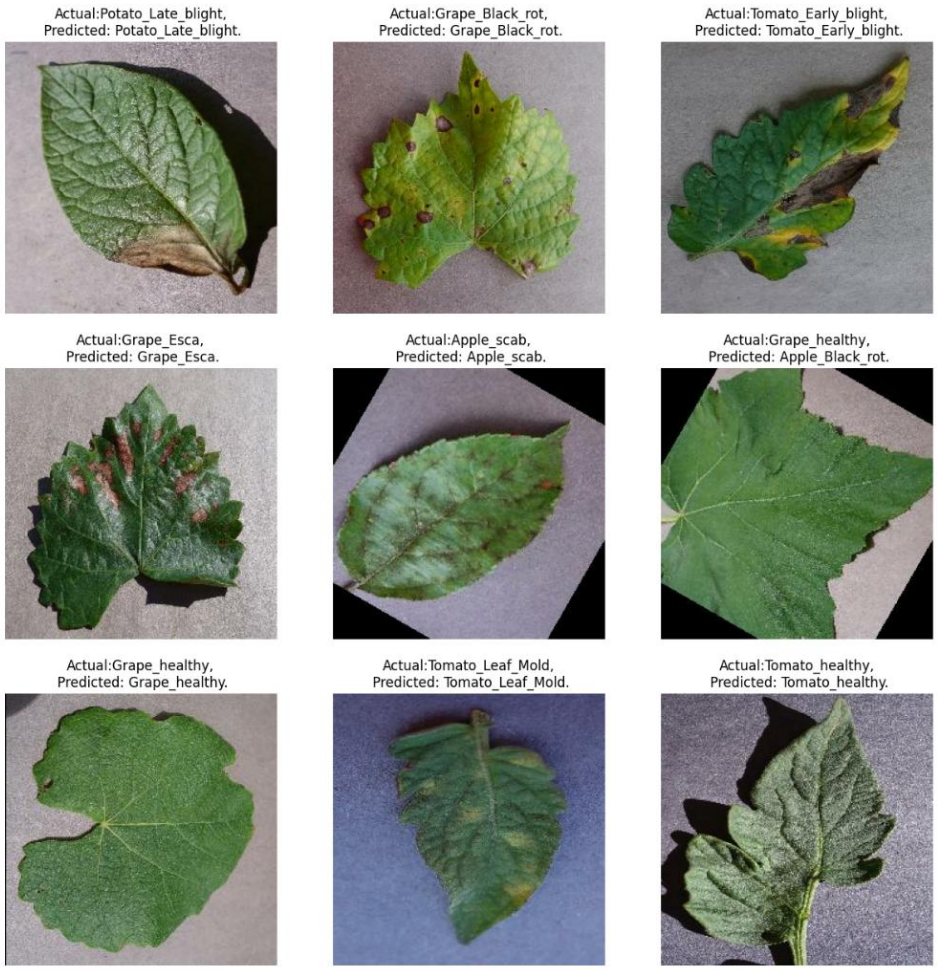


Figure 4.14: Classification results of MobileNet

The figure above demonstrates the classification results of 9 randomly taken samples from the test data that **MobileNet** was able to predict accurately. The accuracy of the results demonstrates the ability of our trained model to detect diseases. Testing accuracy achieved by MobileNet is 90.66%.

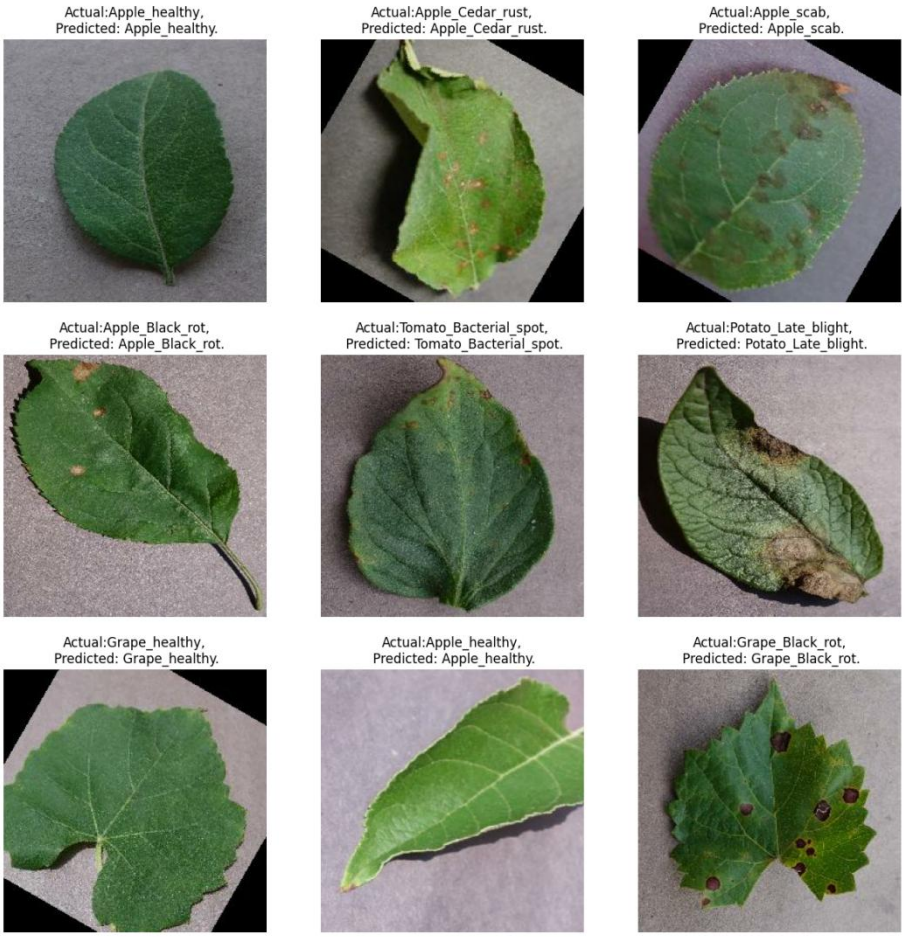


Figure 4.15: Classification results of MobileNetV2

The figure above demonstrates the classification results of 9 randomly taken samples from the test data that **MobileNetV2** was able to predict accurately. The accuracy of the results demonstrates the ability of our trained model to detect diseases. Testing accuracy achieved by MobileNetV2 is 91.04%.

## 4.2 Comparative Analysis:

| Algorithm | Epochs | Batch Size | Testing Accuracy (%) | Precision (%) | Recall | F1-Score (%) |
|---|---|---|---|---|---|---|
| ResNet50 | 50 | 32 | 98.61 | 99.03 | 99.02 | 99.05 |
| VGG-16 | 50 | 32 | 95.73 | 91.72 | 92.06 | 92.06 |
| VGG-19 | 50 | 32 | 91.67 | 95.2 | 95.21 | 95.21 |
| MobileNet | 50 | 32 | 90.66 | 90.29 | 90.16 | 90.16 |
| MobileNetV2 | 50 | 32 | 91.04 | 82.6 | 92.57 | 92.57 |

Table 4.1: Results of our trained model.

Earlier in the report, hyperparameters implemented upon our trained models were discussed, now we move on to the results of our models trained on testing datasets.
As summarised in the table above, ResNet50 has performed extraordinarily well on testing data with an accuracy of 98.61%, trailed by VGG-19 and VGG-16, attaining an accuracy of 95.73% and 91.67%, respectively. Although these models have performed well and have reasonable accuracy scores but due to the trade-off of their model size, complexity, computational resources, and inference time, we will not be selecting either of these architectures to deploy on our end device due to its limited computational power and capacity. Instead, we will be using one of the MobileNet versions due to its certain advantages such as less inference time, small model size, low computational requirement, and its ability to run on low-power devices.

CHAPTER 5

# CONCLUSIONS AND RECOMMENDATIONS

In conclusion, ResNet50 beats VGG16, VGG19, MobileNet, and MobileNetV2 in terms of accuracy, generalization, and stability thanks to its more complex design and creative usage of residual connections. While MobileNet versions have advantages in resource-constrained contexts and VGG16 has fewer layers and a simple structure, ResNet50 remains the best option for applications requiring precision and strong performance. ResNet50's blend of depth, efficiency, and scalability makes it a versatile and strong model for a variety of computer vision problems. The development of an AI-Based Plant Monitoring System for Disease Identification using CDPR serves as a competent solution for the threats exposed to the crops in the domain of plant health management. The background analysis revealed the inadequacies of traditional methods in timely disease detection and the potential of AI, computer vision, and image recognition to revolutionize this aspect of agriculture.

## REFERENCES


[1] Ahmad, Aanis, Dharmendra Saraswat, and Aly El Gamal. "A survey on using deep learning techniques for plant disease diagnosis and recommendations for development of appropriate tools." *Smart Agricultural Technology* 3 (2023): 100083.

[2] https://agrihunt.com/articles/basics-of-agriculture/importance-of-agriculture-in-pakistans-economy-and-development/

[3] Jiang, Peng, et al. "Real-time detection of apple leaf diseases using deep learning approach based on improved convolutional neural networks." *IEEE Access* 7 (2019): 59069-59080.

[4] Sachdeva, Guneet, Preeti Singh, and Pardeep Kaur. "Plant leaf disease classification using deep Convolutional neural network with Bayesian learning." *Materials Today: Proceedings* 45 (2021): 5584-5590.

[5] Fuglie, K., Jelliffe, J., & Morgan, S. (2021). Slowing productivity reduces growth in global agricultural output. *Amber waves: The economics of food, farming, natural resources, and rural America*, *2021*(1490-2022-273).

[6] Ricciardi, Vincent, Navin Ramankutty, Zia Mehrabi, Larissa Jarvis, and Brenton Chookolingo. "How much of the world's food do smallholders produce?." *Global food security* 17 (2018): 64-72.

[7] Khan, Rehan Ullah, Khalil Khan, Waleed Albattah, and Ali Mustafa Qamar. "Image-based detection of plant diseases: from classical machine learning to deep learning journey." *Wireless Communications and Mobile Computing* 2021 (2021): 1-13.

[8] Abbas, A., Mubeen, M., Younus, W., Shakeel, Q., Iftikhar, Y., Bashir, S., ... & Hussain, A. (2023). Plant Diseases and Pests, Growing Threats to Food Security of Gilgit-Baltistan, Pakistan. Sarhad Journal of Agriculture, 39(2).

[9] Vafapour, Reza, et al. "On the applicative workspace and the mechanism of an agriculture 3-dof 4-cable-driven robot." *Int. J. Robot. Autom* 1 (2021): 36.

[10] Radojicic, J., D. Surdilovic, and J. Krüger. "Application challenges of large-scale wire robots in agricultural plants." *IFAC Proceedings Volumes* 46, no. 4 (2013): 77-82.

[11] https://uaf.edu.pk/oubm/Files/Digest/Zarai%20Digest%20201718/13-%20Oct-Dec-%202020/05.%20English%20Articles.pdf

[12] Sultan, M. S., Khan, M. A., Khan, H., & Ahmad, B. (2022). Pathways to strengthening capabilities: A case for the adoption of climate-smart agriculture in Pakistan. APN Science Bulletin, 12(1), 171-183

[13] Phiphiphatphaisit, Sirawan, and Olarik Surinta. "Food image classification with improved MobileNet architecture and data augmentation." In Proceedings of the 3rd International Conference on Information Science and Systems, pp. 51-56. 2020.

[14] Chen, Zhaoyi, Ruhui Wu, Yiyan Lin, Chuyu Li, Siyu Chen, Zhineng Yuan, Shiwei Chen, and Xiangjun Zou. "Plant disease recognition model based on improved YOLOv5." Agronomy 12, no. 2 (2022): 365.

[15] Mohanty, Sharada P., David P. Hughes, and Marcel Salathé. "Using deep learning for image-based plant disease detection." Frontiers in plant science 7 (2016): 1419.

[16] Ramesh, S., Hebbar, R., Niveditha, M., Pooja, R., Shashank, N., & Vinod, P. V. (2018, April). Plant disease detection using machine learning. In 2018 International conference on design innovations for 3Cs compute communicate control (ICDI3C) (pp. 41-45). IEEE.

[17] https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758

[18] Bi, C., Wang, J., Duan, Y., Fu, B., Kang, J. R., & Shi, Y. (2022). MobileNet based apple leaf diseases identification. *Mobile Networks and Applications*, 1-9.

[19] Nguyen, T.H., Nguyen, T.N. and Ngo, B.V., 2022. A VGG-19 model with transfer learning and image segmentation for classification of tomato leaf disease. AgriEngineering, 4(4), pp.871-887.

[20] Kadam, Kalyani Dhananjay, Swati Ahirrao, and Ketan Kotecha. "Efficient approach towards detection and identification of copy move and image splicing forgeries using mask R-CNN with MobileNet V1." Computational Intelligence and Neuroscience 2022 (2022).

[21] He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[22] Zake, Z., Chaumette, F., Pedemonte, N., & Caro, S. (2019). Vision-based control and stability analysis of a cable-driven parallel robot. IEEE Robotics and Automation Letters, 4(2), 1029-1036.

[23] Isuyama, V. K., & de Carvalho Albertini, B. (2021, July). Comparison of Convolutional Neural Network Models for Mobile Devices. In Anais do XX Workshop em Desempenho de Sistemas Computacionais e de Comunicação (pp. 73-83). SBC.

[24] Das, Debasish, Mahinderpal Singh, Sarthak Swaroop Mohanty, and S. Chakravarty. "Leaf disease detection using support vector machine." In 2020 International Conference on Communication and Signal Processing (ICCSP), pp. 1036-1040. IEEE, 2020.

[25] Thakur, Poornima Singh, Tanuja Sheorey, and Aparajita Ojha. "VGG-ICNN: A Lightweight CNN model for crop disease identification." *Multimedia Tools and Applications* 82, no. 1 (2023): 497-520.

[26] Khalid, Mahnoor, et al. "Real-Time Plant Health Detection Using Deep Convolutional Neural Networks." Agriculture 13.2 (2023): 510.

[27] Kumar, Vinod, Hritik Arora, and Jatin Sisodia. "Resnet-based approach for detection and classification of plant leaf diseases." In 2020 international conference on electronics and sustainable communication systems (ICESC), pp. 495-502. IEEE, 2020.

[28] Shrestha, Garima, Majolica Das, and Naiwrita Dey. "Plant disease detection using CNN." 2020 IEEE applied signal processing conference (ASPCON). IEEE, 2020.

[29] Ramesh, S., Hebbar, R., Niveditha, M., Pooja, R., Shashank, N., & Vinod, P. V. (2018, April). Plant disease detection using machine learning. In 2018 International conference on design innovations for 3Cs compute communicate control (ICDI3C) (pp. 41-45). IEEE.

[30] Hassan, Sk Mahmudul, et al. "Identification of plant-leaf diseases using CNN and transfer-learning approach." Electronics 10.12 (2021): 1388.

[31] Chen, Junde, Defu Zhang, Md Suzauddola, and Adnan Zeb. "Identifying crop diseases using attention embedded MobileNet-V2 model." *Applied Soft Computing* 113 (2021): 107901.

[32] Ashqar, Belal AM, and Samy S. Abu-Naser. "Image-based tomato leaves diseases detection using deep learning." (2018).

[33] Mukti, Ishrat Zahan, and Dipayan Biswas. "Transfer learning based plant diseases detection using ResNet50." In *2019 4th International conference on electrical information and communication technology (EICT)*, pp. 1-6. IEEE, 2019.

[34] https://www.raspberrypi.com/

[35] https://www.igyaan.in/99736/drones-agriculture/

[36] Afif, Mouna, Riadh Ayachi, Yahia Said, and Mohamed Atri. "A transfer learning approach for indoor object identification." *SN Computer Science* 2, no. 6 (2021): 424.

[37] Downton, Jonathan E., Olivia Collet, Daniel P. Hampson, and Tanya Colwell. "Theory-guided data science-based reservoir prediction of a North Sea oil field." The Leading Edge 39, no. 10 (2020): 742-750.

[38] Kadam, Kalyani Dhananjay, Swati Ahirrao, and Ketan Kotecha. "Efficient approach towards detection and identification of copy move and image splicing forgeries using mask R-CNN with MobileNet V1." Computational Intelligence and Neuroscience 2022 (2022).

[39] https://www.elektormagazine.com/news/raspberry-pi-camera module-3-comes-in-4-variants-features-autofocus

## APPENDICES

APPENDIX B: CODE

```
# ALEXNET< SQUEEZENET, densenet, espnet,
import matplotlib.pyplot as plt
import numpy as np
import PIL
import tensorflow as tf
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.applications import MobileNet
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

# Load the dataset
dataset = tf.keras.preprocessing.image_dataset_from_pc(
    "/kaggle/input/mini-plant-village",
    shuffle=True,
    image_size=(256, 256),
    batch_size=32
)

# Get names of classes
names_of_classes = dataset.class_types
# Display sample images from the dataset
plot.figure(figsize=(10, 10))
for image_batch, label_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[label_batch[i]])

# Define the model
input_shape = (256, 256, 3)
num_classes = len(class_names)
channels = 3
```

```
epochs = 50

def dataset_from_partitions_tf(dataset, train_split=0.75, val_split=0.15,
test_split=0.10, shuffle=10000):
    # Compute the size of each partition
    dataset_size = len(dataset)
    train_size = int(train_split * dataset_size)
    val_size = int(val_split * dataset_size)
    test_size = int(test_split * dataset_size)

    # Create partitions
    train_ds = dataset.take(train_size)
    remaining_ds = dataset.skip(train_size)
    test_ds = remaining_ds.skip(val_size)

    return train_ds, val_ds, test_ds

train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)

train_ds = train_ds.cache().shuffle(1000)
val_ds = val_ds.cache().shuffle(1000)
test_ds = test_ds.cache().shuffle(1000)

resize_rescale = tf.keras.Sequential([
    layers.experimental.preprocessing.Rescaling(1.0/255)
])

data_augmentation  = tf.keras.Sequential([
    layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
    layers.experimental.preprocessing.RandomRotation(0.2),
])
# Load the MobileNet base model without pretrained weights
base_model = MobileNet(
```

```
    input_shape=input_shape,
    include_top=False,
    weights="imagenet"
)

# Freeze the base model
base_model.trainable = False

# Create the model
model = Sequential([
    base_model,
    layers.Dense(num_classes, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

# Train the model
history = model.fit(train_ds,
                    epochs=epochs,
                    validation_data=val_ds)
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
plt.figure(figsize=(8,8))
# plt.subplot(1,2,1)
plt.plot(range(epochs),acc, label='Training Accuracy')
plt.ylim([0, 1.1])
yticks = [i/10 for i in range(11)]
```

```
plt.legend(loc='lower right')
plt.figure(figsize=(8,8))
# plt.subplot(1,2,1)
plt.plot(range(epochs),loss, label='Training Loss')
plt.plot(range(epochs),val_loss, label='Validation Loss')
plt.ylim([-0.5, 10])
plt.legend(loc='upper right')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title("Training Loss vs Validation Loss")
plt.savefig("loss.png")
plt.figure(figsize=(8,8))
plt.subplot(1,2,1)
plt.plot(range(epochs),
plt.figure(figsize=(8,8))
# plt.subplot(1,2,1)
plt.plot(range(epochs),loss, label='Training Loss')
plt.plot(range(epochs),val_loss, label='Validation Loss')
plt.ylim([-0.5, 10])
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title("Training Loss vs Validation Loss")
plt.savefig("loss.png")

def prediticions(model,img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array,0)

    prediction = model.predictions (img_array)
    confidence = round(100*(np.max(prediction[0])),2)
    return predicted_class, confidence

plt.ylim([0, 1.1])
```

```
plt.legend(loc='lower right')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title("Training Accuracy vs Validation Accuracy")
for images, labels in test_desktop.take(1):
    for i in range(9):
        ax = plt.subplot(3,3,i+1)
        plt.imshow(images[i].numpy().astype("uint8"))
        actual_class = class_names[labels[i]]
        plt.title(f"Actual:{actual_class},\n Predicted: {predicted_class}.")
        plt.axis("off")
        plt.savefig("Results")

plt.subplot(1,2,2)
plt.plot(range(epochs),loss, label='Training Loss')
plt.plot(range(epochs),val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title("Training Loss vs Validation Loss")
plt.savefig("combined.png")
test_accuracy = model.evaluate(test_ds)
    print (f"Test Accuracy  :{i+1} {test_accuracy}")
```