B. Tech. Project Part-II (COC4990)

Report

on

APPLE LEAVES DISEASE PREDICTION USING COMPUTER VISION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF

Bachelor of Technology

IN

COMPUTER ENGINEERING

BY

Suhail Bashir
19COB502

Shamir Hassan
19COB513

Under the Guidance of

Prof. Rashid Ali

Department of Computer Engineering

Zakir Husain College of Engineering & Technology
Aligarh Muslim University
Aligarh (India)-202002, India

Declaration

We hereby declare that our work, which has been presented in this project, Entitled "**APPLE LEAVES DISEASE PREDICTION USING COMPUTER VISION"** submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Engineering, Zakir Hussain College of Engineering and Technology, Aligarh Muslim University, Aligarh is an authentic record of our own work carried out under the able guidance of Prof. Rashid Ali, Department of Computer Engineering, Zakir Hussain College of Engineering and Technology, AMU Aligarh.

Suhail Bashir
19COB502

Shamir Hassan
19COB513

Date: 12 May, 2023
Place: Aligarh, U.P.

Certificate

On basis of the declaration given by the candidates, this is to certify that the Project Report entitled "**APPLE LEAVES DISEASE PREDICTION USING COMPUTER VISION**", being submitted by Suhail Bashir and Shamir Hassan, in partial fulfillment of the requirements for the passing of the Major Project Course (COC4990), during the session 2022-23, in the Department of Computer Engineering, Zakir Husain College of Engineering and Technology, Aligarh Muslim University, is a record of candidates' own work carried out by them under my supervision and guidance.

Prof. Rashid Ali
Department of Computer Engineering
ZHCET, AMU, Aligarh

Date: 12 May, 2023
Place: Aligarh, U.P.

Table of Contents

ABSTRACT

Apples are one of the most important temperate fruit crops in the world. The degradation in the quality and productivity of apples causes huge economic loss to the apple industry every year. Foliar (leaf) diseases and pests are one of the major reasons for the bad quality and lower productivity of apples. Early detection of apple diseases can help in controlling the spread of infections and ensure better productivity. However, early diagnosis and identification of diseases is challenging due to many factors like, lack of expert manforce, presence of multiple symptoms on the same leaf, non-homogeneous background, differences in leaf color due to age of infected cells, varying disease spot sizes etc. In this project we have proposed a very lightweight classification model that can classify different diseases of apple leaves without any human intervention. We have also compared the results of State Of The Art Convolution Neural Network (CNN) models and our lightweight architecture.

We have used a dataset from Kaggle, which contains 19k images of leaves which are almost equally divided into 4 classes namely Apple_scab, Apple_Black_rot, Apple_cedar_rust and Apple_healthy. The dataset contains high-quality RGB images of apple foliar diseases, including a large expert-annotated disease dataset. This dataset reflects real field scenarios by representing non-homogeneous backgrounds of leaf images taken at different maturity stages and at different times of day under different focal camera settings.

Keywords: Object detection algorithms, Deep Learning, Apple leaf diseases, Convolutional Neural Networks.

Acknowledgement

First and foremost, praises and thanks to Allah, the creator of heavens and the earth, we would like to convey our heartfelt thanks to Prof. Rashid Ali for his inspiration and encouragement in completing this job successfully. He has always been friendly, attentive, responsible, and considerate. All the highs and lows of the journey were helpful. He dedicated a significant amount of time to our studies, providing useful comments and going to great lengths to see us through. We have learned a lot from him and will be eternally grateful to him. He committed a significant amount of effort and attention to the project's effective completion.

Date: 12/05/2023
Place: Aligarh
Shamir Hassan, Suhail Bashir

List of figures

List of tables

Chapter 1
Introduction

1.1 Overview

The cultivation of apples is an important part of the agricultural sector and contributes significantly to the economy in a number of ways. However, apple trees are susceptible to a variety of diseases, some of which affect the leaves, which can significantly lower both the yield and the quality of the crop. Traditional methods of plant disease detection can be laborious, and time-consuming despite the fact that early detection and accurate diagnosis of plant diseases are essential for the effective control of these diseases. In recent years, computer vision has emerged as a potentially useful technology for automatically detecting and categorizing a wide variety of plant diseases. This technology involves the use of machine learning algorithms to analyze images of plants and identify disease symptoms.

1.2 Background and Motivation

Apple fruits are one of the most widely grown fruits in the world and are cultivated worldwide. Given its high remedial and nutritive values Apple fruit is one of the most productive fruits. In India, the valley of Kashmir holds the maximum share of Apple production with more than 75% of total apple production. Currently, around 160,000 ha of land in the Valley is under apple cultivation with an annual productivity of around 180,000 MTs (Directorate of Horticulture, 2021), in which an enormous proportion is exported to different parts of the world. However, diseases and pests cause huge economic loss to the apple industry every year. Pests and diseases not only reduce the fruit production but adversely affect the fruit quality as well. Thus, in order to improve the quality and quantity of crops, it is imperative to protect them from diseases. Identifying a disease correctly when it first appears and taking timely action is a crucial step for efficient disease management. Timely and accurate diagnosis of a disease attack is crucial for early and targeted application of curative measures and stopping the disease from spreading. Appropriate knowledge of a

disease would help the farmer to take proper precautionary measures or apply just the right amounts of pesticides, thereby getting both economic and environmental benefits. Farmers relied on planting experience and expert guidance to diagnose diseases in the early days.

However, to achieve this, farmers must continuously monitor their crops and remain in touch with the experts for any help from time to time. The experts must be proficient and should have extensive knowledge of various diseases, their symptoms and treatment. Such methods are labor intensive and time-consuming. Moreover, some of the diseases can only be diagnosed by experts and only a few samples can be examined at any time. A valid alternative to such a labor intensive and costly task will be an automatic system that can detect diseases at an early stage and provide appropriate treatments and recommendations in time. Such a system will be a savior for the farmers and can significantly increase crop production on a sustainable basis.

1.3 Objectives

This project aims to create a computer vision-based apple leaf disease prediction system. More specifically the objectives of the project are:

1. Collect a dataset of apple leaf image: The first objective is to collect a large and diverse dataset of high-quality images of apple leaves. This dataset should include images of healthy leaves as well as leaves affected by different diseases.
2. Develop and evaluate machine learning models: Once the dataset is collected, the next objective is to develop machine learning models that can accurately detect and classify diseases in apple leaves. This involves training and fine tuning various models using deep learning techniques, such as ResNet-50, VGG-16, Xception, etc.
3. Develop a lightweight model from scratch: After training the large pre-trained models, the next objective is to train a lightweight model from scratch with very few parameters as compared to pre-trained models.
4. Compare the performance of different models: After developing the models, the third objective is to compare their performance and evaluate their accuracy, precision, and

recall. This will involve testing all the models on a separate test dataset of apple leaf images that were not used during the training phase, and validation phase.

1.4 Organization of work

The remainder of this report is organized as follows. Chapter 2 provides an overview of the relevant literature on computer vision and its applications in agriculture, with a focus on disease detection in plant leaves. Chapter 3 describes the design methodology used to collect and preprocess the dataset of apple leaf images. Chapter 4 provides project Implementation & Results. Chapter 5 presents the results of the experiments conducted to evaluate the proposed system's performance. Finally, Chapter 6 summarizes the findings and conclusions of the project, and suggests future research directions.

Chapter 2
Literature review

2.1 Overview

This chapter provides an overview of the relevant literature on computer vision and its applications in agriculture, with a focus on disease detection in plant leaves. The chapter starts with an introduction to computer vision and its applications in agriculture. It then presents a review of the state-of-the-art methods, techniques, and approaches used for disease detection in plant leaves. The chapter concludes with a discussion of related work in the area of apple leaf disease detection using computer vision.

2.2 State of the Art Methods

In recent years, there have been significant advances in the field of computer vision for plant disease classification. Deep learning-based approaches, in particular, have shown remarkable performance in accurately identifying and classifying diseases on plant leaves. In the case of apple leaf disease classification, several studies have achieved impressive results using various deep learning models and techniques.

The state-of-the-art for apple leaf disease classification using computer vision techniques has been rapidly evolving in recent years. In a study by Ref. [1], they proposed a deep learning-based approach for apple leaf disease classification using a dataset of 5,782 images. They achieved an accuracy of 99.1% using a ResNet-50 architecture, which outperformed previous studies in the field. Another study by Ref. [2] used a dataset of 1,800 apple leaf images and achieved an accuracy of 96.08% using a pre-trained DenseNet-121 model.

More recently, the Vision Transformer (ViT) Ref. [3] model has shown promising results in various computer vision tasks and has been used for plant disease classification Ref. [4].

Therefore, future work could involve exploring the performance of the ViT model for apple leaf disease classification.

2.3 Related Work

The development of machine learning has provided a new approach to crop disease identification. Using traditional machine-learning methods for apple leaf disease identification requires manual extraction of features such as color, shape, and texture of the disease, followed by feature reduction using traditional artificial intelligence algorithms such as Principal Component Analysis (PCA), genetic algorithm (GA), Support Vector Machine (SVM), and K-Nearest Neighbor (KNN), and finally classification and identification of the disease. For example, Ref. [5] extracted the color, shape, and texture features of apple leaf lesions and used SVM to achieve automatic disease recognition. Ref. [6] proposed a feature fusion-based plant disease-recognition method and obtained 96% recognition accuracy on the apple leaf disease dataset. Ref. [7] overcame the difficulty of feature extraction and selection in classical plant disease recognition methods by mapping observed sample points in high-dimensional space to low-dimensional subspace and obtained more than 90% recognition accuracy on the apple leaf disease dataset. However, their dataset was relatively small, the model's generality was rather poor, and the recognition accuracy was not high in scenarios with high noise, such as uneven lighting.

All the approaches discussed above are based on traditional Machine Learning techniques. These techniques are not fully automatic. In fact, one of the major drawbacks of Machine Learning is the need for manual intervention. Machine Learning techniques can work on small datasets but rely on manual hand crafted features designed by experts. Some of the features extensively used in computer vision are Local Binary Patterns (LBP), Complete Local Binary Pattern [8], Histogram of Gradients (HOG), HSV Histogram, Global Color Histogram and Gabor filters [9]. These handcrafted features are usually not robust, computationally intensive due to high dimensions, and are limited in number. Automated feature engineering on the other hand is more efficient and repeatable than manual feature engineering allowing us to build better predictive models faster.

With the development of deep learning techniques, which is a subset of Machine Learning research, which has gained popularity in the recent past. Deep Learning models are a special type of Artificial Neural Networks which learn (multiple levels of) representation by using a hierarchy of multiple layers. The biggest advantage of Deep Learning techniques is that they do not rely on hand-crafted features. Rather, these networks learn especially the success of convolutional neural networks (CNNs) in computer vision, some scholars have started applying CNNs to the apple leaf disease recognition task. Ref. [10] first proposed to use AlexNet and GoogLeNet to identify apple leaf diseases and obtained 97.62% recognition accuracy. Ref. [11] used global average pooling to replace fully connected layers and used an improved Softmax classifier to identify apple leaf diseases, significantly reducing the model's training and recognition time. Ref. [12] enhanced the feature extraction capability of the model by combining XDNet with DenseNet and Xception, and achieved the recognition of five apple leaf diseases with 98.82% recognition accuracy with a small number of parameters. However, the dataset they used contained a relatively homogeneous background, which was ineffective in practical application scenarios. Under natural conditions in the field, complex environments such as weather conditions, brightness changes, occlusions, and other objects in the disease images can adversely affect the performance of the model, which requires the model to give less attention to irrelevant features such as background and more attention to the disease region itself. Thus, ref. [13] proposed a self-attention convolutional neural network and obtained 95.33% and 98.0% recognition accuracy on AES-CD9214 and MK-D2 datasets. Ref. [14] proposed integrating attention into the Efficient-Net network and got 98.92% recognition accuracy on self-built apple leaf disease datasets. However, the attention mechanism they used is different and specialized, which is not general, and using only a single attention mechanism can potentially cause the overfitting of the model. The parameters and FLOPs of their model are relatively large and difficult to deploy on resource-constrained edge devices. So, ref. [15] proposed to use MobileNet for apple leaf disease recognition, which effectively reduced the parameters and FLOPs of the model but only obtained 73.5% recognition accuracy, which obviously could not meet the practical needs.

2.4 Chapter Summary

The chapter starts with an introduction to computer vision and its role in agriculture, followed by an in-depth review of the state-of-the-art methods, techniques, and approaches used for disease detection in plant leaves. The review covered both traditional computer vision-based methods and deep learning-based methods. Overall, Chapter 2 provides a solid foundation for the proposed approach in the thesis, which aims to develop a deep learning-based system for detecting diseases in apple leaves using computer vision.

Chapter 3
Design Methodology

3.1 Overview

This chapter presents the methodology used to develop the proposed system for predicting diseases of apple leaves using computer vision. The chapter starts with dataset specification, followed by a detailed explanation of the approaches used.

3.2 Functionalities

The main functionalities of our work is that we have trained an end-to-end deep learning model on some classes of diseases that are most commonly found. It is going to scan the leaves through a digital camera based device and recognize the common types of diseases that it was trained on.

3.3 Use case diagram

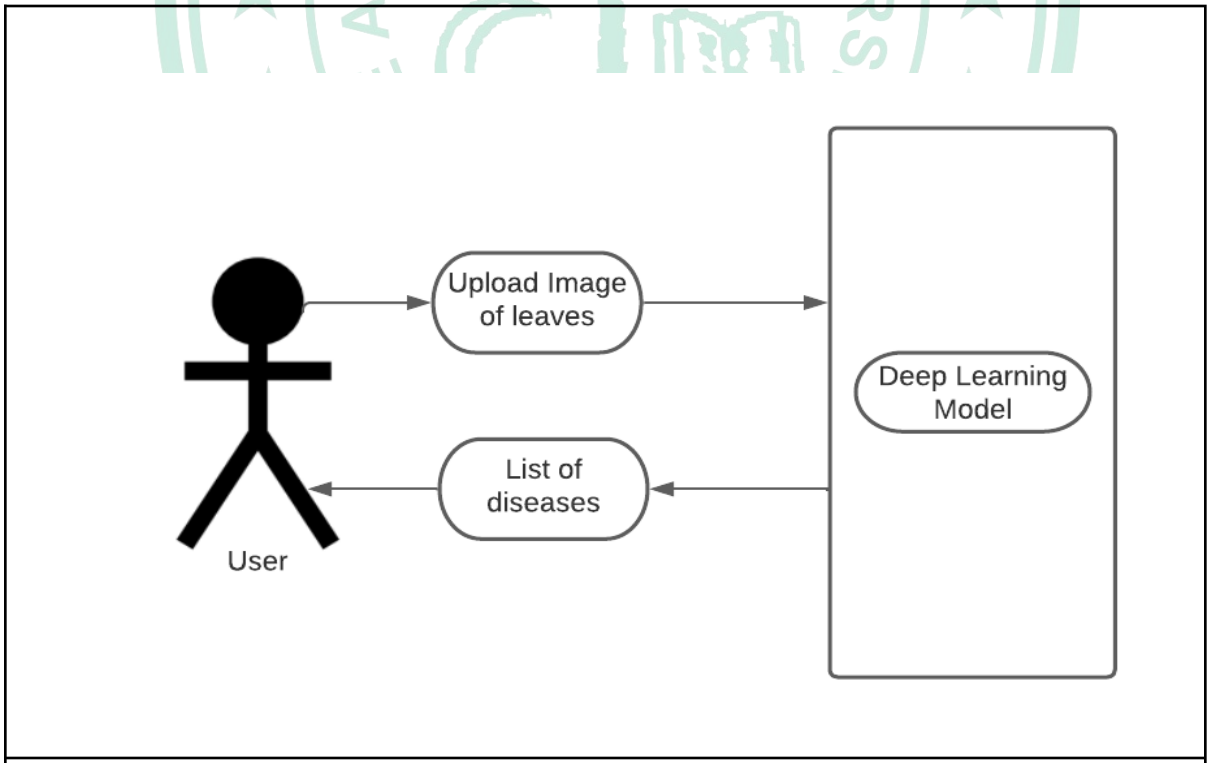

Figure 3.1: Use case diagram

3.4 Functional Requirements

Functional requirements define the internal workings of the software: that is, the technical details, data manipulation and processing and other specific functionality that show how the use cases are to be satisfied. Given an Image by user, the model characterizes the image into one of the labels learned during training.

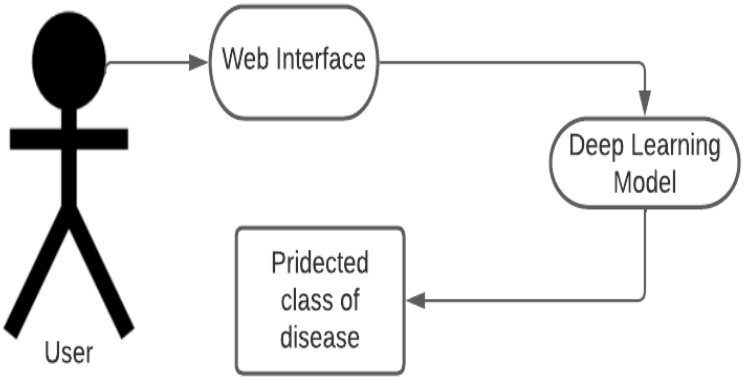

Figure 3.2: Functional Requirement

3.5 Data pipeline:

Data pipeline is a means of automating the machine learning workflow by enabling data to be transformed and correlated into a model that can then be analyzed to achieve outputs.

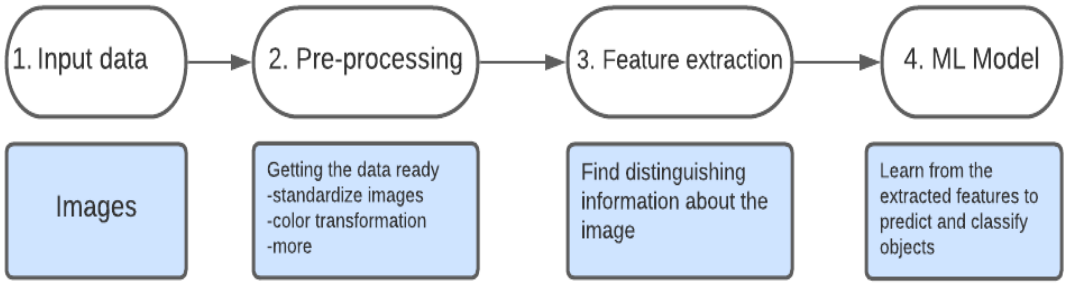

Figure 3.3: Data pipeline

3.6 Design Overview

The design overview of our apple disease prediction system is shown in figure 3.4. Our design consists of using training images to train a neural network and then using our trained model to generate results on test images.

In computer vision, a feature vector collection refers to a set of numeric features that are extracted from an image. These features are usually low-level and high-dimensional representations of the image's visual properties, such as texture, color, shape, or gradient orientation. The purpose of feature extraction is to capture the essential information of the image that is relevant for a particular task, such as classification, detection, or segmentation.

Once the features are extracted, they are typically organized as a feature vector and used as input to the image classifier which in our cas is a deep learning algorithm that learns the mapping between the features and the corresponding output labels.

Once our model learns the mapping between feature vectors and corresponding labels, we take the feature vector of our testing images as an input to the model and our model performs

the classification task.

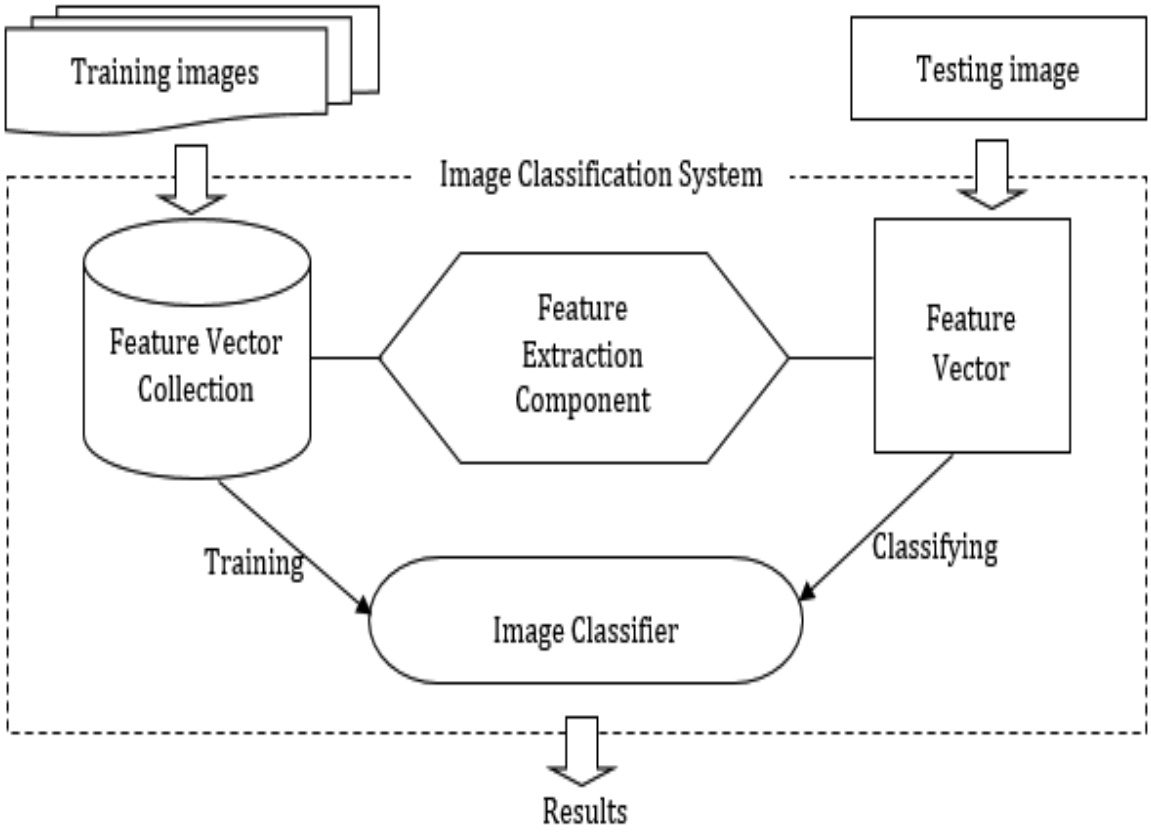

Figure 3.4: Design overview of apple disease prediction system

3.7 Chapter Summary

This chapter discussed the methodology used in our research to develop a system for predicting diseases of apple leaves using computer vision. The approaches used, functionalities, use case diagram, functional requirements, technology used, data pipeline, and design overview were presented. The next chapter discusses the project implementation and system requirements.

Chapter 4
Project Implementation

4.1 Overview

In this chapter, we describe the implementation details of our system for predicting diseases of apple leaves using computer vision. We first discuss the system requirements for the implementation and then a detailed description for training, validating and testing all our models.
We trained and evaluated six different models:

1. ResNet-50.
2. VGG-16.
3. MobileNet-V2.
4. Xception.
5. Inception-V3.
6. Lightweight architecture designed from scratch.

This chapter also provides a detailed description of each architecture as well as their architectural diagrams. We also discuss the training, evaluation process, and a comparison and analysis of their performance.

4.2 System requirements

4.2.1 Software Requirements:

Language:	Python 3
Framework:	Keras and Tensorflow
Operating System	Mac OS / Windows

4.2.2 Hardware Requirements:

GPU:	Above or equal to 8 GB
Hard-disk:	20 GB
RAM:	Above or equal to 4 GB

4.3 Model training

To select the most suitable models for our study, we conducted a comprehensive analysis of various popular convolutional neural network (CNN) models. Based on our analysis, we selected ResNet-50, VGG-16, MobileNet-V2, Xception, and Inception-V3 for our study due to their proven success in image classification tasks. Additionally, we designed a lightweight architecture from scratch for comparison purposes.

For each of our models we resized images to 96 x 96 pixels and processed the input data according to the individual requirements of all pre-trained models by using *preprocess_input* (see appendix A for more details on this) method defined for each pre-trained model available in tensorflow. We then split the dataset into training, validation, and test sets with a ratio of 80:10:10.

We used the Adam optimizer to train the model using sparse categorical cross-entropy loss function with an initial learning rate 0.001, a variable learning rate that decreases the learning parameter if validation loss did not improve for 5 consecutive epochs by multiplying it by 0.2, and lastly, 8-epoch early stopping criterion is used which means training will ends if the validation loss does not decrease for 8 consecutive epochs. Now we will discuss implementation details of each model separately.

4.4 ResNet-50:

ResNet-50 is a convolutional neural network architecture that was introduced by researchers at Microsoft Research Asia in 2015. It is designed to address the vanishing gradient problem that occurs when training deep neural networks. ResNet-50 has 50 layers and uses residual connections to allow the network to learn residual functions. The architecture includes convolutional layers, pooling layers, and fully connected layers. The complete architecture is shown at Fig 4.1

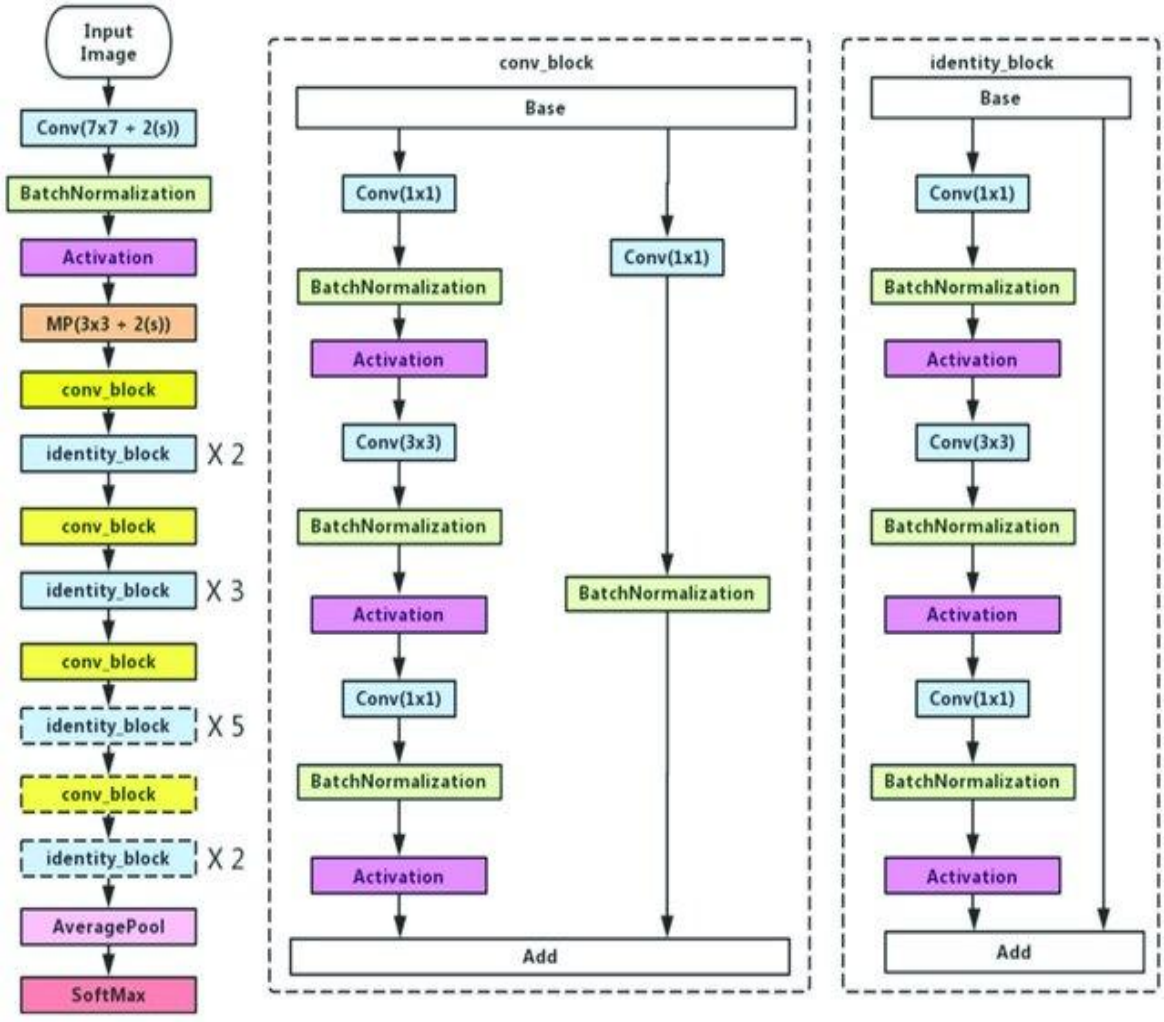

Fig 4.1 ResNet-50 Architecture

To train the ResNet-50 model, we used the mentioned dataset. The dataset contained images of healthy and diseased apple leaves. We trained the model using stochastic gradient descent (SGD) with a learning rate of 0.001 and a batch size of 64. We trained the model for 30 epochs, and the training process took approximately 20 minutes. The ResNet-50 model got the validation accuracy of 99.81%.

To evaluate the ResNet-50 model, we used our hold out test dataset. We calculated the accuracy, precision, recall, and F1 score (see appendix B for more information on these metrics) of the model. The ResNet-34 model achieved an accuracy of 98.46%, a precision of 98.45%, a recall of 98.49%, and an F1 score of 98.66%.

4.5 VGG-16:

VGG-16 is a deep convolutional neural network architecture that was first introduced in 2014 by Karen Simonyan and Andrew Zisserman. It consists of 16 layers and is designed to learn features from images. The VGG16 model is known for achieving a test accuracy of 92.7% in ImageNet, a dataset containing more than 14 million training images across 1000 object classes. The architecture includes convolutional layers, pooling layers, and fully connected layers. The complete architecture is shown at Fig 4.2

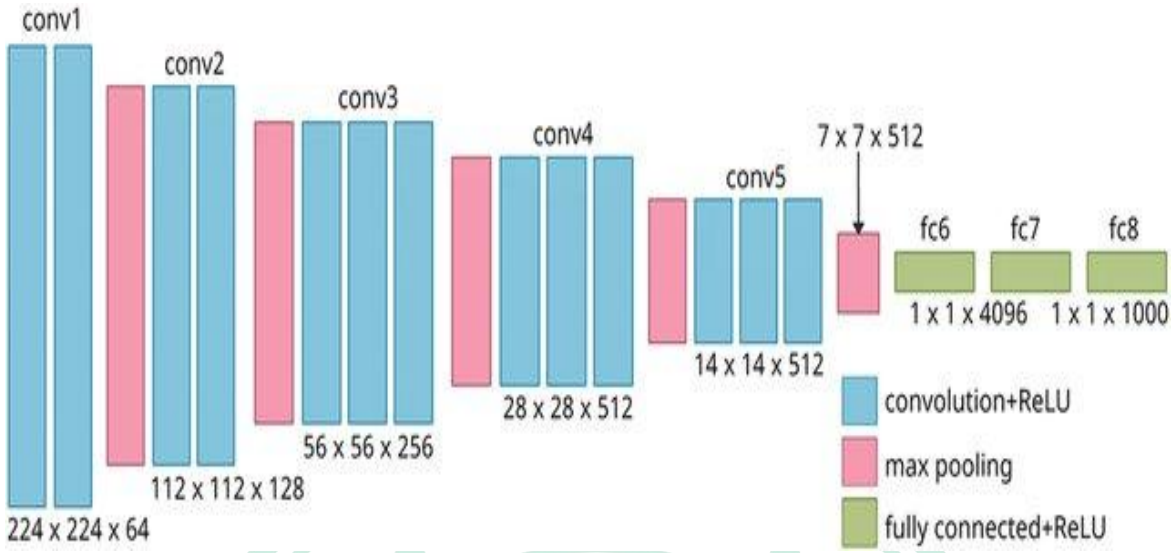

Figure 4.2: VGG-16 Architecture

To evaluate the VGG-16 model, we used our holdout test dataset. We calculated the accuracy, precision, recall, and F1 score of the model. The VGG-16 model achieved an accuracy of 96.41%, a precision of 96.49%, a recall of 96.42%, and an F1 score of 96.41%.

4.6 Xception:

Xception is a deep convolutional neural network architecture that was first introduced in 2016 by François Chollet (creator of keras library). It is designed to learn features from images using depthwise separable convolutions. The architecture includes convolutional layers, pooling layers, and fully connected layers.

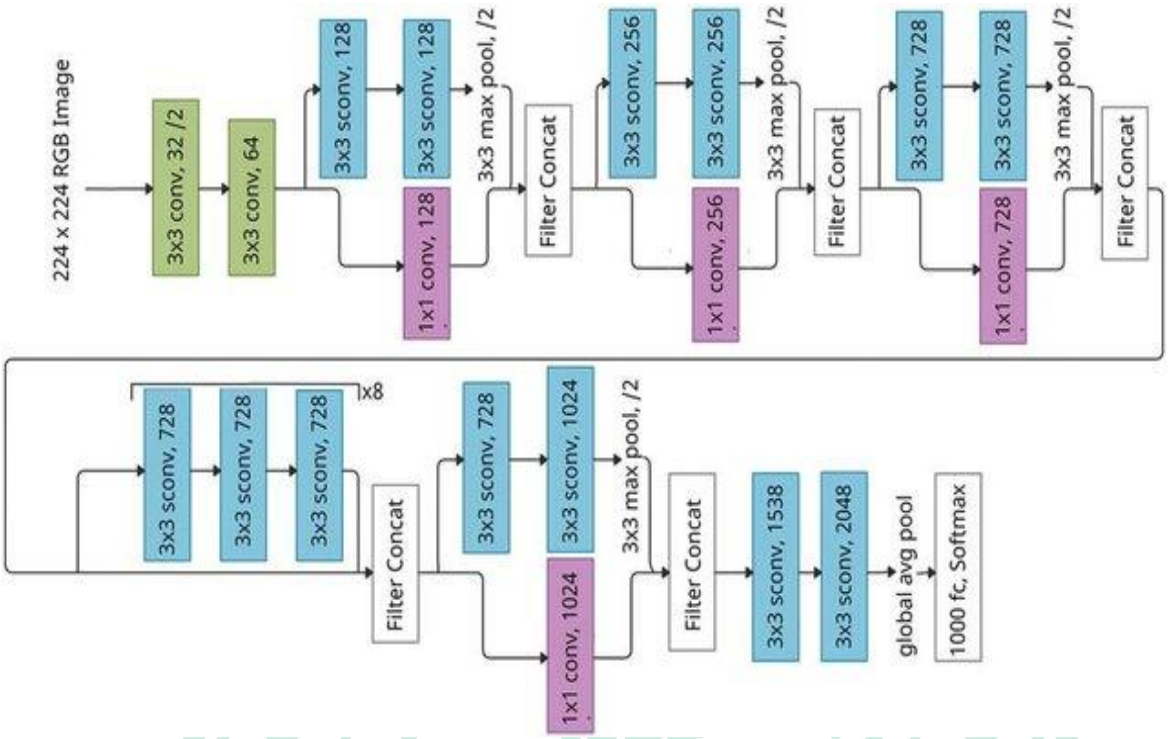

Figure 4.3: Xception Architecture

The Xception model got the validation accuracy of 98.49%. To evaluate the Xception model, we used our holdout test dataset. We calculated the accuracy, precision, recall, and F1 score of the model. The Xception model achieved an accuracy of 94.47%, a precision of 94.54%, a recall of 94.45%, and an F1 score of 94.47%.

4.7 Inception V-3:

Inception v3 is a convolutional neural network architecture that was developed by researchers at Google in 2015. It is a variant of the original Inception architecture, which was designed to improve the computational efficiency of deep neural networks. Inception v3 uses a combination of 1x1, 3x3, and 5x5 convolutional filters in parallel to extract features from input images. Inception v3 achieved state-of-the-art performance on the ImageNet

classification benchmark and has been widely used in various computer vision applications, including object recognition and detection, image segmentation, and medical imaging. The complete architecture is shown at Fig 4.4

To evaluate the Inception model, we used our holdout test dataset. We calculated the accuracy, precision, recall, and F1 score of the model. The model achieved an accuracy of 88.11%, a precision of 88.12%, a recall of 88.99%, and an F1 score of 88%.

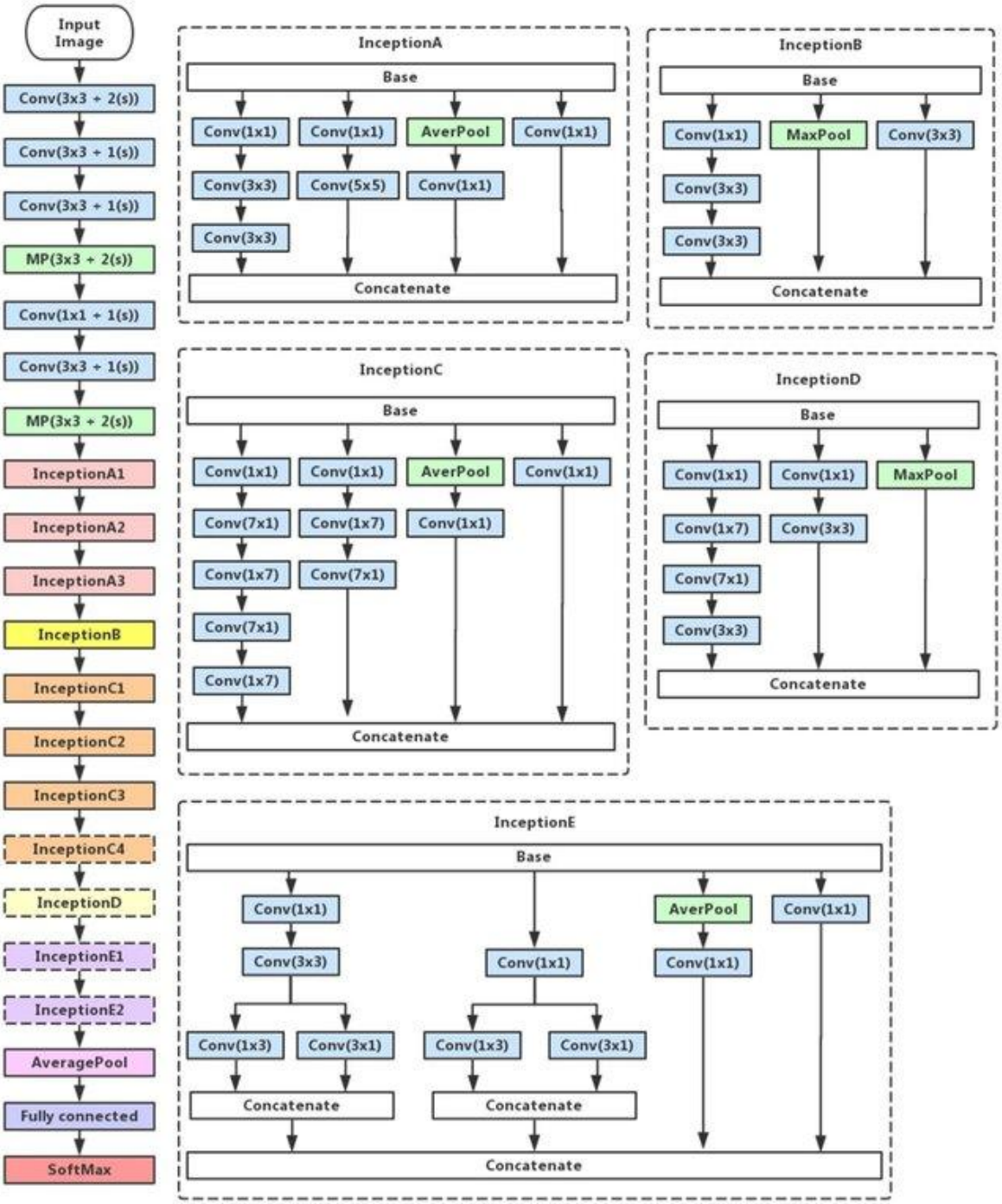

Figure 4.4: Inception-V3 Architecture

4.8 MobileNet V-2

MobileNetV2 is a deep neural network architecture developed by Google for efficient mobile vision applications. It is an improved version of the original MobileNet architecture, with a focus on reducing model size and increasing accuracy. MobileNetV2 achieves this by using a combination of depthwise separable convolutions and linear bottlenecks. It is a light weight model having very few parameters as compared to the other models used.

Depthwise separable convolutions split the traditional convolution operation into two parts: a depthwise convolution and a pointwise convolution. This reduces the computational cost by decreasing the number of parameters required to process each input feature map. Linear bottlenecks further reduce the dimensionality of feature maps by applying a 1x1 convolution before and after the depthwise separable convolution. The complete architecture is shown at Fig 4.5

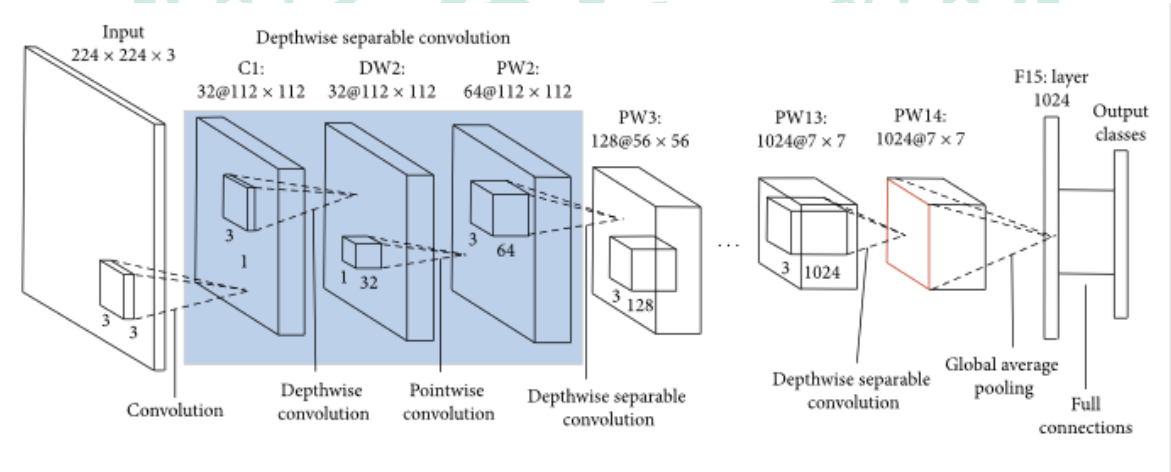

Figure 4.5: MobileNet V-2 Architecture

To evaluate the MobileNet-V2 model, we used our holdout test dataset. We calculated the accuracy, precision, recall, and F1 score of the model. The MobileNet-V2 model achieved an accuracy of 99.28%, a precision of 99.28%, a recall of 99.30%, and an F1 score of 99.28%.

4.9 Proposed model:

Our architecture is a custom-designed CNN that we created specifically for the task of disease prediction in apple leaves. Designing this architecture was an iterative process. We added and removed layers continuously to satisfy parameters vs accuracy tradeoff. At last we settled with the architecture which consists of 10 layers of standard convolutional layers, each of which is followed by a batch normalization layer, a ReLU activation layer, and a dropout layer to prevent overfitting. Additionally, we incorporated five max pooling layers with a pool size of 2 to reduce the dimensionality of the output. The complete architecture is shown at Fig 4.6 and Fig 4.7

To evaluate the proposed model, we used our holdout test dataset. We calculated the accuracy, precision, recall, and F1 score of the model. The model achieved an accuracy of 94.77%, a precision of 95.98%, a recall of 95.22%, and an F1 score of 95.58%.

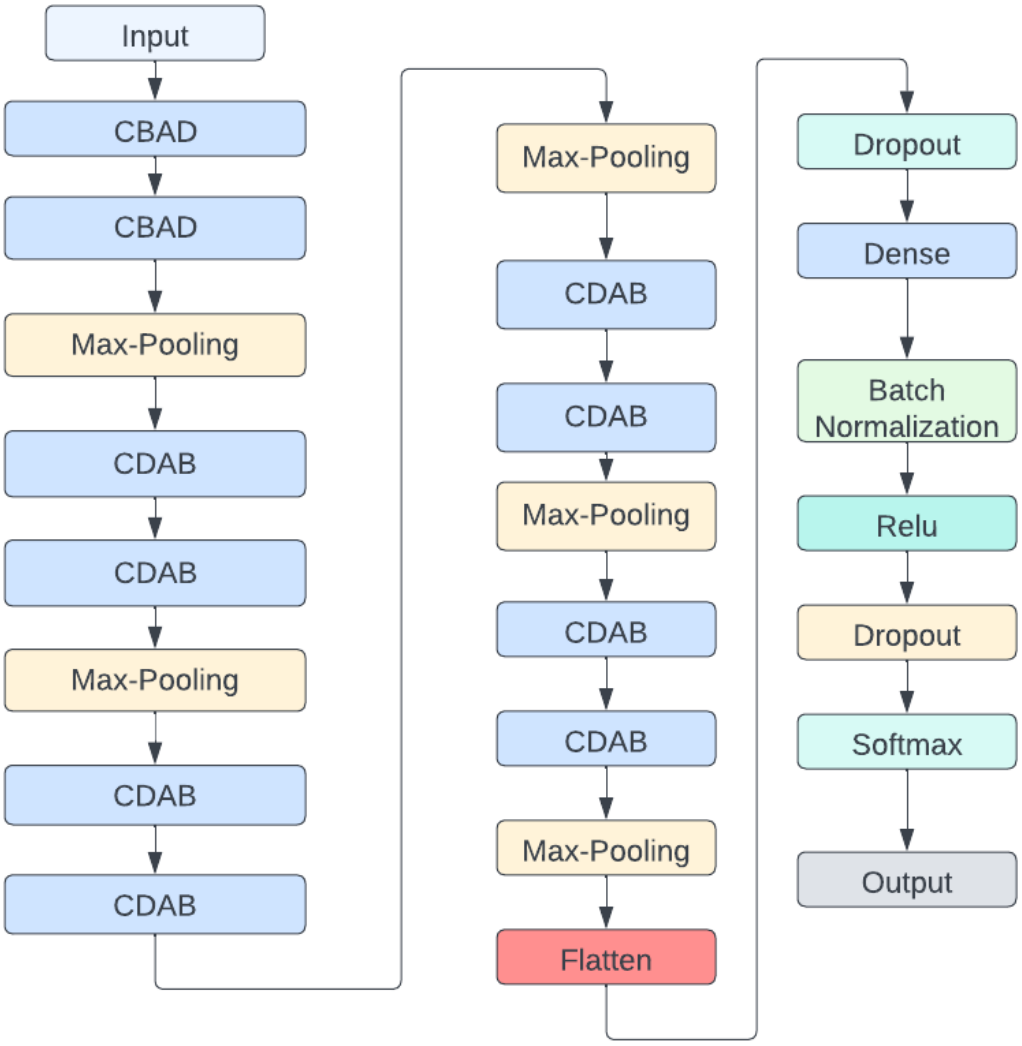

Figure 4.6: Proposed model Architecture

Figure 4.7: One CBAD block of Proposed Model

In total, our model has 146,820 trainable parameters. While this may seem like a relatively small number compared to other state-of-the-art models like ResNet-50 or InceptionV3, it was a deliberate choice on our part. We wanted to create a lightweight model that could be trained quickly and easily making it accessible to a wider range of users and potentially more useful in practical applications. The lightweight architecture can also be used in edge devices with memory constraints such as agricultural drones that are used to maintain large areas of orchards.

Despite its relatively small size, our model achieved impressive results in predicting diseases in apple leaves. In the next chapter, we will compare its performance to other popular architectures and discuss the implications of our findings.

4.10 Proposed Model Code Snapshot:

Following is the snapshot of code that makes our proposed model.

```
IMG_SIZE = (250,250)
rescale = tf.keras.Sequential(
    [
        tf.keras.layers.Rescaling(scale=1./255),
    ]
)

def get_model(image_shape=IMG_SIZE):

  '''In this function we are defining our own model from
scratch'''
  input_shape = IMG_SIZE + (3,)
  input = tf.keras.Input(shape=input_shape)
  x = rescale(input)

  x = tf.keras.layers.Conv2D(16, 3, padding='same')(x)
  x = tf.keras.layers.BatchNormalization()(x)
  x = tf.keras.layers.Activation('relu')(x)
  x = tf.keras.layers.Dropout(0.1)(x)
  x = tf.keras.layers.Conv2D(16, 3, padding='same')(x)
  x = tf.keras.layers.BatchNormalization()(x)
  x = tf.keras.layers.Activation('relu')(x)
  x = tf.keras.layers.Dropout(0.1)(x)
  x = tf.keras.layers.MaxPooling2D()(x)
```

```
x = tf.keras.layers.Conv2D(16, 3, padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Dropout(0.1)(x)
x = tf.keras.layers.Conv2D(16, 3, padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Dropout(0.1)(x)
x = tf.keras.layers.MaxPooling2D()(x)

x = tf.keras.layers.Conv2D(32, 3, padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Dropout(0.1)(x)
x = tf.keras.layers.Conv2D(32, 3, padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Dropout(0.1)(x)
x = tf.keras.layers.MaxPooling2D()(x)

x = tf.keras.layers.Conv2D(32, 3, padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Dropout(0.1)(x)
x = tf.keras.layers.Conv2D(32, 3, padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Dropout(0.1)(x)
x = tf.keras.layers.MaxPooling2D()(x)

x = tf.keras.layers.Conv2D(64, 3, padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Dropout(0.1)(x)
x = tf.keras.layers.Conv2D(64, 3, padding='same')(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Dropout(0.1)(x)
x = tf.keras.layers.MaxPooling2D()(x)


x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dropout(0.4)(x)
x = tf.keras.layers.Dense(16)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)
x = tf.keras.layers.Dropout(0.4)(x)
output = tf.keras.layers.Dense(4, activation='softmax')(x)
model = tf.keras.Model(input, output)

return model
```

Figure 4.8: Code Snapshot

4.11 Chapter Summary:

This chapter discusses the practical details of our approach to predict diseases of apple leaves using computer vision. We have described the five different models we trained, including ResNet-50, VGG-16, Xception, Inception-V3, MobileNet, and a lightweight architecture designed from scratch. We also provide the results of each model that we evaluated on our test dataset. In the next chapter we will compare and analyze these results**.**

Chapter 5
Experiment analysis

5.1 Overview:

This chapter presents the results obtained from the experiments conducted to evaluate the performance of the six different models. The experiments were conducted on the hold out test dataset. The models were evaluated based on various metrics, such as accuracy, precision, recall, and F1-score (see appendix B for more details on metrics used). The following sections provide a detailed analysis of the results obtained and the discussions on the findings.

5.2 Dataset specification:

The data [16] set we used is publicly available on kaggle platform. This repository contains leaf diseases of different plants but we downloaded only apple plant leaves which are of our use. This dataset has 19428 images of apple leaves belonging to 4 classes:

1. Apple_scab,
2. Apple_Black_rot,
3. Apple_cedar_rust,
4. Apple_healthy.

Table 5.1: Data summary

Diseases	Number of Images
Black_rot	4968
Scab	5040
Cedar_apple_rust	4400
Healthy	5020

We have divided the data set into three separate folders which are training, validation, and testing. The ratio that we have used to split our data is 80% for training, 10% for validation and remaining 10% for testing.

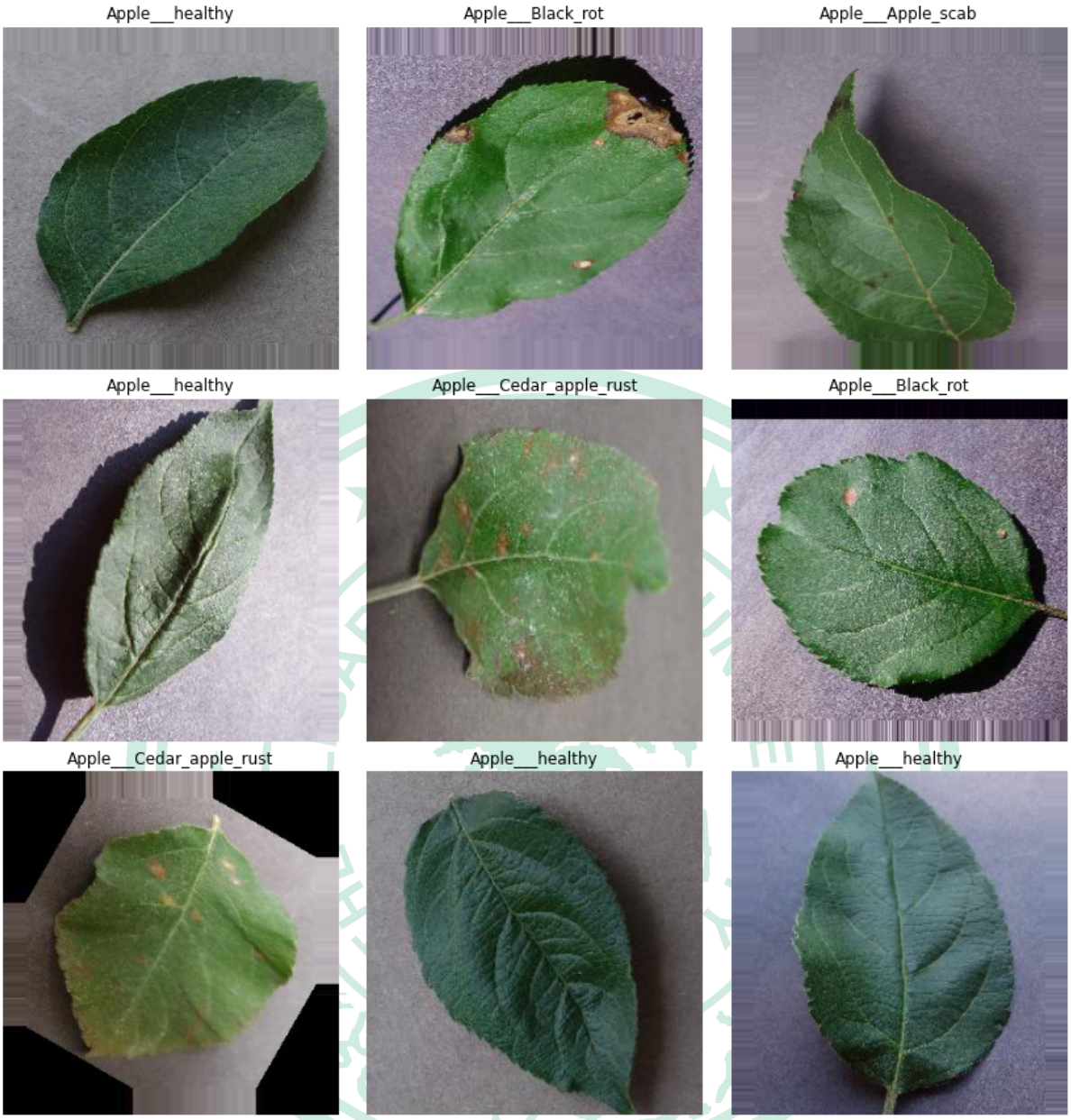

Figure 5.1: Sample Images of Dataset

5.3 Performance metrics:

In this section, we describe the various evaluation metrics that were used to assess the performance of our models. Particularly accuracy, precision, recall, and F1 score will be discussed with the corresponding formulas.

TP: True Positive, TN: True Negative, FP: False Positive, FN: False Negative

1. Accuracy: Accuracy measures the percentage of correctly classified instances over the total number of instances in the test set. It is given by the following formula:
   Accuracy = (TP + TN) / (TP + FP + TN + FN)
2. Precision: Precision measures the fraction of true positives among the total number of positive predictions made by the model. It is given by the following formula: Precision = TP / (TP + FP)
3. Recall: Recall measures the fraction of true positives among the total number of actual positive instances in the test set. It is given by the following formula: Recall = TP / (TP + FN)
4. F1 Score: F1 Score is the harmonic mean of precision and recall. It is a balanced metric that takes into account both precision and recall. It is given by the following formula:

   F1 Score = 2 * (Precision * Recall) / (Precision + Recall)

5.4 Technology used:

We have used following technologies in our project:

- Deep Learning
- Keras
- Tensorflow
- Transfer learning

5.5 Analysis of Outputs:

In the following sections we will discuss the results of individual models that we have trained which is followed by a section that compares all the results.

5.5.1 ResNet-50:

We start by presenting the confusion matrix for the ResNet-50 model in figure 5.1. Based on this matrix, we can see that the model has achieved an overall accuracy of 98.67%, with

highest accuracy for classes *Apple_healthy* and *Apple_cedar_rust*. We can also observe that the model has a low accuracy for *Apple_Black_rot* followed by *Apple_scab*, which could indicate that the model is not performing well in detecting these particular diseases.

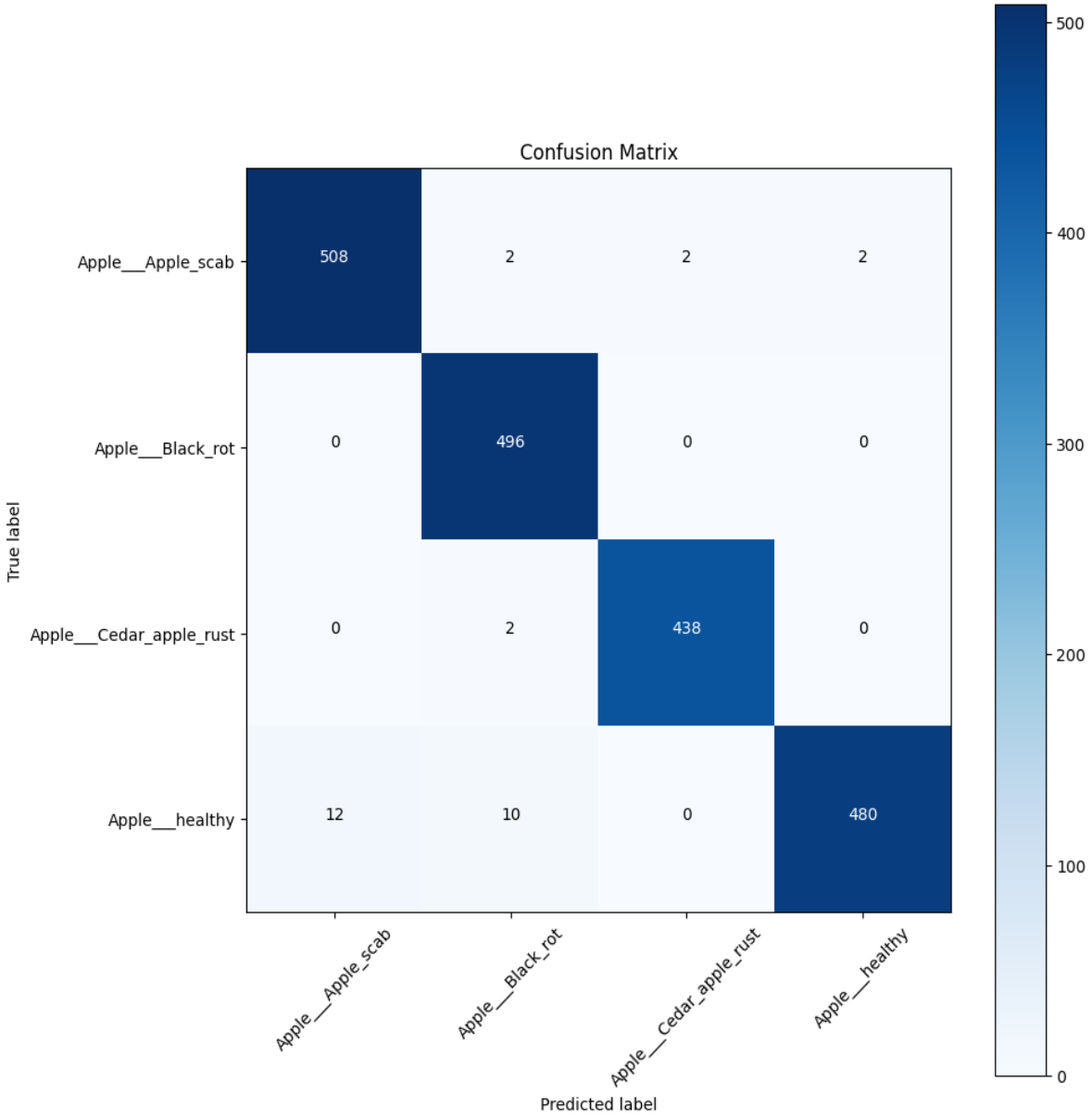

Figure 5.2: ResNet-50 confusion matrix

5.5.2 VGG-16:

Next, we present the confusion matrix for the VGG-16 model in figure 5.2. We can see that this model has achieved the highest accuracy for *Apple_healthy* and lowest for class

Apple_scab. This model also has a high false positive rate for class *Apple_scab* which could indicate that the model is not performing well in detecting these particular diseases.

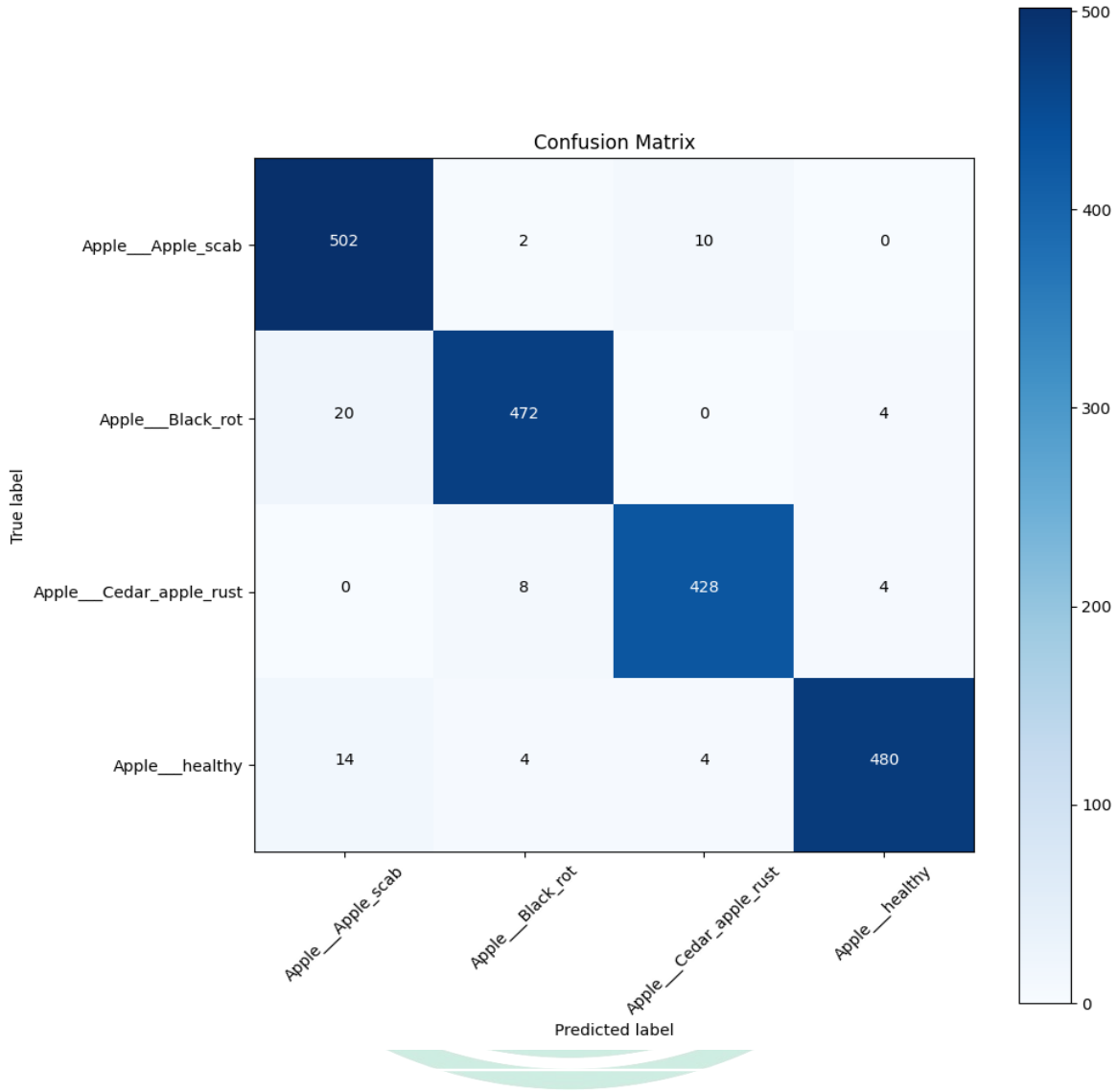

Figure 5.3: VGG-16 confusion matrix

5.5.3 MobileNet-V2

Next we present the confusion matrix of the most successful model of this study- MobileNet-V2. We can observe this model has the lowest false positive rate for all the

classes. It performs extremely well for the Apple_scab and Apple_black_rot, the two classes that all the previous models struggled with.

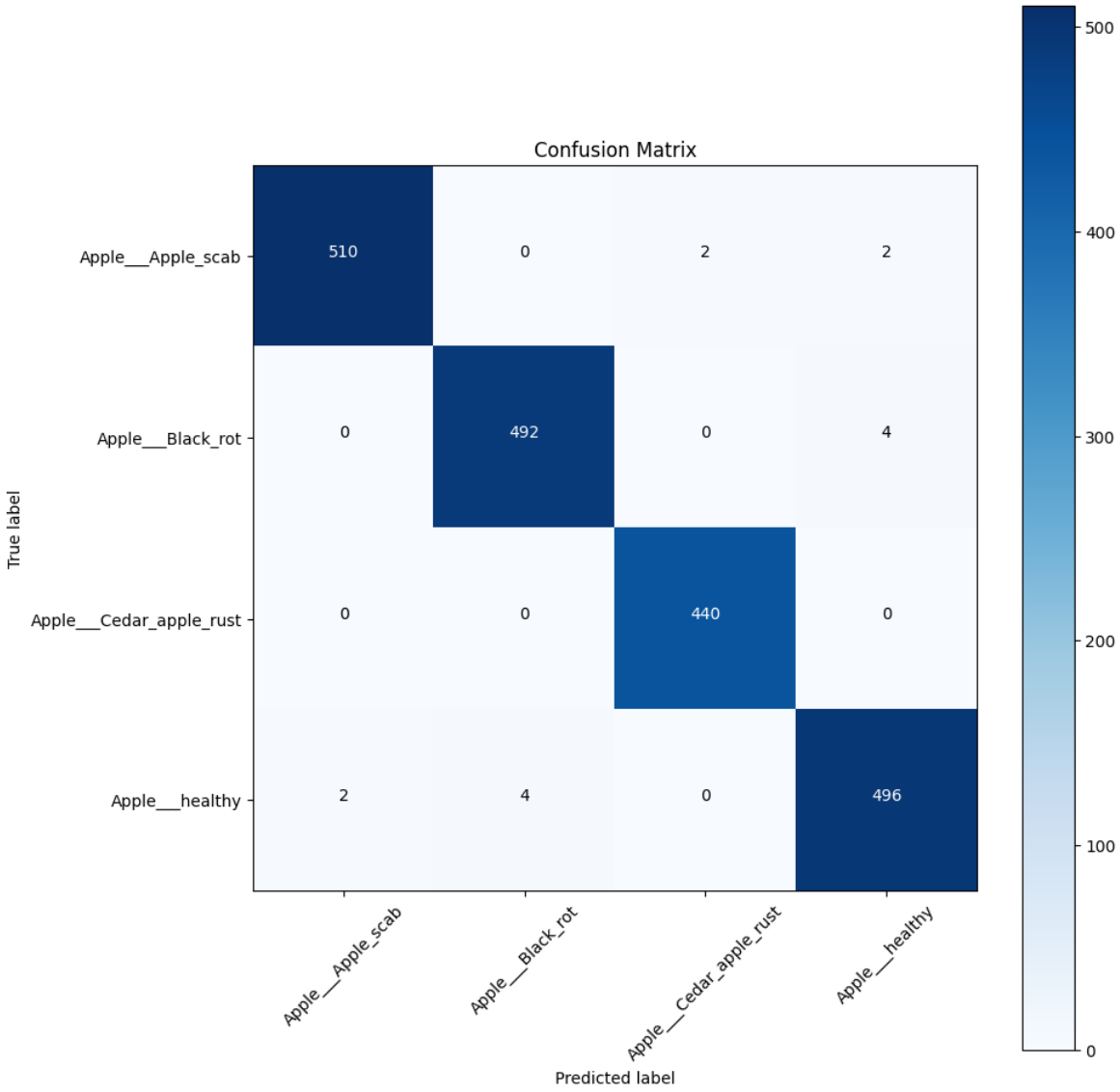

Figure 5.4: MobileNet-V2 confusion matrix

5.5.4 Inception-V3

The confusion matrix for inception-V3 is shown below. As it has already shown Inception-V3 performs lowest of all six models, it can be observed there is a high false-positive as well true-negative rate for all the classes.

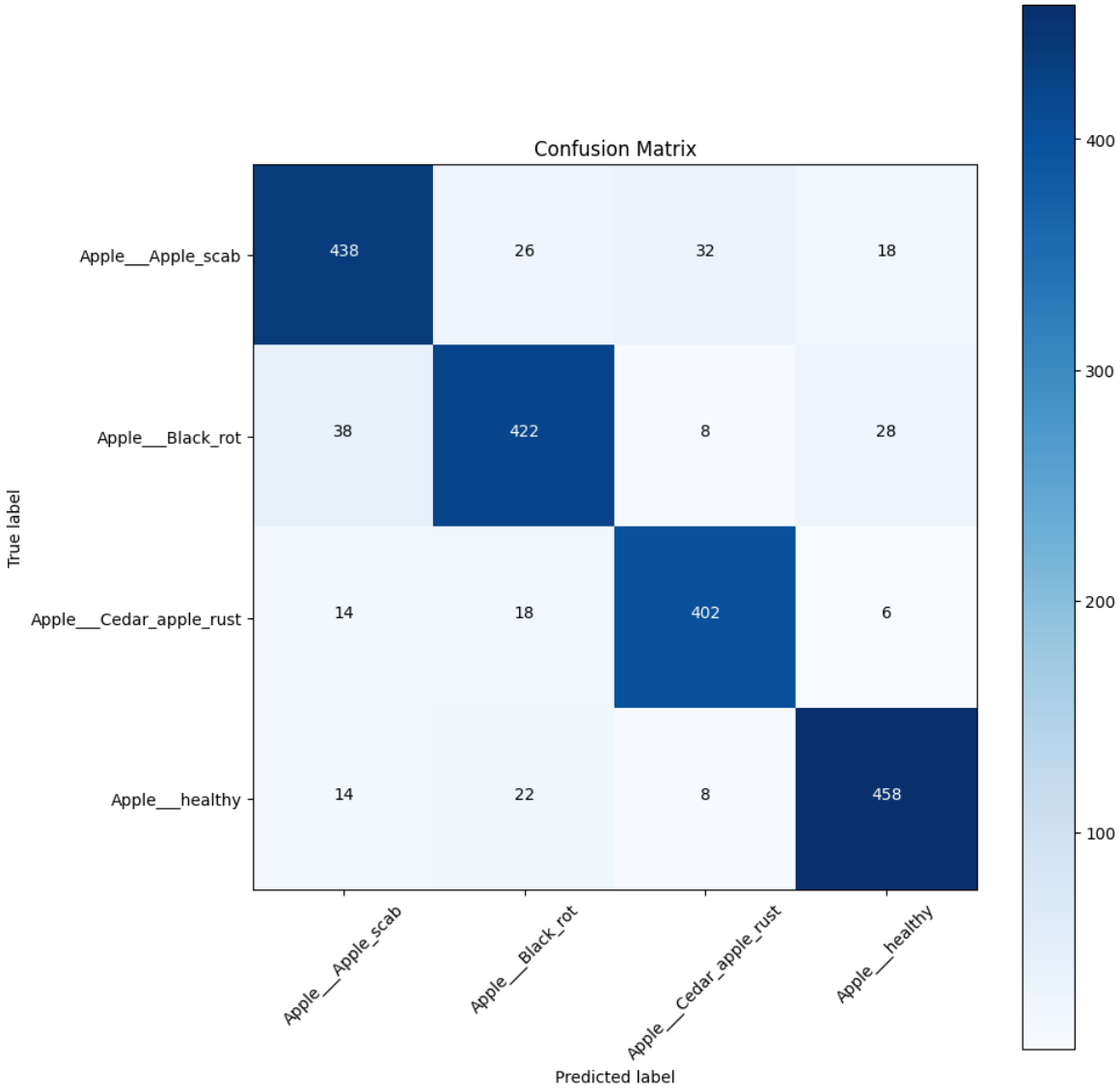

Figure 5.5: Inception-V3 confusion matrix

5.5.5 Xception

The confusion matrix for Xception is shown below. It can be observed that the model performs very well for *Apple_healthy* class as well as *Apple_cedar_rust* but struggles a bit with *Apple_scab* and *Apple_black_rot*.

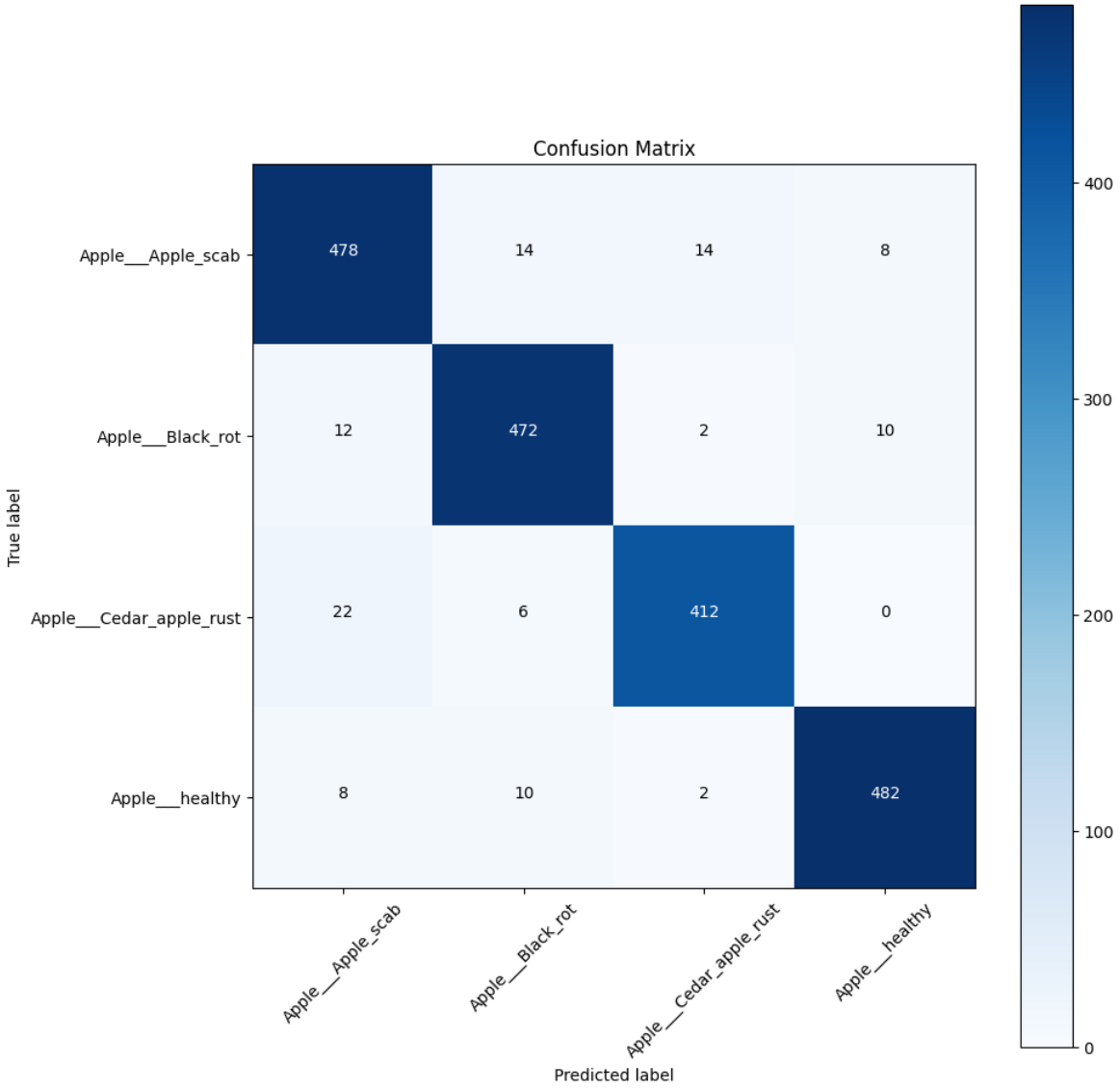

Fig 5.6: Xception confusion matrix

5.5.6 Proposed model

The confusion matrix for the Proposed Model is shown below. It can be observed that our model has a very high true-positive rate for *Apple_Scab*, *Apple_black_rot*, and *Apple_cedar_rust* classes but struggles a bit with the *Apple_healthy* class.

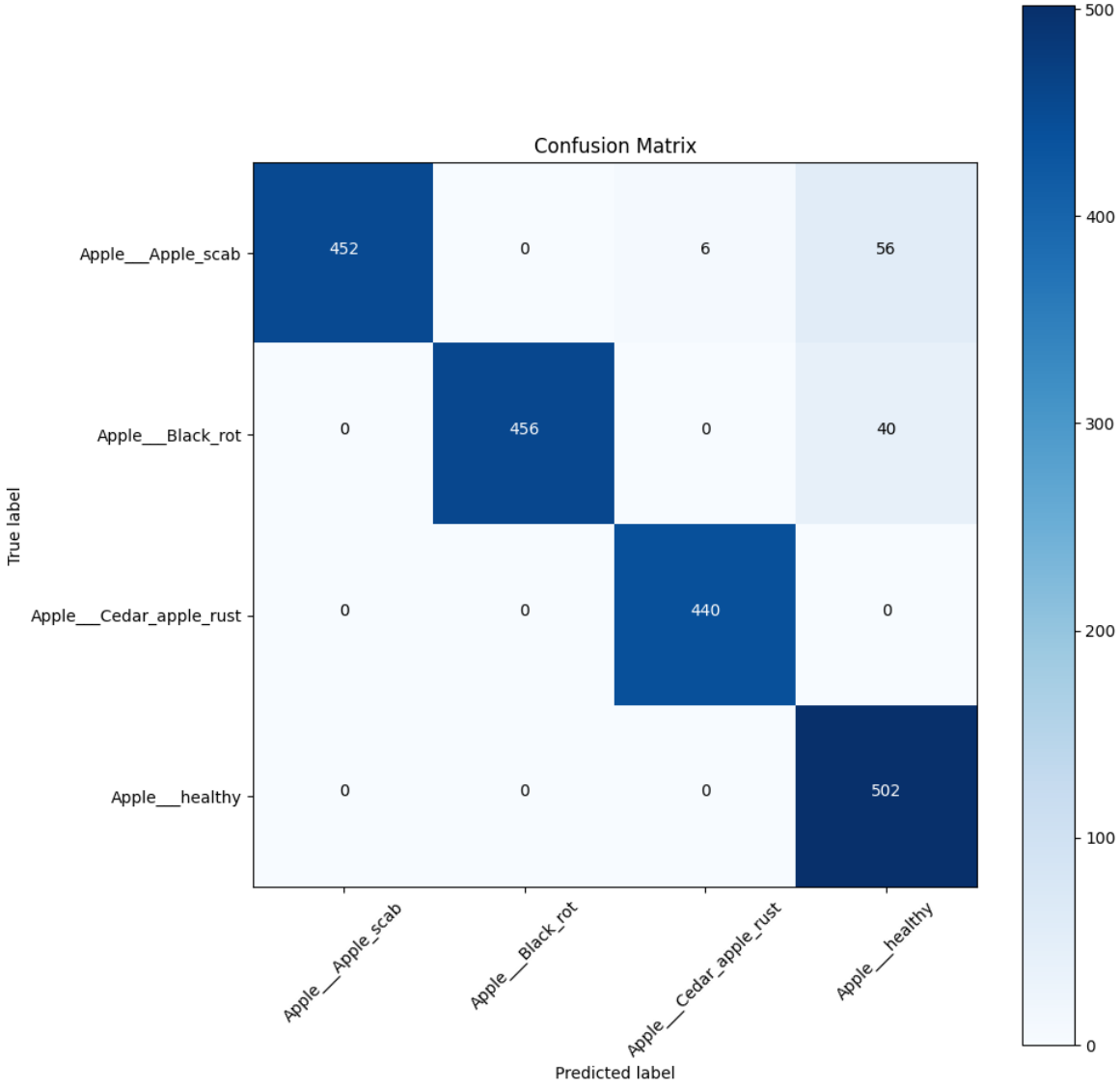

Figure 5.7: Proposed model confusion matrix

5.6 Performance Comparison

From table 5.1, we can see that the MobileNet-V2 model achieved the highest accuracy of 99.28%, followed by ResNet-50 , VGG-16, and then Xception. Our custom-designed model with an accuracy of 94.77% performed very well if we consider the relative size of parameters of each model. The Inception v3 model achieved slightly lower accuracy scores of 88.1%.

When we look at the precision scores, we can see that here also MobileNet-V2 outperforms all other models with the precisions of 99.28%, followed by the ResNet-50 model with a score of 98.45%. The VGG-16 model and Xception model also performed well with precision scores of 96.49% and 94.54%, respectively. Our light weight model achieved a precision of 95.98% which is better than the Inception v3 model which achieved the lowest precision score of 88.12%.

In terms of recall, the MobileNet-V2 model achieved the highest recall of 99.3%, followed by the ResNet-50 model with a recall of 98.49%. The VGG-16 model and Xception model also performed well with recall scores of 96.42% and 94.45%, respectively. The Inception v3 model achieved the lowest recall score of 88.99% which was lower than our custom model which had the recall of 95.22%.

Finally, when we consider the F1-score, MobileNet-V2 model achieved the highest score of 99.28%, followed by the ResNet-50 model with a recall of 98.66%. The VGG-16 model and Xception model also performed well with recall scores of 96.41% and 94.47%, respectively. The Inception v3 model achieved the lowest F1-score of 88% which was lower than our custom model with recall of 95.58%.

Table 5.2 Performance comparison of the trained models

S NO.	Models	Accuracy	Precision	Recall	Parameters(M)
1	ResNet-50	98.67%	98.45%	98.49%	25.59
2	VGG-16	96.41%	96.49%	96.42%	14.7
3	MobileNet-V2	99.28%	99.28%	99.3%	2.26
4	Inception-V3	88.11%	88.12%	88.99%	21.8
5	Xception	94.47%	94.54%	94.45%	21.8
6	Our work	94.77%	95.98%	95.22%	0.146

5.7 Chapter Summary

In this chapter we have discussed the development and training of six different image classification models, including ResNet-50, VGG-16, Xception, Inception-V3, MobileNet-V2, and a lightweight architecture designed from scratch. Through comprehensive evaluations and comparisons, we have demonstrated that the performance of the lightweight architecture is comparable to that of state-of-the-art models while being significantly more computationally efficient.

Our project has also highlighted the potential of lightweight models for practical deployment on resource-constrained edge devices that have limited memory and processing power, such as drones for monitoring large apple orchards.

Chapter 6
Conclusion and future work

6.1 Conclusion:

In conclusion, the project aimed to develop a computer vision-based approach for predicting diseases of apple leaves. Six different models were trained, including ResNet-50, VGG-16, Xception, InceptionV3, MobileNet-V2, and a custom lightweight architecture. The models were evaluated based on their accuracy, precision, recall, and F1-score. The results showed that MobileNet-V2 outperformed the other models in terms of accuracy and F1-score, while the custom lightweight architecture achieved comparable results with significantly fewer parameters.

Moreover, the confusion matrix analysis provided insights into the models' performance in classifying healthy and diseased leaves. The analysis showed that the models had high accuracy in predicting healthy leaves, but some models struggled to classify specific diseases accurately.

Overall, the project demonstrated the potential of computer vision techniques in predicting diseases of apple leaves. The study's findings can be useful for farmers and orchard owners to detect diseases early and take necessary measures to prevent crop loss.

The lightweight architecture we designed in this project has great potential for implementation in edge devices like drones to monitor large apple orchards. With its small size and low computational requirements, it can be easily integrated into the limited processing power and battery life of drones. By using this architecture, the drones can capture images of apple leaves in the orchard and perform real-time disease detection and classification directly on the device without the need for internet connectivity or cloud computing. This can enable farmers to quickly and accurately identify diseases in their apple orchards, facilitating timely intervention and reducing crop loss. With further optimization

and validation, this approach can be extended to other crops and used in a variety of settings, including precision agriculture and environmental monitoring

6.2 Future work:

Future work in this project could include the following:

1. Integration with mobile application: The developed model can be integrated with a mobile application to make it more accessible to farmers.
2. Dataset expansion: In the current study, we used a limited dataset. Expanding the dataset to include more variations of diseases and healthy leaves could improve the model's accuracy.
3. The future work could also explore the potential of using the recently introduced Vision Transformer (ViT) for disease prediction in apple leaves, given its promising results in various computer vision tasks.

References

[1] Singh, Swati, et al. "Deep learning based automated detection of diseases from apple leaf images." CMC-Computers, Materials & Continua 71.1 (2022): 1849-1866.

[2] Zhong, Yong, and Ming Zhao. "Research on deep learning in apple leaf disease recognition." Computers and Electronics in Agriculture 168 (2020): 105146.

[3] Darmé, Luc, and Enrico Nardi. "Exact accidental U (1) symmetries for the axion." Physical Review D 104.5 (2021): 055013.

[4] Huo, X.; Liu, T.; Liu, J.; Wei, Y.; Yao, X.; Ma, X.; Lu, F. 2020 China Apple Industry Development Report (Simplified Version). Chin. Fruit 2022, 42, 1–6.

[5] Wang, N.; Ning, F.; Lu, S. Research on identification method of apple leaf diseases based on support vector machine. Shandong Agric. 2015, 141, 122–125.

[6] Li, C.; Pang, J.; Zhang, S. Apple leaf disease identification method based on feature fusion and local discriminant mapping. Guangdong Agric. Sci. 2016, 43, 134–139.

[7] Shi, Y.; Huang, W.; Zhang, S. Apple disease recognition based on two-dimensionality subspace learning. Comput. Eng. Appl. 2017, 53, 180–184

[8] Dubey SR, Jalal AS. Detection and classification of apple fruit diseases using complete local binary patterns. In2012 Third International Conference on Computer and Communication Technology 2012 Nov 23 (pp. 346-351). IEEE.

[9] Uravashi Solanki, Udesang K. Jaliya and Darshak G. Thakore ," A Survey on Detection of Disease and Fruit Grading", International Journal of Innovative and Emerging Research in Engineering Volume 2, Issue 2, 2015.

[10] Liu, B.; Zhang, Y.; He, D.; Li, Y. Identification of apple leaf diseases based on deep convolutional neural networks. Symmetry 2017, 10, 11.

[11] Zhang, S.; Zhang, Q.; Li, P. Apple disease recognition based on improved deep convolution neural network. J. For. Eng. 2019, 4, 107–112.

[12] Chao, X.; Sun, G.; Zhao, H.; Li, M.; He, D. Identification of apple tree leaf diseases based on deep learning models. Symmetry 2020, 12, 1065.

[13] Zeng, W.; Li, M. Crop leaf disease recognition based on Self-Attention convolutional neural network. Comput. Electron. Agric. 2020, 172, 105341.

[14] Wang, P.; Niu, T.; Mao, Y.; Zhang, Z.; Liu, B.; He, D. Identification of Apple Leaf Diseases by Improved Deep Convolutional Neural Networks With an Attention Mechanism. Front. Plant Sci. 2021, 12, 723294

[15] Bi, C.; Wang, J.; Duan, Y.; Fu, B.; Kang, J.R.; Shi, Y. MobileNet based apple leaf diseases identification. Mob. Netw. Appl. 2022, 27, 172–180.

[16] Link for data set- New Plant Diseases Dataset-: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset
last accessed: 05 May 2023.

Appendix-A

The *preprocess_input* method is a function defined in the *tensorflow.keras.applications* module that is used to preprocess input data for pre-trained models in Tensorflow. This method takes an input image as a parameter and applies a set of preprocessing steps to prepare the image for use with the pre-trained model.

The preprocessing steps may vary depending on the specific pre-trained model being used, but generally include normalization, resizing, and channel reordering. The normalization step ensures that the pixel values of the input image are scaled to a range suitable for the pre-trained model. The resizing step resizes the image to the appropriate dimensions expected by the pre-trained model. Finally, the channel reordering step rearranges the color channels of the image to match the input format of the pre-trained model.

The preprocess_input method is defined for each pre-trained model in Tensorflow, with the specific preprocessing steps tailored to the requirements of each model. Following examples are to call the preprocess_input method ResNet-50 and VGG-16.

from tensorflow.keras.applications.resnet50 import preprocess_input as resnet50_preprocess_input
from tensorflow.keras.applications.vgg16 import preprocess_input as vgg16_preprocess_input