FARM ASSIST – A DECISION SUPPORT SYSTEM FOR FARMERS

A Project Report submitted in the partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

Submitted by

ARAVALA SAI SURYA ANURAG

Reg. No. 168297601001

Under the esteemed guidance of

Prof. P. SURESH VARMA

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIVERSITY COLLEGE OF ENGINEERING

ADIKAVI NANNAYA UNIVERSITY

RAJAMAHENDRAVARAM

2016-2020

ADIKAVI NANNAYA UNIVERSITY

RAJAMAHENDRAVARAM

UNIVERSITY COLLEGE OF ENGINEERING

CERTIFICATE

This is to certify that the project report entitled **"DECISION SUPPORT SYSTEM FOR FARMERS"** submitted by **Mr. Aravala Sai Surya Anurag**, Department of Computer Science and Engineering, Adikavi Nannaya University, is a record of bonafide Project work carried out by him under my supervision and guidance and is worthy of consideration for the award of the degree of Bachelor of Technology in Computer Science and Engineering.

INTERNAL GUIDE **EXTERNAL EXAMINER**

HEAD OF THE DEPARTMENT

DECLARATION

I, **Aravala Sai Surya Anurag** with Registered Number **168297601001**, hereby declare that the project report entitled **“DECISION SUPPORT SYSTEM FOR FARMERS”** done by me under the guidance of **Prof P. Suresh Varma**, Department of Computer Science and Engineering, Adikavi Nannaya University, is submitted for the fulfilment of requirement for the award of the degree, Bachelor of Technology in Computer Science and Engineering in the academic year 2016-2020. I assert the statements made and conclusions drawn are an outcome of my research work. I further certify that I have followed the guidelines provided by the University in writing the report. Whenever I have used materials from other sources, I have given due credit to them in the text of the report and giving their details in the references.

ACKNOWLEDGEMENT

As I present my project on **“DECISION SUPPORT SYSTEM FOR FARMERS”**, I take this opportunity to express my sincere thanks to all those without whose guidance this project might have remained a dream for me. I express my deepest gratitude to Prof. P. Suresh Varma whose guidance and ideas channeled my conscientious endeavors towards the project. I have been fortunate enough that my guide gave me the freedom, support and whole hearted co-ordination for the completion of my project.

ABSTRACT

Agricultural operations work far differently than those a few decades ago, primarily because of advancements in technology including sensors, devices, machines etc. Farmers are utilising scientific data and technology to improve crop yields and are keeping themselves up-to-date with cutting edge methods of farming. The project "DECISION SUPPORT SYSTEM FOR FARMERS" is a new and simplified approach to the farming. This project is an Android Application which intends to help the Indian Farming community to aid their decision making in some aspects of Agriculture. This project aims to provide an integrated and accessible Decision Support System to ease the thought process of Farmers by suggesting a set of crops that are favourable to grow, based on Monsoon, Weather Conditions and Soil Texture. This Application also helps the Farmers to identify Leaf Diseases using Deep Learning. The main idea is to allow Farmers to gain a better understanding of the situation on the ground through advanced technology that can tell them more about their situation than that seen with the naked eye at no price. This Application, is an example to develop the farming approach in India using latest technological advancements in a fruitful manner.

INDEX

S. No.	CONTENTS	Page No.
1.	**Introduction**	**1**
	1.1 Introduction	1
	1.2 Scope	2
2.	**System Analysis**	**3**
	2.1 Existing System	3
	2.2 Proposed System	4
3.	**System Requirements**	**5**
	3.1 Functional requirements	5
	3.2 Non-Functional Requirements	6
	3.3 Hardware and Software Requirements	7
	3.4 Feasibility Study	7
4.	**Technology Description**	**9**
	4.1 Deep Learning	9
	4.2 Colab	9
	4.3 Android Studio	10
	4.4 Tensorflow	10
	4.5 Keras	11
	4.6 Camerax	11
5.	**System Design**	**12**
	5.1 Use Case Diagram	12
	5.2 Sequence Diagram	13
	5.3 Activity Diagram	15
	5.4 Component Diagram	18

6.	**Implementation**	**19**
	6.1 Crop Selection	19
	6.2 Disease Detection	21
7.	**Coding**	**27**
	7.1 Activity CropRecommendation.java	27
	7.2 Activity DiseaseDetection.java	46
	7.3 DiseaseClassifier.py	55
8.	**Output**	**68**
	8.1 Home Screen	68
	8.2 Requesting Location Access	69
	8.3 Crop Recommendations	70
	8.4 Requesting Camera Access	71
	8.5 Disease Detection	72
9.	**Testing**	**73**
10.	**Conclusion**	**74**
11.	**Future Scope & Application**	**76**

CHAPTER 1
INTRODUCTION

India is the second largest Agricultural Hub around the globe and more than 50% of the Indian workforce are dependent on the Agricultural sector in which most of them are small and marginal farmers. India being a large county, its agriculture is composed of many crops, with the foremost food staples being rice and wheat they also grow pulses, vegetables and non-food items. Due to its vastness we come across different soils textures and weather conditions changing from region to region which leads to the concept of suitable crop. Because of this, the farmers need to be highly precise about suitability factors for Crop Selection. Agriculture has also been witnessing changing trends mainly due to increasing rural-urban migration. Thus Farming Population is decreasing day by day and food production has lost its importance. Not having sufficient domestic food production to meet requirement of 1.25 billion plus and still expanding will have a huge burden on Indian Economy. Many schemes and revolutions are in implementation, one such Green Revolution of 1960 is not withstanding todays conditions due to unpredictable Monsoon, ever decreasing Ground Water, decreasing Soil Fertility, unstable Market condition and Mainly lack of Awareness and support in the Farming community. This comes as a huge loss to them and many commit suicide unable to clear their financial debts.

These terrible situations of farmer suicide and huge decline in agricultural productivity lead to national concern as they contribute to 17-18% of countries GDP. To normalize this situations, government came up with promising schemes to encourage farming unit both at state and central level. Majority of these schemes concentrate more on financial aid which contribution is negligible and barely insufficient to at least support one term crop. Apart from this there are many Non-Government Organizations working for the cause of farming community. Every organization works individually on different aspects of farming rather than an applicable collaborative work.

1.1 Scope of the Project

The main goal of this project is to design an Integrated Decision Support System which is capable of handling basic and tough farming traits through an easy scan with a simple mobile application.

⇨ Objectives of DECISION SUPPORT SYSTEM FOR FARMERS

- Support the farmers in decision making process of Crop Selection
- Recommending suitable crops based on local conditions (Precision Farming)
- Identifying the leaf disease caused to a particular crop

CHAPTER 2
SYSTEM ANALYSIS

2.1 Existing System

- Tremendous Research in digitalizing decision support in Indian agriculture yielded many useful applications. Each of which contributing to support decision making only in a single phase either in crop selection or disease detection or yield improvisation. But still there is no reliable integrated solution for Decision Support.

- There are many volunteer federations forming up which are actively undertaking awareness programs and services in many parts of the country so as to help farmers to device their yield. But reaching out to every potential farmer is practically difficult and time taking.

- Framers also get the crop recommendation by government manually through Agricultural Officers. This process may sometimes be delayed and most of the times the situation becomes worst due to corrupt officials and ultimately the farmer is unfamiliar with any of the recommendation.

- Indian Government is extensively working with new technologies and has deployed many web applications. Most of them are not convenient and accessible for direct usage by the farmers.

- There are many Startups emerging in this field releasing commercial IoT based applications which needs extra subscriptions and cannot fully support the farmer in a potential way. Majority of these startups are confined to villages near city region and being highly distant from the actual farming units the farmer might be unaware of the resources available to him.

- ⇨ There are also many Decision Support Systems existed today. These systems use traditional techniques for decision making which in many cases are outdated and are less reliable.

2.2 Proposed System

- ⇨ Proposed system is an Integrated Decision Support System which helps the farmer in few aspect of decision making through latest technology.
- ⇨ DSS are software-based systems that gather and analyse data from a variety of sources. Their purpose is to smoothen the decision-making process for management, operations, planning, and simply recommends optimal solution path. It helps farmers to solve complex issues related to crop production.
- ⇨ This system is an Android based Application with no sensor requirements and simple User Interface making it less complex to use without any technical background.
- ⇨ This system is a proposal which combines all functional requirements for a Healthy Farming Environment with the active user being farmer himself.
- ⇨ This application while considering all required real-time entities provides appropriate suggestions related to farming and thus can greatly reduce the damage caused due to improper awareness and age-old information.

CHAPTER 3
SYSTEM REQUIREMENTS

3.1 Functional Requirements

The functional requirements are statement of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situation. Those requirements depend on the type of software being developed and the expected users of the software,In this case,the Farmer.

3.1.1 Input Credentials

- The system should not include any registration process or any other information gathering forms which are generally intended to farmers
- Requires only two sets of inputs that is, location coordinates and real-time crop images mainly including the diseased leaf of crop.

3.1.2 Interface

- A simple user-interface to be handled even by undereducated farmers
- Should use Automated methods for data processing

3.1.3 Report Generation

- The system should display the soil textureand recommend a set of crops that are favorable to grow in that particular location.
- The System should also be able to identify the Leaf Disease of the plant/crop caused due to external factors.

3.2 Non-Functional Requirements

Nonfunctional requirements are requirements that are not directly concerned with thespec ified function delivered by the system. They may relate to emergentsystem properties such as reliability, response time and store occupancy.

Some of the non-functional requirements related with this system are hereby below:

Performance

- Response Time: The system should give results instantaneously as farmer is not aware of such things and may consider the system is not working.
- Usability: As the farmer is not acquainted well with technology, It is necessary to create a hassle free interface.

Reliability

- Availability & Output : This System should work at any time the farmer wishes to use it for crop recommendations and should provide accurate results.

Scalability

- The system must be scalable to any precise farm location in India
- It must be capable of detecting all the included type of leaf disease accurately

3.3 Hardware and SoftwareRequirements

3.3.1 Software Requirements

➢ Operating System	: Windows 10
➢ IDE:Android Studio 4.0, Google Colab	
➢ Languages	:Python, Java

3.3.2 Hardware Requirements

➢ Processor	: Intel i7 3rd gen
➢ RAM	: 8GB
➢ Graphic Card	: Nvidia GeForce GT 710 2GB

3.4 Feasibility Study

➢ **Technical Feasibility**

As Colab is a free cloud service, we do not require our own high power computing device with GPUs and TPUs for Model Training. Android Studio is open sourced and hence we can develop any kind of app without worrying about procurement of licensed software. The APIs selected for this project have no call limit and hence we can use the same credentials for multiple users.

➢ **Operational Feasibility**

Operational Feasibility is a measure of how well a proposed system solves the problem and during scope definition. The following points were considered for this project

a. The system analyses the image of diseased leaf without capturing.

b. The systemgives the set of recommended crops based on a reliable Soil API run by UNO, Information from Government websites and Research Papers.

- **Behavioral Feasibility**

  The working of system is quite easy to learn due to its simple and convenient user interface. User does not require any special training about the functionality of the system.

CHAPTER 4
TECHNOLOGY DESCRIPTION

4.1 Deep Learning

- Deep learning is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised or unsupervised or semi-supervised.
- Deep learning architectures such as deep belief networks, recurrent neural networks and convolutional neural networks have been applied to fields including computer vision, machine vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases surpassing human expert performance.

4.2 Colab

- Colab is the short form of Colaboratory, a product from Google Research.
- Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including CPUs, GPUs and TPUs.
- It allows anybody to write and execute arbitrary python code through the browser. It is also well suited to machine learning, data analysis and education.
- Colab runs entirely in the cloud. It has facilities like Google Docs i.e. members of the team can modify or change the notebook. It has the facility to connect to the Google Drive for Dataset instead of browsing the local machine.

4.3 Android Studio

- Android Studio is the official Integrated Development Environment (IDE) for Android app development.It is built on JetBrains' IntelliJ IDEA software. It is available for download on Windows, macOS and Linux based operating systems
- It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.
- Android Studio supports all the same programming languages of IntelliJ (and CLion) and more with extensions, such as Go. Android Studio supports all released Java versions, up to Java 12. At least some new language features up to Java 12 are usable in Android.
- Android Studio offers even more features that enhance productivity when building Android apps
- Latest version is Android Studio 4.0

4.4 Tensorflow

- Tensorflow is a free and open-source library for dataflow and differentiable programming developed by Google Brain team for internal Google use.
- It is a symbolic math library, and is also used for Machine Learning applications such as Neural Networks.
- Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

4.5Keras

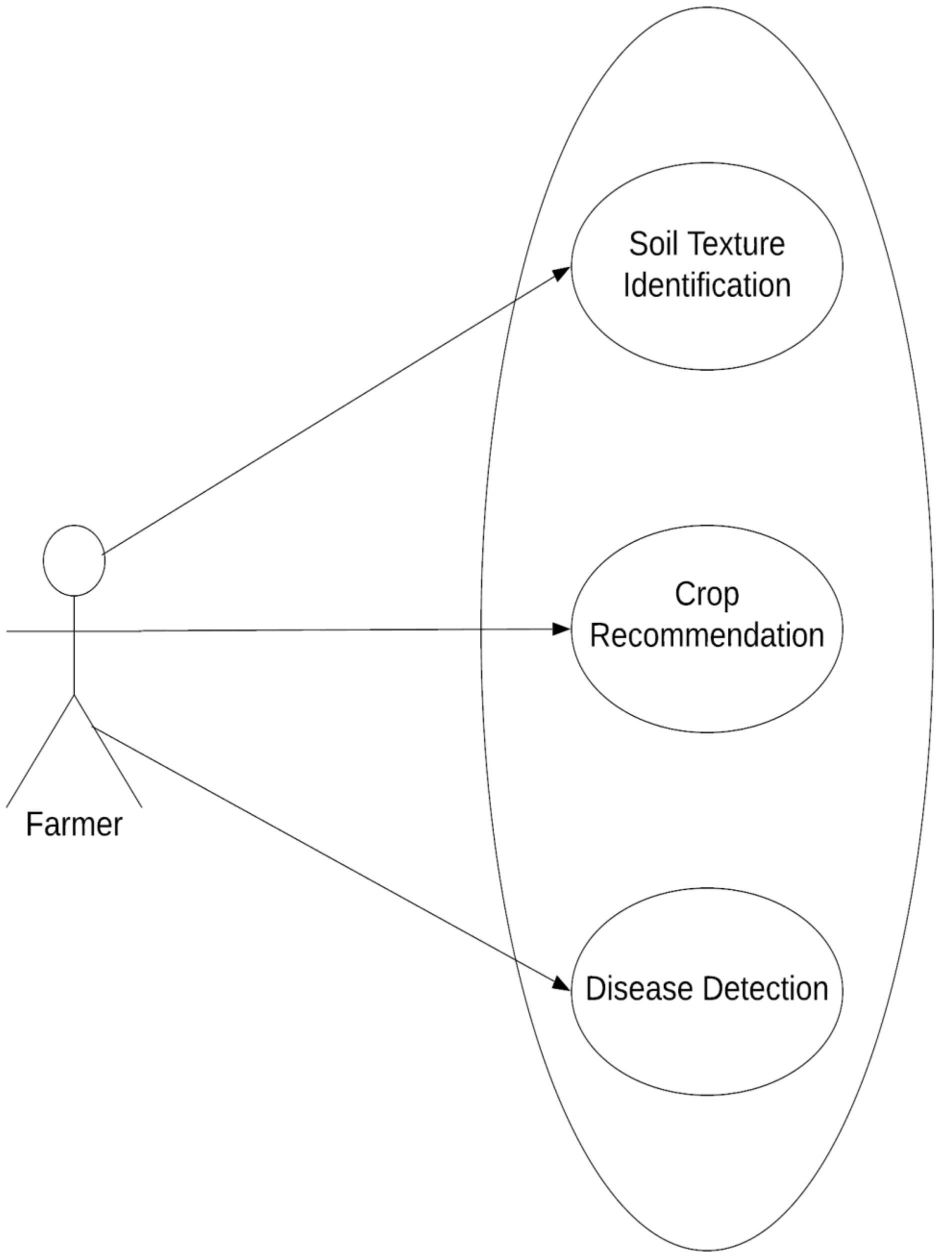
- Keras is an Open-Source Neural Network library written in Python.
- It is capable of running on top of Tensorflow, Microsoft Cognitive Toolkit, R and Theano.
- It follows best practices for reducing Cognitive load : It offers consistent and simple APIs, minimizes the number of user actions required for common use cases and provides clear & actionable error messages.

4.6 Camerax

- CameraX is a Jetpack support library, built to help you make camera app development easier.
- It provides a consistent and easy-to-use API surface that works across most Android devices, with backward-compatibility to Android 5.0 (API level 21).
- While it leverages the capabilities of camera2, it uses a simpler, use case-based approach that is lifecycle-aware. It also resolves device compatibility issues for you so that you don't have to include device-specific code in your code base.
- It supports the image analysis use case which provides your app with a CPU-accessible image to perform computer vision, or machine learning inference.

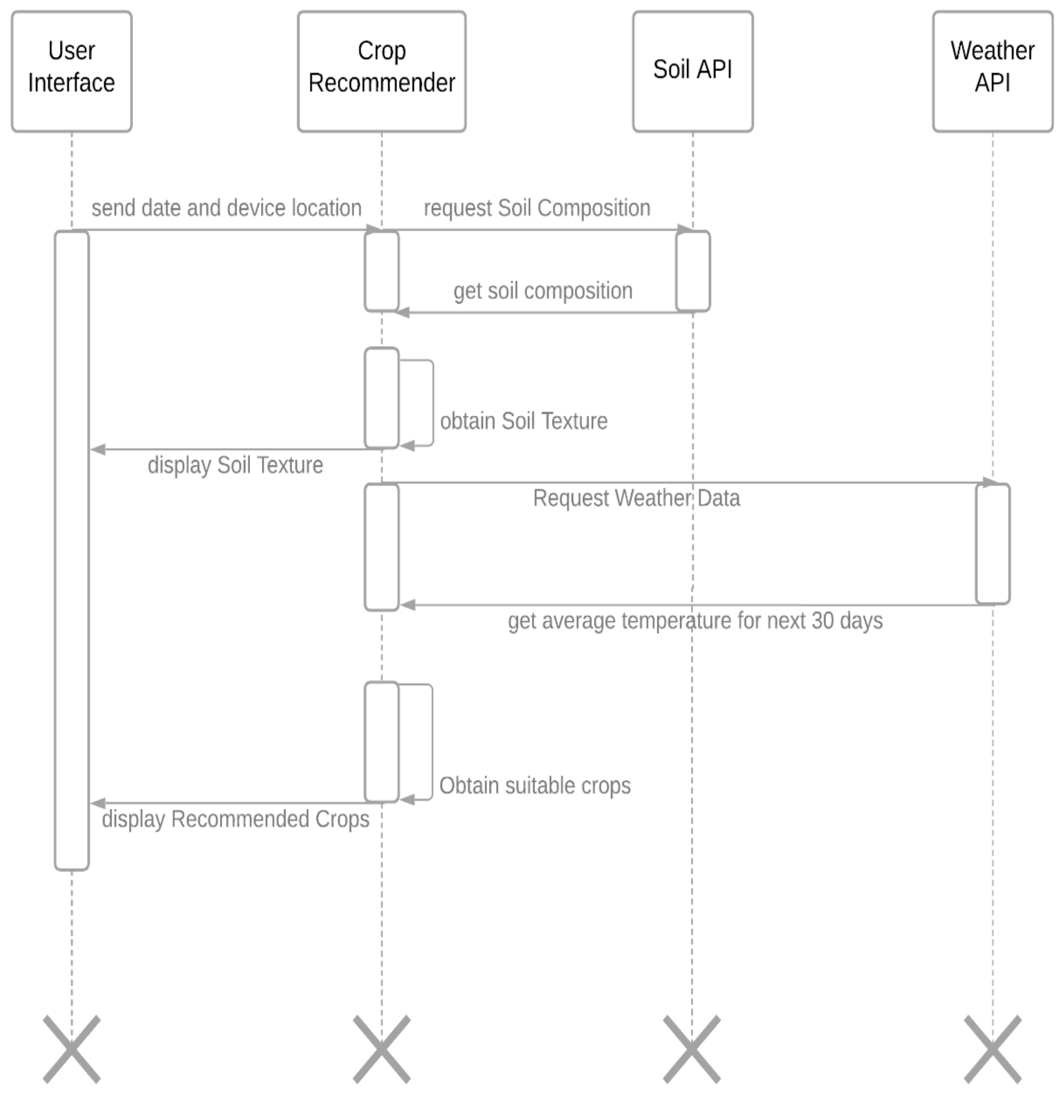
CHAPTER 5
SYSTEM DESIGN

5.1 Use Case Diagram

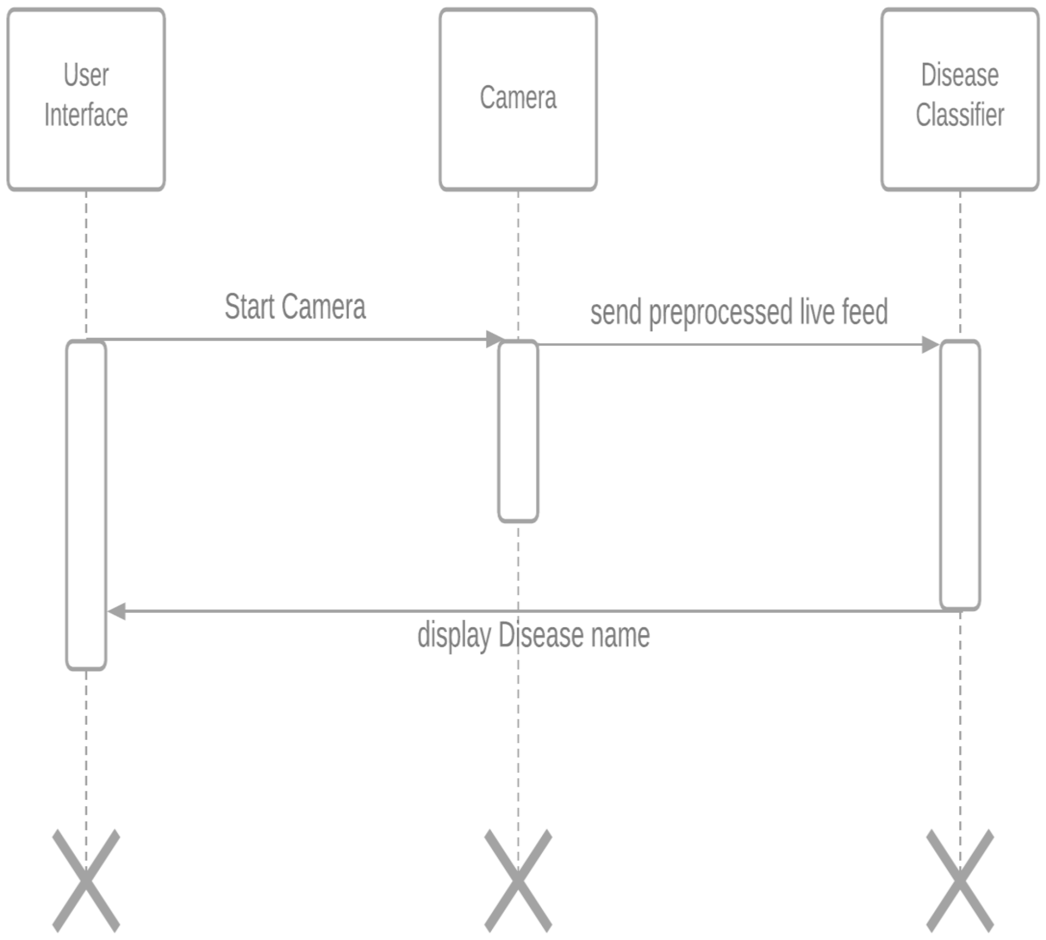


5.2 Sequence Diagram

Sequence Diagram for Crop Recommendation

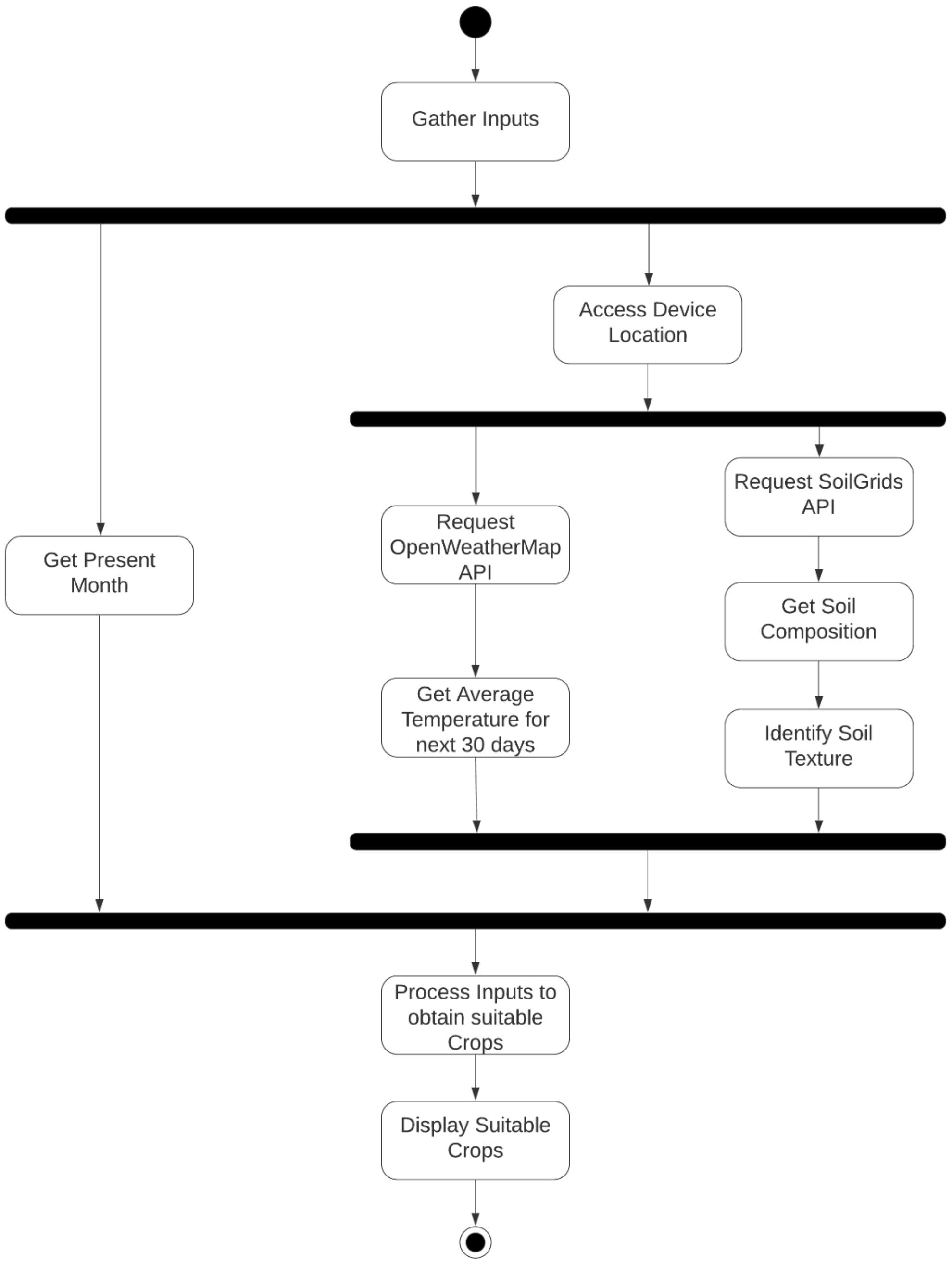


Sequence Diagram for Disease Detection

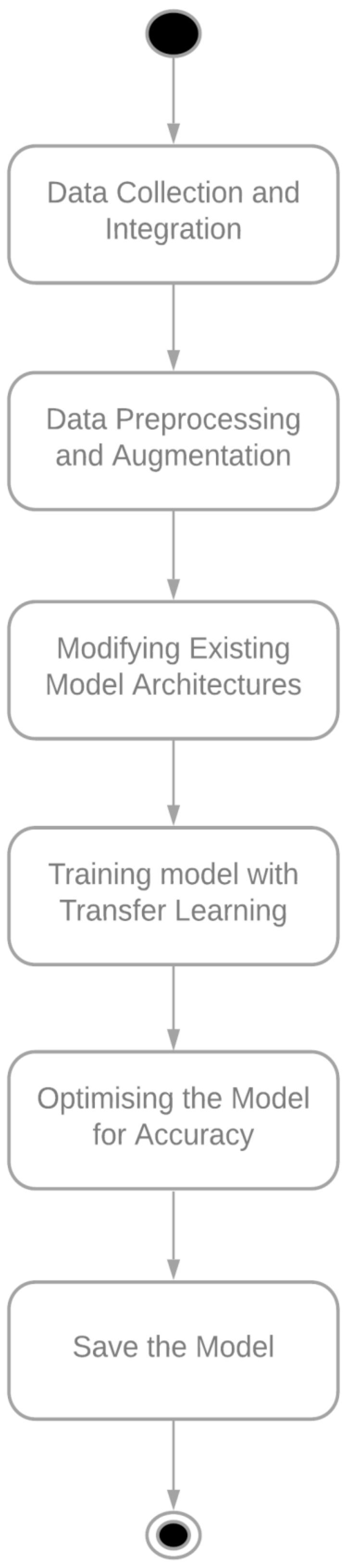


5.3 Activity Diagram

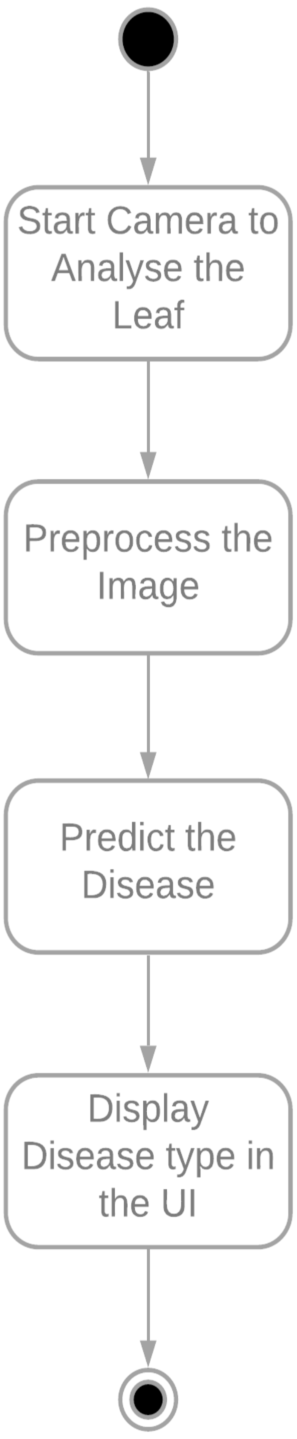
Activity Diagram for Crop Recommendation



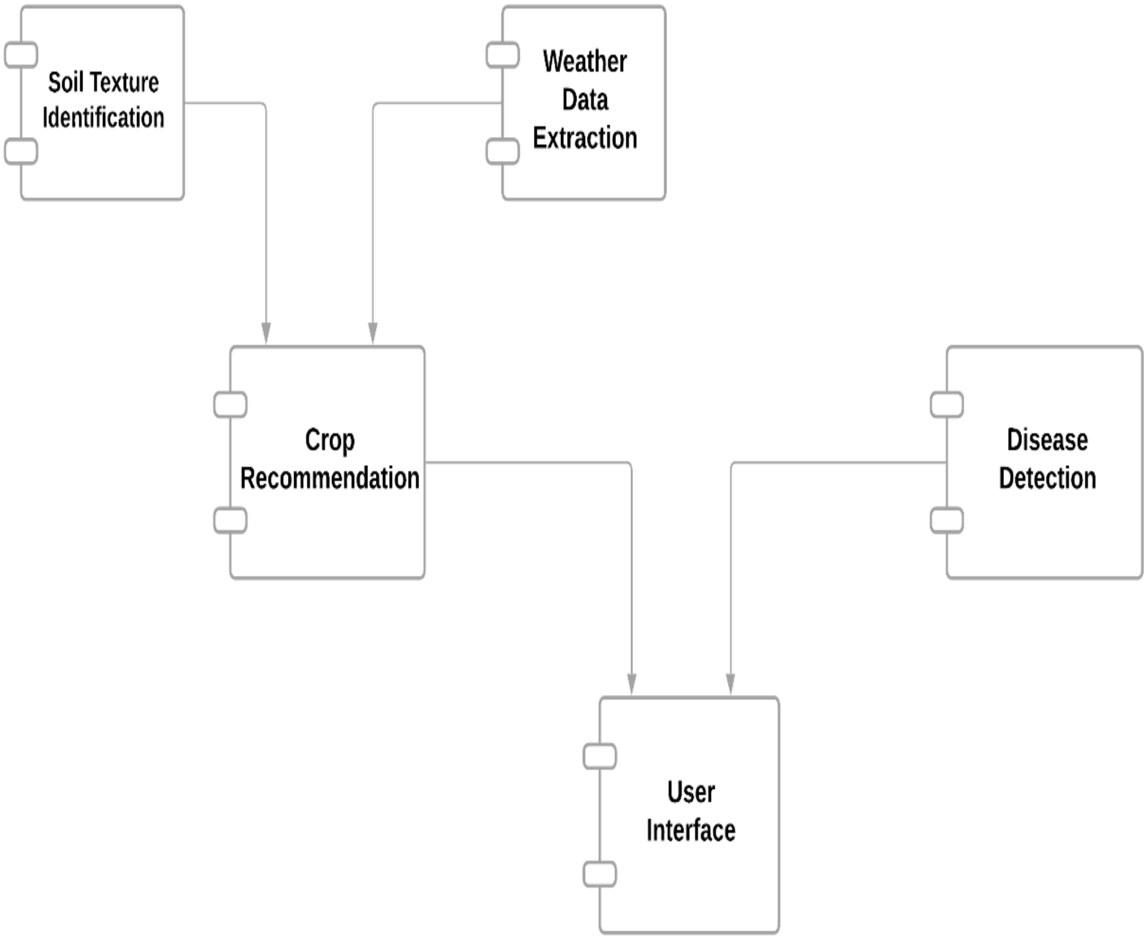
Activity Diagram for Training Disease Classifier


Data Collection and Integration
Data Preprocessing and Augmentation
Modifying Existing Model Architectures
Training model with Transfer Learning
Optimising the Model for Accuracy
Save the Model

Activity Diagram for Disease Detection


Start Camera to Analyse the Leaf
Preprocess the Image
Predict the Disease
Display Disease type in the UI

5.4 Component Diagram



CHAPTER 6
IMPLEMENTATION

This application includes two main modules i.e. crop recommendation and leaf disease detection. Each and every module is an integrated output of a stepwise hierarchal procedures which are capable of providing valid output at each level of hierarchy.

6.1 CROP SELECTION

The main objective of this module is to recommend a suitable crop to the required area, respectively taking various requirements into consideration.

6.1.1 Fundamental prerequisite

Every crop has its own flourishing environments. So, the basic requirements of crop selection are mainly dependent on the suitability factors for a crop. Because of its different prevailing conditions, we acquire information about crop and its suitable growing conditions like type of soil, favourable temperature and suitable periods of initial seed sowing. The information accumulated is precise and reliable as it is retrieved form authorised websites which hold accurate information.

6.1.2 Congregating Inputs

After gathering fundamental requirements, the main objective is to utilise the information in a way to completely automate the crop selection process without any further inputs from the farmer. This automation process requires integration of the inputs that characterise crop selection

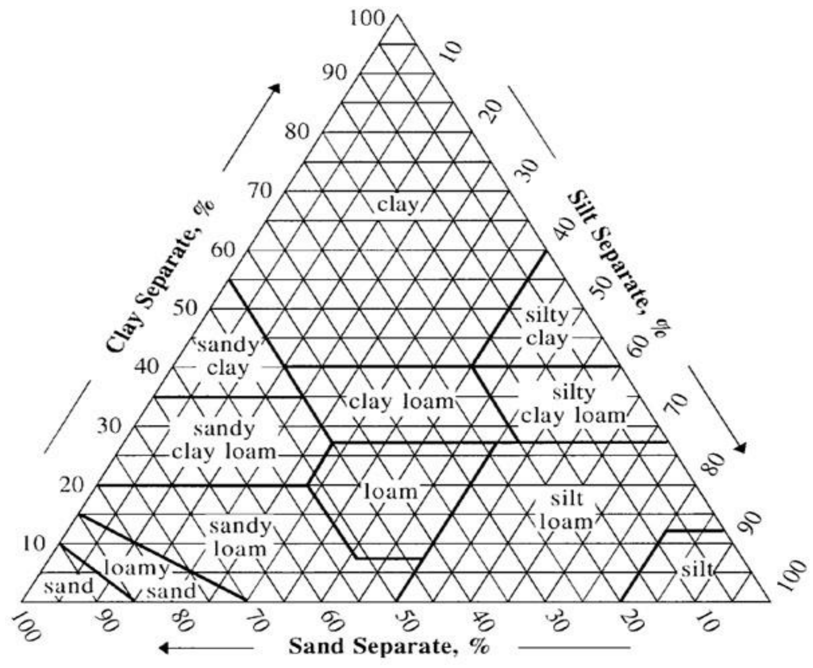
i) Weather Data

Weather conditions is one of the factors in determining the suitable crop. World Weather Online provides worldwide city and town land based real-time and future weather forecast. This weather API provides access to present weather conditions and up to 30 days weather forecast across any remote locations. The weather API give information of the real time day temperature and further forecasts. The average temperature over the upcoming 30 days is calculated and taken as one of the input prerequisites for crop selection.

ii) Soil Texture

The Soil Texture includes silt, clay and sand present at various proportions. The Soil Texture is determined by the amount of each component present in the soil. SoilGrids is a system for automated mapping of soil properties based on global soil profile and covariate data and machine learning algorithms and it is a collection of updatable soil property calibrated to make unbiased predictions at any location within the global soil mask. Integration of SoilGrids API give us the percentage of each component in that particular soil i.e. silt, clay and sand.

This real time percentage generated is compared with the static information from USDA Soil Texture Triangle. Further this comparison determines the soil texture at that particular region.



6.1.3 List of Suitable Crops

The real time quantities, temperature, month and soil textureobtained are important inputs for crop recommendation. The month input is calculated from the calendar instance of the system at that particular day. After determining the inputs, we validate the crops list information prepared earlier with the observed inputs through simple conditional statements. If the suitability factor of crop i.e. temperature, soil type and preferable month is same as the observed units then the crop is appended to the list of suitable crops, which is then finally displayed to the farmer.

6.2 DISEASE DETECTION

In disease detection module, we identify the leaf disease caused to the plant. It involves the following procedure:

6.2.1 Data Collection

The data for plant disease detection is formed from multiple datasets which include PlantVillage dataset (a research dataset by PSU-USA), Citrus leaf diseases dataset and Rice leaf diseases dataset. The combined dataset consists of **46** classes of both healthy and diseased leaf classes.

6.2.2 Data pre-processing

In data pre-processing the integrated dataset is then applied to Augmentation

Data Augmentation

Data Augmentation is technique to increase the diversity of the training set by applying random but realistic transformations. Following are the augmentation methods used in this project

- Position augmentation

  Scaling

  Cropping

  Flipping

  Padding

  Rotation

  Translation

  Affine transformation

- Colour augmentation

  Brightness

  Contrast

  Saturation

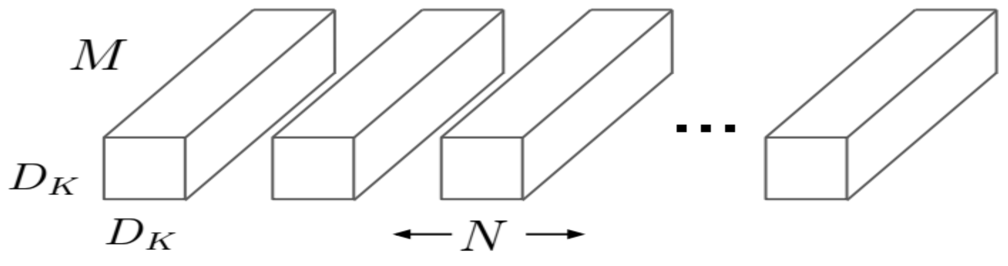
  Hue

6.2.3 Model Development

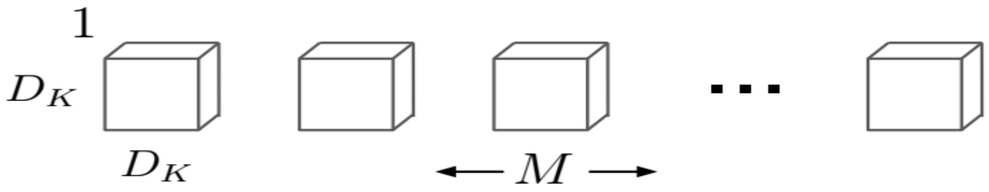
In this project, I have used transfer learning to achieve greater accuracy. **Transfer Learning** is a machine learning technique where a model trained on one task is re-purposed on a second related task. It reduces the Training time and complexity and proven to achieve high accuracy in less no of epochs on a related task. In this project, I chose **MobileNetV1** model for Transfer learning.

6.2.3.1 MobileNet : MobileNet is a small, low-latency, low-power model parameterized to meet the resource constraints of a variety of use cases. It can be built upon for classification, detection, embeddings and segmentation similar to how other popular large scale models, such as Inception, are used. MobileNet can be run efficiently on mobile devices with TensorFlow Lite. It achieves better accuracy with less number of computations and less memory size and hence It is a recommended model for running inference on mobile devices.

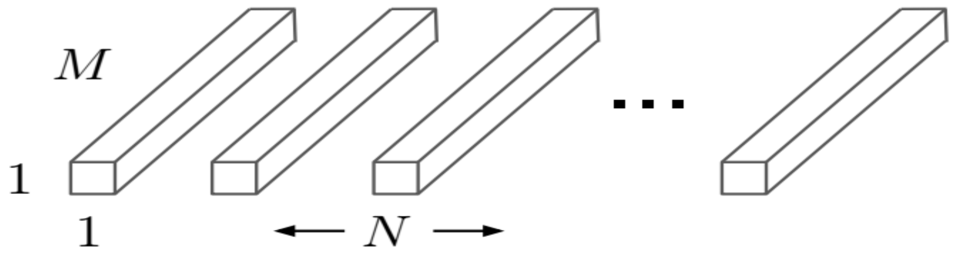
MobileNet is a deep Convolutional Neural Network developed by Google based on the Xception Architecture. The core layer of MobileNet and Xception architecture is depthwise seperable filters, known as Depthwise Seperable Convolution.



(a) Standard Convolution Filters



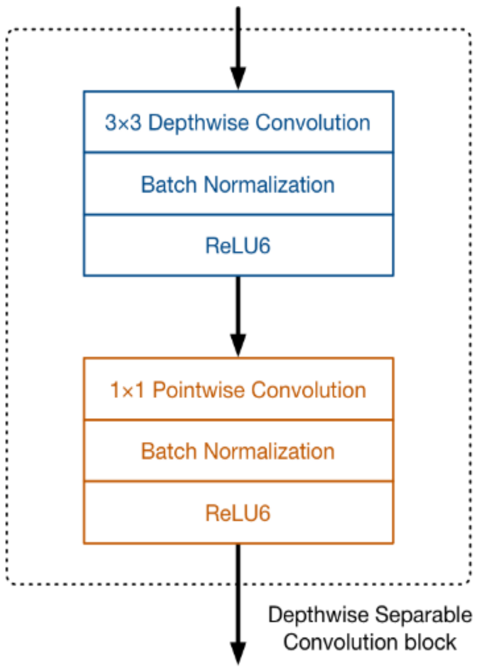
(b) Depthwise Convolutional Filters

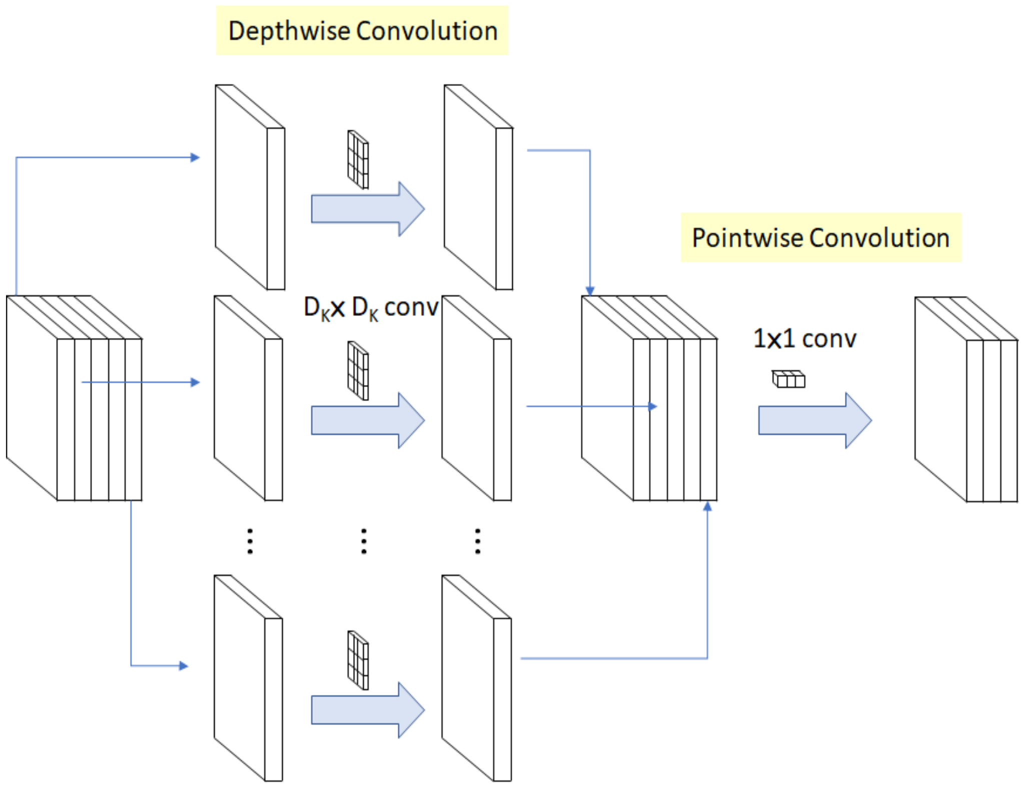


(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Depthwise separable convolutions which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1×11×1 convolution called a pointwise convolution. In MobileNet, the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×11×1 convolution to combine the outputs the depthwise convolution.

the required classes and froze the initial convolutional layers for transfer learning.


3×3 Depthwise Convolution
Batch Normalization
ReLU6
1×1 Pointwise Convolution
Batch Normalization
ReLU6
Depthwise Separable Convolution block



In this project, The output (softmax) layer of MobileNet is modified according to the required classes and froze the initial convolutional layers for transfer learning.

6.2.4 Export the model to Tflite

TensorFlow Lite is a set of tools to help developers run TensorFlow models on mobile, embedded, and IoT devices. It enables on-device machine learning inference with low latency and a small binary size.

The Model thus saved is converted into Tflite and then embedded into Android Application for running inference using Tflite interpreter.

CHAPTER 7
CODING

7.1 Activity CropRecommendation.java

```
package com.anurag.myapp;


import android.Manifest;

import android.annotation.SuppressLint;

import android.content.Context;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.location.Location;

import android.location.LocationManager;

import android.os.Bundle;

import android.os.Looper;

import android.provider.Settings;

import android.widget.ArrayAdapter;

import android.widget.ListAdapter;

import android.widget.ListView;

import android.widget.TextView;

import android.widget.Toast;


import androidx.annotation.NonNull;

import androidx.appcompat.app.AppCompatActivity;

import androidx.core.app.ActivityCompat;
import com.android.volley.Request;

import com.android.volley.RequestQueue;

import com.android.volley.toolbox.JsonObjectRequest;

import com.android.volley.toolbox.Volley;

import com.google.android.gms.location.FusedLocationProviderClient;

import com.google.android.gms.location.LocationCallback;

import com.google.android.gms.location.LocationRequest;

import com.google.android.gms.location.LocationResult;

import com.google.android.gms.location.LocationServices;


import org.json.JSONException;


import java.util.ArrayList;

import java.util.Calendar;


public class CropRecommendation extends AppCompatActivity {

    FusedLocationProviderClient mFusedLocationClient;

double latitude,longitude,temperature;

int month;

    ArrayList<String>crops;

    String soil;

    TextView textView;

    ListView listView;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_crop_recommendation);
textView = findViewById(R.id.textView1);
listView = findViewById(R.id.listView);
mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this);
        getLocation();


    }


private void getInputs() {


// Method to obtain all the Inputs without user intervention
String url =
"https://api.worldweatheronline.com/premium/v1/weather.ashx?q="+latitude+","+l
ongitude+"&key=d7cd298c36d842cd985130454202009&format=json&mca=yes&fx
=no&cc=no";
        String url1 =
"https://rest.soilgrids.org/query?lon="+longitude+"&lat="+latitude;
        RequestQueue q = Volley.newRequestQueue(this);
        JsonObjectRequest jsonObjectRequest = new
JsonObjectRequest(Request.Method.GET, url1, null, response -> {
try {
double clay =
response.getJSONObject("properties").getJSONObject("CLYPPT").getJSONObject("
M").getDouble("sl1");
double sand =
response.getJSONObject("properties").getJSONObject("SNDPPT").getJSONObject("
M").getDouble("sl1");
double silt =
response.getJSONObject("properties").getJSONObject("SLTPPT").getJSONObject("
M").getDouble("sl1");
if (sand>=0 && sand<=45 && silt>=0 && silt<=40 && clay>=40 && clay<=100) {
soil = "CLAY";
            }
else if (sand>=0 && sand<=20 && silt>=40 && silt<=60 && clay>=40 && clay<=60) {
soil = "SILTY CLAY";
            }
else if (sand>=45 && sand<=65 && silt>=0 && silt<=20 && clay>=35 && clay<=55) {
soil = "SANDY CLAY";
            }
else if (sand>=0 && sand<=20 && silt>=40 && silt<=73 && clay>=27 && clay<=40) {
soil = "SILTY CLAY LOAM";
            }
else if (sand>=45 && sand<=80 && silt>=0 && silt<=28 && clay>=20 && clay<=35) {
soil = "SANDY CLAY LOAM";
            }
else if (sand>=20 && sand<=45 && silt>=15 && silt<53 && clay>=27 && clay<=40) {
soil = "CLAY LOAM";
```

```
            }
else if (sand>=0 && sand<=20 && silt>=80 && silt<=100 && clay>=0 && clay<=12) {
soil = "SILT";
            }
else if ((sand>=20 && sand<=50 && silt>=50 && silt<=80 && clay>=0 && clay<=27)
|| (sand>=0 && sand<=8 && silt>=80 && silt<=88 && clay>=12 && clay<=20)) {
soil = "SILT LOAM";
            }
else if (sand>=23 && sand<=52 && silt>=28 && silt<=50 && clay>=7 && clay<=27) {
soil = "LOAM";
            }
else if ((sand>=52 && sand<=70 && silt>=10 && silt<=48 && clay>=0 && clay<=20)
|| (sand>=43 && sand<=52 && silt>=43 && silt<=50 && clay>=0 && clay<=7) ||
(sand>=70 && sand<=85 && silt>=0 && silt<=15 && clay>=15 && clay<=20)) {
soil = "SANDY LOAM";
            }
else if (sand>=70 && sand<=85 && silt>=0 && silt<=30 && clay>=0 && clay<=15) {
soil = "LOAMY SAND";
            }
else {
soil = "SAND";
            }
textView.setText(soil);
            JsonObjectRequest jsonObjectRequest1 = new
JsonObjectRequest(Request.Method.GET, url, null, responses -> {
```

```
try {
final Calendar c = Calendar.getInstance();
month = c.get(Calendar.MONTH);
temperature =
responses.getJSONObject("data").getJSONArray("ClimateAverages").getJSONObject
(0).getJSONArray("month").getJSONObject(month).getDouble("avgTemp");
                        getCrops();
                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }, error -> {


                });
q.add(jsonObjectRequest1);
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }, error -> {


        });
        q.add(jsonObjectRequest);
    }
@SuppressWarnings("StringEquality")
private void getCrops() {
```

```
// Method to obtain the list of Recommended Crops

crops = new ArrayList<>();

if ((temperature>=18 &&temperature<=37) && (soil=="SANDY LOAM" ||
soil=="LOAMY SAND" || soil=="SANDY CLAY LOAM") && (month==3 ||
month==4 || month==5 || month==6)) {
crops.add("Cotton");
        }
if ((temperature>=20 &&temperature<=37) && (soil=="CLAY LOAM" ||
soil=="CLAY" || soil=="LOAM" || soil=="SILTY CLAY" || soil=="SILTY CLAY
LOAM") && (month==3 || month==4 || month==5 || month==6 || month==7)) {
crops.add("Paddy");
        }
if ((temperature>=16 &&temperature<=30) && (soil=="SANDY" || soil=="SILTY
CLAY LOAM" || soil=="LOAM" || soil=="SANDY LOAM" || soil=="SANDY
CLAY LOAM" || soil=="CLAY LOAM" || soil=="SANDY CLAY") && (month==0
|| month==1 || month==5 || month==6 || month==8 || month==9 || month==10)) {
crops.add("Maize");
        }
if ((temperature>=18 &&temperature<=25) && (soil=="LOAM" || soil=="SANDY
LOAM" || soil=="SANDY CLAY LOAM" || soil=="CLAY" || soil=="CLAY
LOAM") && (month==8 || month==9 || month==10 || month==11 || month==0)) {
crops.add("Wheat");
        }
if ((temperature>=20 &&temperature<=30) && (soil=="CLAY" || soil=="LOAM" ||
soil=="CLAY LOAM" || soil=="SANDY LOAM") && (month==3 || month==4 ||
month==5 || month==6 || month==7)) {
crops.add("Bajra");
        }
if ((temperature>=23 &&temperature<=33) && (soil=="CLAY LOAM" ||
soil=="LOAM" || soil=="SANDY LOAM") && (month==0 || month==1 || month==2
|| month==3 || month==4 || month==5 || month==6 || month==7 || month==8 ||
month==9 || month==10 || month==11)) {
crops.add("Jowar");
        }
if ((temperature>=15 &&temperature<=32) && (soil=="SANDY LOAM" ||
soil=="SILTY CLAY LOAM" || soil=="LOAM" || soil=="CALY LOAM")
&&(month==1 || month==2 || month==5 || month==6)) {
crops.add("Soybean");
        }
if ((temperature>=20 &&temperature<=34) && (soil=="LOAM" || soil=="SANDY
CLAY LOAM" || soil=="SANDY LOAM" || soil=="CLAY LOAM" ||
soil=="LOAMY SAND") && (month==3 || month==4 || month==5 || month==6 ||
month==9 || month==10 || month==11)) {
crops.add("Ragi");
        }
if ((temperature>=15 &&temperature<=28) && (soil=="SANDY LOAM" ||
soil=="LOAM" || soil=="SANDY CLAY LOAM") && (month==0 || month==1 ||
month==9 || month==10 || month==11)) {
crops.add("Sunflower");
        }
```

```
if ((temperature>=20 &&temperature<=30) && (soil=="SANDY LOAM" ||
soil=="LOAM" || soil=="SANDY CLAY LOAM") && (month==0 || month==5 ||
month==6 || month==7 || month==10 || month==11)) {
crops.add("Groundnut");
        }
if ((temperature>=14 &&temperature<=30) && (soil=="SANDY LOAM" ||
soil=="LOAMY SAND" || soil=="LOAM" || soil=="SILT LOAM" || soil=="SANDY
CLAY LOAM" || soil=="SANDY CLAY" || soil=="SILTY CLAY" || soil=="SILTY
CLAY LOAM" || soil=="SILT" || soil=="CLAY LOAM") && (month==2 ||
month==3 || month==5 || month==6 || month==9 || month==10)) {
crops.add("Tomato");
        }
if ((temperature>=20 &&temperature<=30) && (soil=="LOAM" || soil=="SANDY
LOAM" || soil=="SILT LOAM") && (month==1 || month==2 || month==4 ||
month==5 || month==6 || month==8 || month==9 || month==10)) {
crops.add("Chilli");
        }
if ((temperature>=12 &&temperature<=24) && (soil=="LOAM" || soil=="SANDY
LOAM" || soil=="SILT LOAM" || soil=="CLAY LOAM") && (month==0 ||
month==5 || month==6 || month==7 || month==9 || month==10 || month==11)) {
crops.add("Brinjal");
        }
if ((temperature>=20 &&temperature<=35) && (soil=="SANDY LOAM" ||
soil=="CLAY LOAM" || soil=="LOAM" || soil=="SANDY CLAY LOAM") &&
(month==3 || month==4 || month==5 || month==6)) {
```

```
crops.add("Turmeric");

        }

if ((temperature>=22 &&temperature<=35) && (soil=="CLAY LOAM" ||

soil=="SANDY LOAM" || soil=="LOAM") && (month==0 || month==1 || month==2

|| month==5 || month==6 || month==7)) {

crops.add("Lady's Finger");

        }

if ((temperature>=22 &&temperature<=35) && (soil=="SANDY LOAM" ||

soil=="LOAM") && (month==1 || month==2 || month==5 || month==6)) {

crops.add("Green Gram");

        }

if ((temperature>=22 &&temperature<=35) && (soil=="SANDY LOAM" ||

soil=="LOAM") && (month==1 || month==2 || month==5 || month==3 || month==6))

{

crops.add("Black Gram");

        }

if ((temperature>=18 &&temperature<=35) && (soil=="SANDY LOAM" ||

soil=="LOAM" || soil=="SANDY CLAY LOAM" || soil=="CLAY LOAM") &&

(month==0 || month==1 || month==3 || month==4 || month==5 || month==6)) {

crops.add("Bottle Gourd");

        }

if ((temperature>=5 &&temperature<=19) && (soil=="SANDY LOAM" ||

soil=="LOAMY SAND" || soil=="LOAM" || soil=="SILT LOAM" || soil=="SANDY

CLAY LOAM" || soil=="SANDY CLAY" || soil=="SILTY CLAY" || soil=="SILTY

CLAY LOAM" || soil=="SILT" || soil=="CLAY LOAM") && (month==2 ||
month==3 || month==4 || month==9 || month==10)) {
crops.add("Pea");
        }
if ((temperature>=5 &&temperature<=27) && (soil=="SANDY" || soil=="LOAMY
SAND" || soil=="SANDY LOAM" || soil=="LOAM" || soil=="CLAY LOAM" ||
soil=="SANDY CLAY LOAM") && (month==0 || month==9 || month==10 ||
month==11)) {
crops.add("Barley");
        }
if ((temperature>=5 &&temperature<=25) && (soil=="CLAY LOAM" ||
soil=="SILT LOAM" || soil=="SILTY CLAY" || soil=="CLAY" || soil=="SANDY
CLAY" || soil=="SILTY CLAY LOAM" || soil=="SANDY CLAY LOAM" ||
soil=="LOAM" || soil=="SANDY LOAM") && (month==9 || month==10 ||
month==11)) {
crops.add("Oats");
        }
if ((temperature>=18 &&temperature<=30) && (soil=="LOAM" || soil=="SILT
LOAM" || soil=="SILTY CLAY LOAM" || soil=="SANDY LOAM" ||
soil=="SANDY CLAY LOAM" || soil=="CLAY LOAM") && (month==5 ||
month==6 || month==9 || month==10)) {
crops.add("Coriander");
        }
if ((temperature>=20 &&temperature<=30) && (soil=="SANDY LOAM" ||
soil=="CLAY LOAM" || soil=="LOAM" || soil=="SANDY CLAY LOAM") &&
(month==9 || month==10)) {
```

```
crops.add("Bengal Gram");
        }
if ((temperature>=13 &&temperature<=25) && (soil=="CLAY LOAM" ||
soil=="SILT LOAM" || soil=="SILTY CLAY" || soil=="CLAY" || soil=="SANDY
CLAY" || soil=="SILTY CLAY LOAM" || soil=="SANDY CLAY LOAM" ||
soil=="LOAM" || soil=="SANDY LOAM") && (month==0 || month==4 || month==5
|| month==6 || month==7 || month==8 || month==9 || month==10 || month==11)) {
crops.add("Onion");
        }
if ((temperature>=10 &&temperature<=30) && (soil=="SANDY CLAY" ||
soil=="SANDY CLAY LOAM" || soil=="CALY LOAM" || soil=="LOAM" ||
soil=="SANDY LOAM") && (month==5 || month==6 || month==9 || month==10)) {
crops.add("Garlic");
        }
if ((temperature>=15 &&temperature<=26) && (soil=="SANDY" || soil=="SANDY
LOAM" || soil=="LOAMY SAND" || soil=="LOAM" || soil=="SILT LOAM") &&
(month==7 || month==8 || month==9 || month==10 || month==11)) {
crops.add("Carrot");
        }
if ((temperature>=13 &&temperature<=26) && (soil=="LOAM" || soil=="CLAY
LOAM" || soil=="SANDY CLAY LOAM" || soil=="SANDY LOAM") &&
(month==4 || month==5 || month==6 || month==7 || month==8 || month==9 ||
month==10)) {
crops.add("Cauliflower");
        }
```

```
if ((temperature>=10 &&temperature<=30) && (soil=="LOAM" || soil=="SANDY
LOAM" || soil=="SILT LOAM" || soil=="SANDY CLAY LOAM") && (month==0
|| month==5 || month==6 || month==7 || month==9 || month==10)) {
crops.add("Potato");
        }
if ((temperature>=10 &&temperature<=30) && (soil=="LOAM" || soil=="CLAY
LOAM" || soil=="SANDY CLAY LOAM") && (month==8 || month==9 ||
month==10 || month==11)) {
crops.add("Rapeseed");
        }
if ((temperature>=18 &&temperature<=33) && (soil=="SANDY LOAM" ||
soil=="LOAM" || soil=="SANDY CLAY LOAM" || soil=="CLAY LOAM" ||
soil=="LOAMY SAND" || soil=="SILT LOAM" || soil=="SILTY CLAY LOAM")
&& (month==2 || month==1 || month==0 || month==10 || month==11)) {
crops.add("Watermelon");
        }
if ((temperature>=18 &&temperature<=30) && (soil=="SANDY LOAM" ||
soil=="LOAMY SAND" || soil=="SANDY CLAY LOAM" || soil=="LOAM" ||
soil=="SILT") && (month==0 || month==1 || month==2 || month==3 || month==10 ||
month==11)) {
crops.add("Muskmelon");
        }
if ((temperature>=18 &&temperature<=28) && (soil=="LOAM" || soil=="LOAMY
SAND" || soil=="SANDY LOAM" || soil=="SANDY CLAY LOAM") &&
(month==0 || month==1 || month==2 || month==8 || month==9 || month==10 ||
month==11)) {
crops.add("Pumpkin");
        }
if ((temperature>=15 &&temperature<=25) && (soil=="CLAY LOAM" ||
soil=="SANDY CLAY LOAM" || soil=="SILT LOAM" || soil=="LOAM" ||
soil=="SANDY LOAM" || soil=="LOAMY SAND" || soil=="SANDY") &&
(month==0 || month==1 || month==2 || month==3 || month==5)) {
crops.add("Cucumber");
        }
if ((temperature>=18 &&temperature<=28) && (soil=="SANDY LOAM" ||
soil=="LOAM" || soil=="CLAY LOAM" || soil=="SANDY CLAY LOAM" ||
soil=="SANDY CLAY" || soil=="CLAY" || soil=="SILT LOAM" || soil=="SILTY
CLAY LOAM") && (month==0 || month==1 || month==2 || month==5 || month==6))
{
crops.add("Bitter Gourd");
        }
if ((temperature>=20 &&temperature<=35) && (soil=="SANDY LOAM" ||
soil=="LOAM" || soil=="CLAY LOAM" || soil=="SILT" || soil=="SILTY CLAY
LOAM" || soil=="SILT LOAM" || soil=="SANDY CLAY LOAM") && (month==1
|| month==2 || month==3 || month==0 || month==5 || month==6)) {
crops.add("Ridge Gourd");
        }
if ((temperature>=15 &&temperature<=22) && (soil=="SANDY LOAM" ||
soil=="LOAM" || soil=="SANDY CLAY LOAM" || soil=="CLAY LOAM" ||
soil=="SANDY CLAY" || soil=="CLAY") && (month==8 || month==9 ||
month==10)) {

crops.add("Cabbage");

        }

if ((temperature>=24 &&temperature<=38) && (soil=="LOAM" || soil=="SANDY

LOAM" || soil=="SANDY CLAY LOAM" || soil=="CLAY LOAM" || soil=="SILT

LOAM") && (month==1 || month==2 || month==3 || month==4)) {

crops.add("Jute");

        }

if ((temperature>=21 &&temperature<=38) && (soil=="SANDY LOAM" ||

soil=="LOAM" || soil=="LOAMY SAND" || soil=="SANDY CLAY LOAM" ||

soil=="CLAY LOAM" || soil=="CLAY" || soil=="SANDY CLAY" || soil=="SILT

LOAM" || soil=="SILTY CLAY LOAM") && (month==1 || month==2 || month==3 ||

month==4)) {

crops.add("Ginger");

        }

if ((temperature>=20 &&temperature<=30) && (soil=="SANDY CLAY LOAM" ||

soil=="CLAY LOAM" || soil=="LOAM" || soil=="SANDY LOAM") && (month==8

|| month==9 || month==10 || month==11)) {

crops.add("Capsicum");

        }

if ((temperature>=10 &&temperature<=25) && (soil=="SILT" || soil=="SILTY

CLAY" || soil=="SANDY" || soil=="SANDY LOAM" || soil=="LOAM" ||

soil=="LOAMY SAND" || soil=="SANDY CLAY LOAM" || soil=="CLAY LOAM"

|| soil=="CLAY" || soil=="SANDY CLAY" || soil=="SILT LOAM" || soil=="SILTY

CLAY LOAM") && (month==8 || month==9 || month==10)) {
```

```
crops.add("Mustard");
        }
        ListAdapter listAdapter = new ArrayAdapter<>(this,
android.R.layout.simple_list_item_1, crops);
listView.setAdapter(listAdapter);
    }


@SuppressLint("MissingPermission")
private void getLocation(){


// Obtain Lat Long Coordinates
if (checkPermissions()) {
if (isLocationEnabled()) {
mFusedLocationClient.getLastLocation().addOnCompleteListener(
                        task -> {
                            Location location = task.getResult();
if (location == null) {
                                requestNewLocationData();
                            } else {
latitude = location.getLatitude();
longitude = location.getLongitude();
                                getInputs();
                            }
                        }
                );
```

```
            } else {
                Toast.makeText(this, "Turn on location", Toast.LENGTH_LONG).show();
                Intent intent = new
Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS);
                startActivity(intent);
            }
        } else {
            requestPermissions();
        }
    }


@SuppressLint("MissingPermission")
private void requestNewLocationData(){

// Request Device Location
LocationRequest mLocationRequest = new LocationRequest();
        mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        mLocationRequest.setInterval(0);
        mLocationRequest.setFastestInterval(0);
        mLocationRequest.setNumUpdates(1);

mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this);
mFusedLocationClient.requestLocationUpdates(
                mLocationRequest, mLocationCallback,
                Looper.myLooper()
        );

    }

private LocationCallback mLocationCallback = new LocationCallback() {
@SuppressLint("SetTextI18n")
@Override
public void onLocationResult(LocationResult locationResult) {
            Location mLastLocation = locationResult.getLastLocation();
latitude = mLastLocation.getLatitude();
longitude = mLastLocation.getLongitude();
        }
    };

private boolean checkPermissions() {
return (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) ==
PackageManager.PERMISSION_GRANTED) &&
                (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) ==
PackageManager.PERMISSION_GRANTED);
    }

private void requestPermissions() {
        ActivityCompat.requestPermissions(
this,
new String[]{Manifest.permission.ACCESS_COARSE_LOCATION,
Manifest.permission.ACCESS_FINE_LOCATION},
50
);
    }

private boolean isLocationEnabled() {
        LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
assert locationManager != null;
return locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER) ||
locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER);
    }

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
super.onRequestPermissionsResult(requestCode, permissions, grantResults);
if (requestCode == 50) {
if (grantResults.length >0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                getLocation();
            }
        }
    }
}
```

7.2 Activity DiseaseDetection.java

```
package com.anurag.myapp;


import android.Manifest;
import android.annotation.SuppressLint;
import android.content.pm.PackageManager;
import android.content.res.AssetFileDescriptor;
import android.content.res.AssetManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Rect;
import android.graphics.YuvImage;
import android.os.Bundle;
import android.widget.TextView;


import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;
import androidx.camera.core.Camera;
import androidx.camera.core.CameraSelector;
import androidx.camera.core.CameraX;
```

```java
import androidx.camera.core.ImageAnalysis;

import androidx.camera.core.ImageProxy;

import androidx.camera.core.Preview;

import androidx.camera.lifecycle.ProcessCameraProvider;

import androidx.camera.view.PreviewView;

import androidx.core.app.ActivityCompat;

import androidx.core.content.ContextCompat;


import com.google.common.util.concurrent.ListenableFuture;


import org.tensorflow.lite.Interpreter;


import java.io.BufferedReader;

import java.io.ByteArrayOutputStream;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.nio.ByteBuffer;

import java.nio.ByteOrder;

import java.nio.MappedByteBuffer;

import java.nio.channels.FileChannel;

import java.util.ArrayList;

import java.util.concurrent.ExecutionException;

import java.util.concurrent.ExecutorService;

import java.util.concurrent.Executors;
```

```
public class DiseaseDetection extends AppCompatActivity {

    AssetManager assetManager;

    ByteBuffer imgData;

int[] intValues = new int[224 * 224];

static final int IMAGE_MEAN = 128;

static final float IMAGE_STD = 128.0f;

    ArrayList<String>labelList;

    ImageAnalysis imageAnalysis;

    Interpreter interpreter;

    PreviewView previewView;

    TextView textView;

    ListenableFuture<ProcessCameraProvider>cameraProviderFuture;

    ExecutorService executor;

    CameraSelector cameraSelector;

    Preview preview;

    Camera camera;

    String output;


@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_disease_detection);

previewView = findViewById(R.id.previewView);

textView = findViewById(R.id.textView);
labelList = new ArrayList<>();
cameraProviderFuture = ProcessCameraProvider.getInstance(this);
executor = Executors.newSingleThreadExecutor();
assetManager = getAssets();
        startCamera();
    }


private void startCamera() {


// Method to Start Live feed
if(checkPermissions()) {
cameraProviderFuture.addListener(() -> {
try {
                    ProcessCameraProvider cameraProvider = cameraProviderFuture.get();
                    bindPreview(cameraProvider);
                } catch (InterruptedException | ExecutionException e) {
                }
            }, ContextCompat.getMainExecutor(this));
        }
else {
            ActivityCompat.requestPermissions(DiseaseDetection.this,new
String[]{Manifest.permission.CAMERA},100);
        }
    }
```

```
@SuppressLint("RestrictedApi")
private void bindPreview(@NonNull ProcessCameraProvider cameraProvider) {

//Method to create preview and bind it to the preview lifecycle
CameraX.unbindAll();
preview = new Preview.Builder().build();
imageAnalysis = new
ImageAnalysis.Builder().setBackgroundExecutor(executor).build();
imageAnalysis.setAnalyzer(executor, image -> {
            Bitmap bitmap;
            bitmap = DiseaseDetection.this.imageProxyToBitmap(image);
try {
                classify(bitmap);
                image.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        });
cameraSelector = new
CameraSelector.Builder().requireLensFacing(CameraSelector.LENS_FACING_BACK).
build();
camera = cameraProvider.bindToLifecycle(this, cameraSelector, preview,
imageAnalysis);
preview.setSurfaceProvider(previewView.createSurfaceProvider(camera.getCameraInfo
()));
```

```
    }

private void classify(Bitmap bitmap) throws IOException {

//Method to classify image
BufferedReader reader = new BufferedReader(new
InputStreamReader(assetManager.open("labels.txt")));
        String line;
labelList.clear();
while ((line = reader.readLine()) != null) {
labelList.add(line);
        }
        reader.close();
float[][] labelProbArray = new float[1][labelList.size()];
//noinspection deprecation
interpreter = new Interpreter(loadModelFile());
//noinspection PointlessArithmeticExpression
imgData = ByteBuffer.allocateDirect(4 * 1 * 224 * 224 * 3);
imgData.order(ByteOrder.nativeOrder());
imgData = bitmapToByteBuffer(bitmap);
interpreter.run(imgData, labelProbArray);
interpreter.close();
int index = getMaxIndex(labelProbArray,labelList.size());
output = labelList.get(index);
        runOnUiThread(() ->textView.setText(output));
    }

private int getMaxIndex(float[][] arr,int size) {

// Method to obtain the label with highest probability
int maxInd = 0;
float max = arr[0][0];
for (int i = 0; i<1; i++) {
for (int j = 0; j<size; j++) {
if(arr[i][j] > max) {
                maxInd = j;
                max = arr[i][j];
            }
        }
    }
return maxInd;
    }

private ByteBuffer bitmapToByteBuffer(Bitmap bitmap) {

// Method to create ByteBuffer from Bitmap
imgData.rewind();
bitmap.getPixels(intValues,0,bitmap.getWidth(),0,0,bitmap.getWidth(),bitmap.getHeight
());
int pixel = 0;
```

```
for (int i=0; i<224 ; ++i){
for (int j=0; j<224 ; ++j){
final int val = intValues[pixel++];
imgData.putFloat((((val >>16) &0xFF)-IMAGE_MEAN)/IMAGE_STD);
imgData.putFloat((((val >>8) &0xFF)-IMAGE_MEAN)/IMAGE_STD);
imgData.putFloat((((val) &0xFF)-IMAGE_MEAN)/IMAGE_STD);
            }
        }
return imgData;
    }

private MappedByteBuffer loadModelFile() throws IOException {

// Method to include the Tflite model
AssetFileDescriptor fileDescriptor = assetManager.openFd("model.tflite");
        FileInputStream inputStream = new
FileInputStream(fileDescriptor.getFileDescriptor());
        FileChannel fileChannel = inputStream.getChannel();
long startOffset = fileDescriptor.getStartOffset();
long declaredLength = fileDescriptor.getDeclaredLength();
return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset,
declaredLength);
    }

private Bitmap imageProxyToBitmap(ImageProxy image) {
```

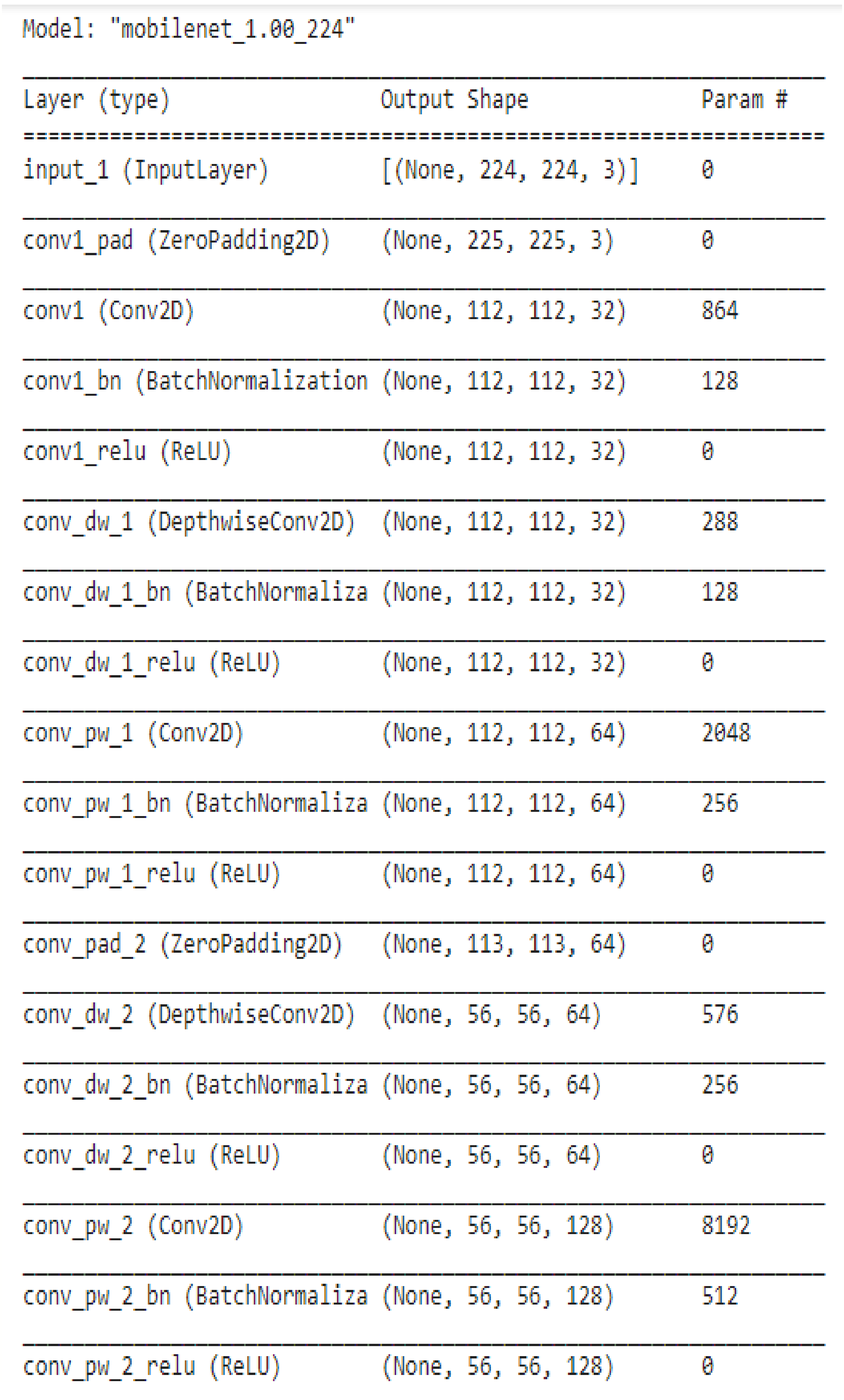
```
// Method to create Bitmap
ImageProxy.PlaneProxy planeProxy = image.getPlanes()[0];
        ByteBuffer yBuffer = planeProxy.getBuffer();
        planeProxy = image.getPlanes()[1];
        ByteBuffer uBuffer = planeProxy.getBuffer();
        planeProxy = image.getPlanes()[2];
        ByteBuffer vBuffer = planeProxy.getBuffer();
int ySize = yBuffer.remaining();
int uSize = uBuffer.remaining();
int vSize = vBuffer.remaining();
byte[] b = new byte[ySize + uSize + vSize];
        yBuffer.get(b,0,ySize);
        vBuffer.get(b,ySize,vSize);
        uBuffer.get(b,ySize+vSize,uSize);
        YuvImage yuvImage = new YuvImage(b, 17,image.getWidth(),image.getHeight(),
null);
        ByteArrayOutputStream out = new ByteArrayOutputStream();
        yuvImage.compressToJpeg(new Rect(0, 0, yuvImage.getWidth(),
yuvImage.getHeight()), 100, out);
byte[] imageBytes = out.toByteArray();
        Bitmap bit = BitmapFactory.decodeByteArray(imageBytes,0,imageBytes.length);
return (Bitmap.createScaledBitmap(bit,224,224,false));
    }
```

```
private boolean checkPermissions() {
return ActivityCompat.checkSelfPermission(DiseaseDetection.this,
Manifest.permission.CAMERA) == PackageManager.PERMISSION_GRANTED;
   }
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
super.onRequestPermissionsResult(requestCode, permissions, grantResults);
if (requestCode == 100) {
if (grantResults.length >0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            startCamera();
        }
     }
   }
}
```

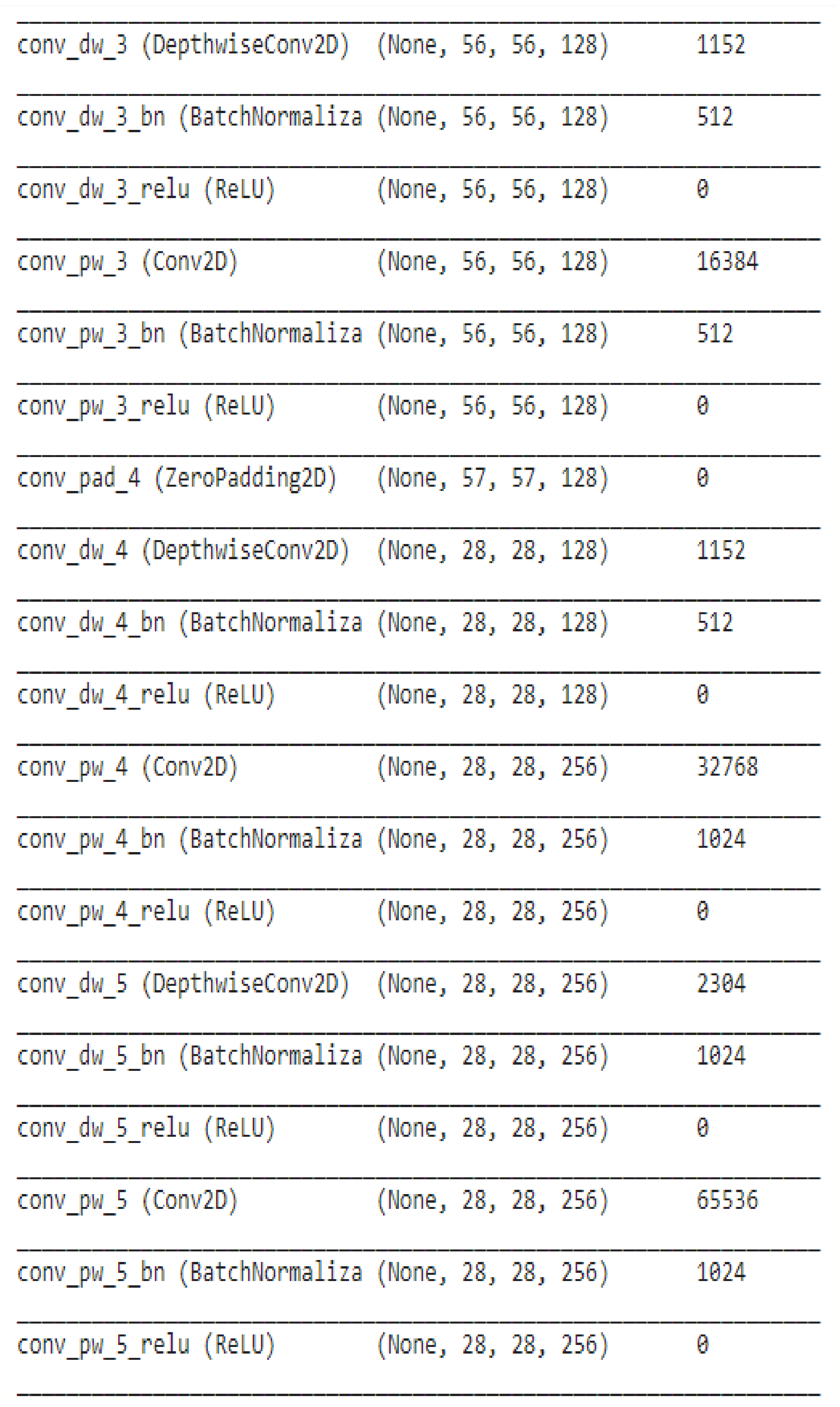
7.3 DiseaseClassifier.py

```
import tensorflow as tf
import keras
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.mobilenet import MobileNet
from keras.optimizers import Adam
from keras.models import Model,load_model
from keras.layers import Dense,GlobalAveragePooling2D,Dropout,Reshape,Conv2D
from keras.applications.mobilenet import preprocess_input
import h5py
%matplotlib inline
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix,plot_confusion_matrix,ConfusionMatrixDi
splay
import numpy as np

# Augment the existing dataset
ImageDataGenerator(
    featurewise_center=True,
    samplewise_center=True,
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    brightness_range=None,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
    rescale=None,
    preprocessing_function=preprocess_input,
    validation_split=0.1,
)
```

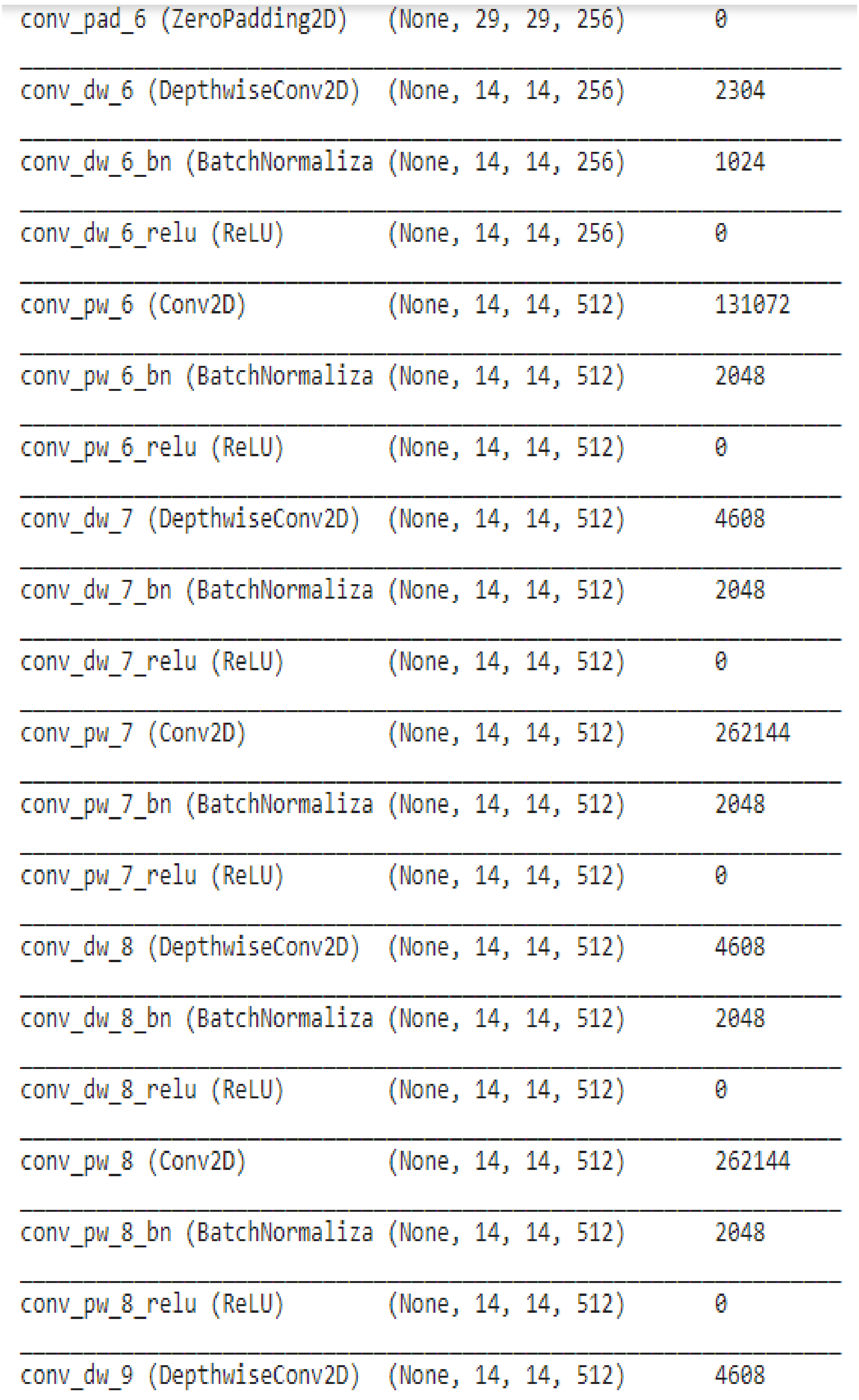
```
# training and validation split
gen = ImageDataGenerator(preprocessing_function=preprocess_input,validation_split=0.
1)
gen.flow_from_directory('drive/My Drive/disease_data/train')
test_gen = ImageDataGenerator(preprocessing_function=preprocess_input)

## Creating the Model
# Modifying Output layer and transfering weights
mobile = MobileNet(weights=None,include_top=True,classes=46)
mobile2 = MobileNet(weights='imagenet',include_top = True)
for i,j in zip(mobile.layers[:-5],mobile2.layers[:-5]):
  i.set_weights(j.get_weights())
for i in mobile.layers[:24]:
  i.trainable=False
for i in mobile.layers[24:]:
  i.trainable=True
# Create data generators
train_data = gen.flow_from_directory('drive/My Drive/disease_data/train',target_size=(22
4,224),batch_size=32,class_mode='categorical',color_mode='rgb',subset='training')
val_data = gen.flow_from_directory('drive/My Drive/disease_data/train',target_size=(224
,224),batch_size=32,class_mode='categorical',color_mode='rgb',subset='validation')
test_data = test_gen.flow_from_directory('drive/My Drive/disease_data/test',target_size=(
224,224),batch_size=32,shuffle=False)
```



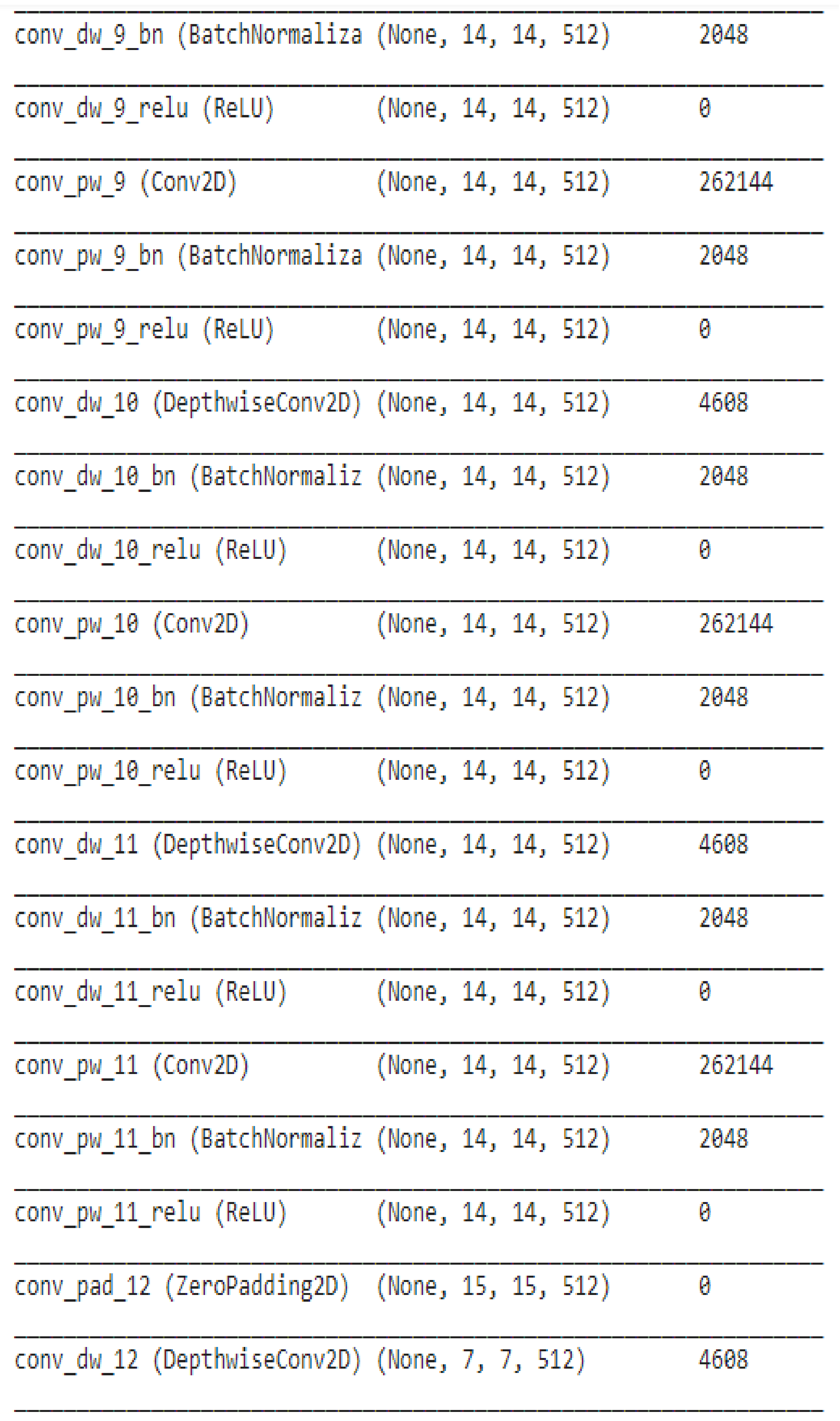
```
Model: "mobilenet_1.00_224"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 225, 225, 3)       0
_________________________________________________________________
conv1 (Conv2D)               (None, 112, 112, 32)      864
_________________________________________________________________
conv1_bn (BatchNormalization (None, 112, 112, 32)      128
_________________________________________________________________
conv1_relu (ReLU)            (None, 112, 112, 32)      0
_________________________________________________________________
conv_dw_1 (DepthwiseConv2D)  (None, 112, 112, 32)      288
_________________________________________________________________
conv_dw_1_bn (BatchNormaliza (None, 112, 112, 32)      128
_________________________________________________________________
conv_dw_1_relu (ReLU)        (None, 112, 112, 32)      0
_________________________________________________________________
conv_pw_1 (Conv2D)           (None, 112, 112, 64)      2048
_________________________________________________________________
conv_pw_1_bn (BatchNormaliza (None, 112, 112, 64)      256
_________________________________________________________________
conv_pw_1_relu (ReLU)        (None, 112, 112, 64)      0
_________________________________________________________________
conv_pad_2 (ZeroPadding2D)   (None, 113, 113, 64)      0
_________________________________________________________________
conv_dw_2 (DepthwiseConv2D)  (None, 56, 56, 64)        576
_________________________________________________________________
conv_dw_2_bn (BatchNormaliza (None, 56, 56, 64)        256
_________________________________________________________________
conv_dw_2_relu (ReLU)        (None, 56, 56, 64)        0
_________________________________________________________________
conv_pw_2 (Conv2D)           (None, 56, 56, 128)       8192
_________________________________________________________________
conv_pw_2_bn (BatchNormaliza (None, 56, 56, 128)       512
_________________________________________________________________
conv_pw_2_relu (ReLU)        (None, 56, 56, 128)       0
```



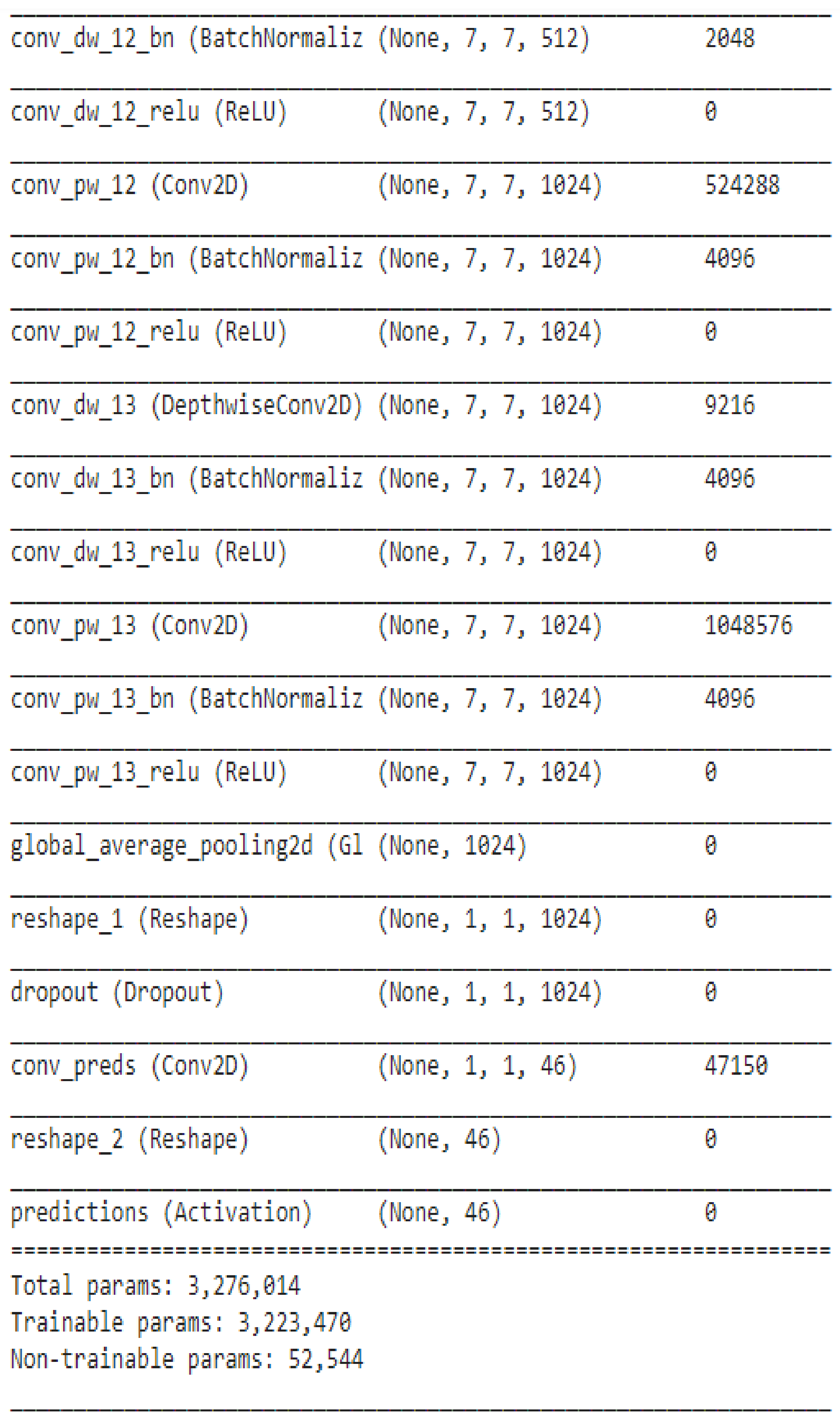
```
_________________________________________________________________
conv_dw_3 (DepthwiseConv2D)  (None, 56, 56, 128)       1152
_________________________________________________________________
conv_dw_3_bn (BatchNormaliza (None, 56, 56, 128)       512
_________________________________________________________________
conv_dw_3_relu (ReLU)        (None, 56, 56, 128)       0
_________________________________________________________________
conv_pw_3 (Conv2D)           (None, 56, 56, 128)       16384
_________________________________________________________________
conv_pw_3_bn (BatchNormaliza (None, 56, 56, 128)       512
_________________________________________________________________
conv_pw_3_relu (ReLU)        (None, 56, 56, 128)       0
_________________________________________________________________
conv_pad_4 (ZeroPadding2D)   (None, 57, 57, 128)       0
_________________________________________________________________
conv_dw_4 (DepthwiseConv2D)  (None, 28, 28, 128)       1152
_________________________________________________________________
conv_dw_4_bn (BatchNormaliza (None, 28, 28, 128)       512
_________________________________________________________________
conv_dw_4_relu (ReLU)        (None, 28, 28, 128)       0
_________________________________________________________________
conv_pw_4 (Conv2D)           (None, 28, 28, 256)       32768
_________________________________________________________________
conv_pw_4_bn (BatchNormaliza (None, 28, 28, 256)       1024
_________________________________________________________________
conv_pw_4_relu (ReLU)        (None, 28, 28, 256)       0
_________________________________________________________________
conv_dw_5 (DepthwiseConv2D)  (None, 28, 28, 256)       2304
_________________________________________________________________
conv_dw_5_bn (BatchNormaliza (None, 28, 28, 256)       1024
_________________________________________________________________
conv_dw_5_relu (ReLU)        (None, 28, 28, 256)       0
_________________________________________________________________
conv_pw_5 (Conv2D)           (None, 28, 28, 256)       65536
_________________________________________________________________
conv_pw_5_bn (BatchNormaliza (None, 28, 28, 256)       1024
_________________________________________________________________
conv_pw_5_relu (ReLU)        (None, 28, 28, 256)       0
_________________________________________________________________
```



```
conv_pad_6 (ZeroPadding2D)   (None, 29, 29, 256)       0
_________________________________________________________________
conv_dw_6 (DepthwiseConv2D)  (None, 14, 14, 256)       2304
_________________________________________________________________
conv_dw_6_bn (BatchNormaliza (None, 14, 14, 256)       1024
_________________________________________________________________
conv_dw_6_relu (ReLU)        (None, 14, 14, 256)       0
_________________________________________________________________
conv_pw_6 (Conv2D)           (None, 14, 14, 512)       131072
_________________________________________________________________
conv_pw_6_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_6_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_dw_7 (DepthwiseConv2D)  (None, 14, 14, 512)       4608
_________________________________________________________________
conv_dw_7_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_7_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_7 (Conv2D)           (None, 14, 14, 512)       262144
_________________________________________________________________
conv_pw_7_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_7_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_dw_8 (DepthwiseConv2D)  (None, 14, 14, 512)       4608
_________________________________________________________________
conv_dw_8_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_8_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_8 (Conv2D)           (None, 14, 14, 512)       262144
_________________________________________________________________
conv_pw_8_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_8_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_dw_9 (DepthwiseConv2D)  (None, 14, 14, 512)       4608
```



```
_________________________________________________________________
conv_dw_9_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_9_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_9 (Conv2D)           (None, 14, 14, 512)       262144
_________________________________________________________________
conv_pw_9_bn (BatchNormaliza (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_9_relu (ReLU)        (None, 14, 14, 512)       0
_________________________________________________________________
conv_dw_10 (DepthwiseConv2D) (None, 14, 14, 512)       4608
_________________________________________________________________
conv_dw_10_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_10_relu (ReLU)       (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_10 (Conv2D)          (None, 14, 14, 512)       262144
_________________________________________________________________
conv_pw_10_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_10_relu (ReLU)       (None, 14, 14, 512)       0
_________________________________________________________________
conv_dw_11 (DepthwiseConv2D) (None, 14, 14, 512)       4608
_________________________________________________________________
conv_dw_11_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
conv_dw_11_relu (ReLU)       (None, 14, 14, 512)       0
_________________________________________________________________
conv_pw_11 (Conv2D)          (None, 14, 14, 512)       262144
_________________________________________________________________
conv_pw_11_bn (BatchNormaliz (None, 14, 14, 512)       2048
_________________________________________________________________
conv_pw_11_relu (ReLU)       (None, 14, 14, 512)       0
_________________________________________________________________
conv_pad_12 (ZeroPadding2D)  (None, 15, 15, 512)       0
_________________________________________________________________
conv_dw_12 (DepthwiseConv2D) (None, 7, 7, 512)         4608
_________________________________________________________________
```

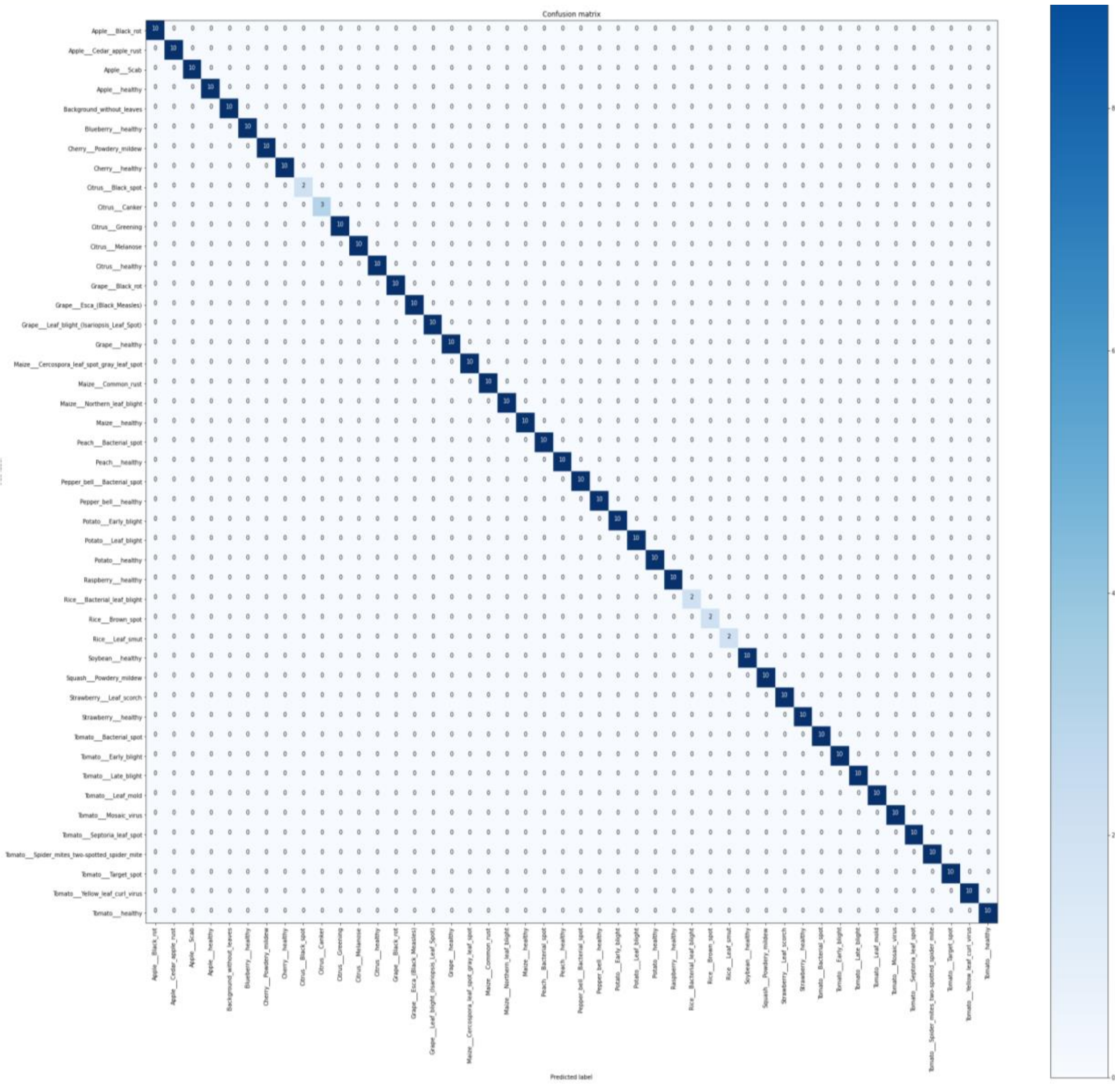


```
_________________________________________________________________
conv_dw_12_bn (BatchNormaliz (None, 7, 7, 512)         2048      
_________________________________________________________________
conv_dw_12_relu (ReLU)       (None, 7, 7, 512)         0         
_________________________________________________________________
conv_pw_12 (Conv2D)          (None, 7, 7, 1024)        524288    
_________________________________________________________________
conv_pw_12_bn (BatchNormaliz (None, 7, 7, 1024)        4096      
_________________________________________________________________
conv_pw_12_relu (ReLU)       (None, 7, 7, 1024)        0         
_________________________________________________________________
conv_dw_13 (DepthwiseConv2D) (None, 7, 7, 1024)        9216      
_________________________________________________________________
conv_dw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096      
_________________________________________________________________
conv_dw_13_relu (ReLU)       (None, 7, 7, 1024)        0         
_________________________________________________________________
conv_pw_13 (Conv2D)          (None, 7, 7, 1024)        1048576   
_________________________________________________________________
conv_pw_13_bn (BatchNormaliz (None, 7, 7, 1024)        4096      
_________________________________________________________________
conv_pw_13_relu (ReLU)       (None, 7, 7, 1024)        0         
_________________________________________________________________
global_average_pooling2d (Gl (None, 1024)              0         
_________________________________________________________________
reshape_1 (Reshape)          (None, 1, 1, 1024)        0         
_________________________________________________________________
dropout (Dropout)            (None, 1, 1, 1024)        0         
_________________________________________________________________
conv_preds (Conv2D)          (None, 1, 1, 46)          47150     
_________________________________________________________________
reshape_2 (Reshape)          (None, 46)                0         
_________________________________________________________________
predictions (Activation)     (None, 46)                0         
=================================================================
Total params: 3,276,014
Trainable params: 3,223,470
Non-trainable params: 52,544
_________________________________________________________________
```

```
# Configuring the model

mobile.compile(Adam(lr=0.001),loss='categorical_crossentropy',metrics=['accuracy'])

train_steps = train_data.n//train_data.batch_size

val_steps = val_data.n//val_data.batch_size

callback1 = keras.callbacks.TensorBoard(log_dir='drive/My Drive/Colab Notebooks/logs'
, histogram_freq=0, write_graph=True, write_images=True)

callback2 = keras.callbacks.callbacks.ModelCheckpoint('drive/My Drive/Colab Notebook
s/checkpoints/model.{epoch:02d}-{val_acc:.2f}.hdf5', monitor='val_acc', verbose=1)


# Training the model

mobile.fit_generator(train_data,steps_per_epoch=train_steps,epochs=6,validation_data=v
al_data,validation_steps=val_steps,callbacks=[callback1,callback2],verbose=1)


# Save the model

mobile.save('drive/My Drive/Colab Notebooks/model1.h5')


# Evaluating the model

loss,acc = mobile.evaluate_generator(test_data,steps=test_data.n)

print(loss,acc)

preds = mobile.predict_generator(test_data)

predictions = np.argmax(preds,axis=1)


# Create Array of class labels

arr = np.array([])

for i in range(8):
  for j in range(10):
    arr = np.append(arr,i)
for i in range(2):
  arr = np.append(arr,8)
for i in range(3):
  arr = np.append(arr,9)
for i in range(10,29):
  for j in range(10):
    arr = np.append(arr,i)
for i in range(29,32):
  for j in range(2):
    arr = np.append(arr,i)
for i in range(32,46):
  for j in range(10):
    arr = np.append(arr,i)
arr = arr.astype(int)
print(arr.shape)
lbls = test_data.class_indices
x = lbls.keys()

# Confusion Matrix display
import itertools
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix')


    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
```

```
    plt.ylabel('True label')

    plt.xlabel('Predicted label')

np.set_printoptions(precision=2)

plt.figure(figsize=(30,20))

mat = confusion_matrix(arr,predictions)

plot_confusion_matrix(mat, classes=x)

plt.show()
```

Confusion Matrix



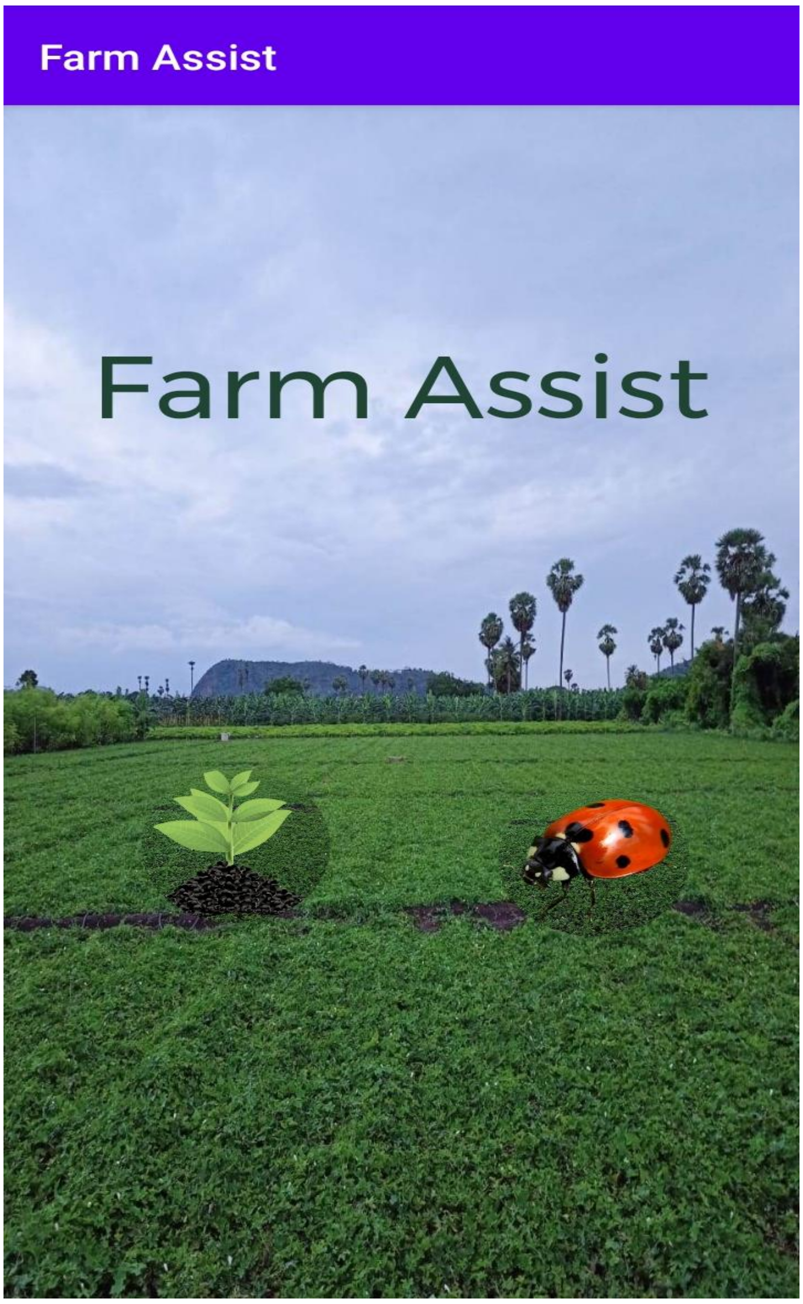
# Tflite Conversion

```
converter = tf.lite.TFLiteConverter.from_keras_model(mobile)

tflite_model = converter.convert()

tflite_model_name = "drive/My Drive/Colab Notebooks/model1.tflite"

open(tflite_model_name, "wb").write(tflite_model)


# Test for proper creation of Tflite

interpreter = tf.lite.Interpreter(model_path="drive/My Drive/Colab Notebooks/model.tflit
e")

interpreter.allocate_tensors()

input_details = interpreter.get_input_details()

output_details = interpreter.get_output_details()

input_shape = input_details[0]['shape']

input_data = np.array(np.random.random_sample(input_shape), dtype=np.float32)

interpreter.set_tensor(input_details[0]['index'], input_data)

interpreter.invoke()

output_data = interpreter.get_tensor(output_details[0]['index'])

print(output_data)
```

CHAPTER 8
OUTPUT

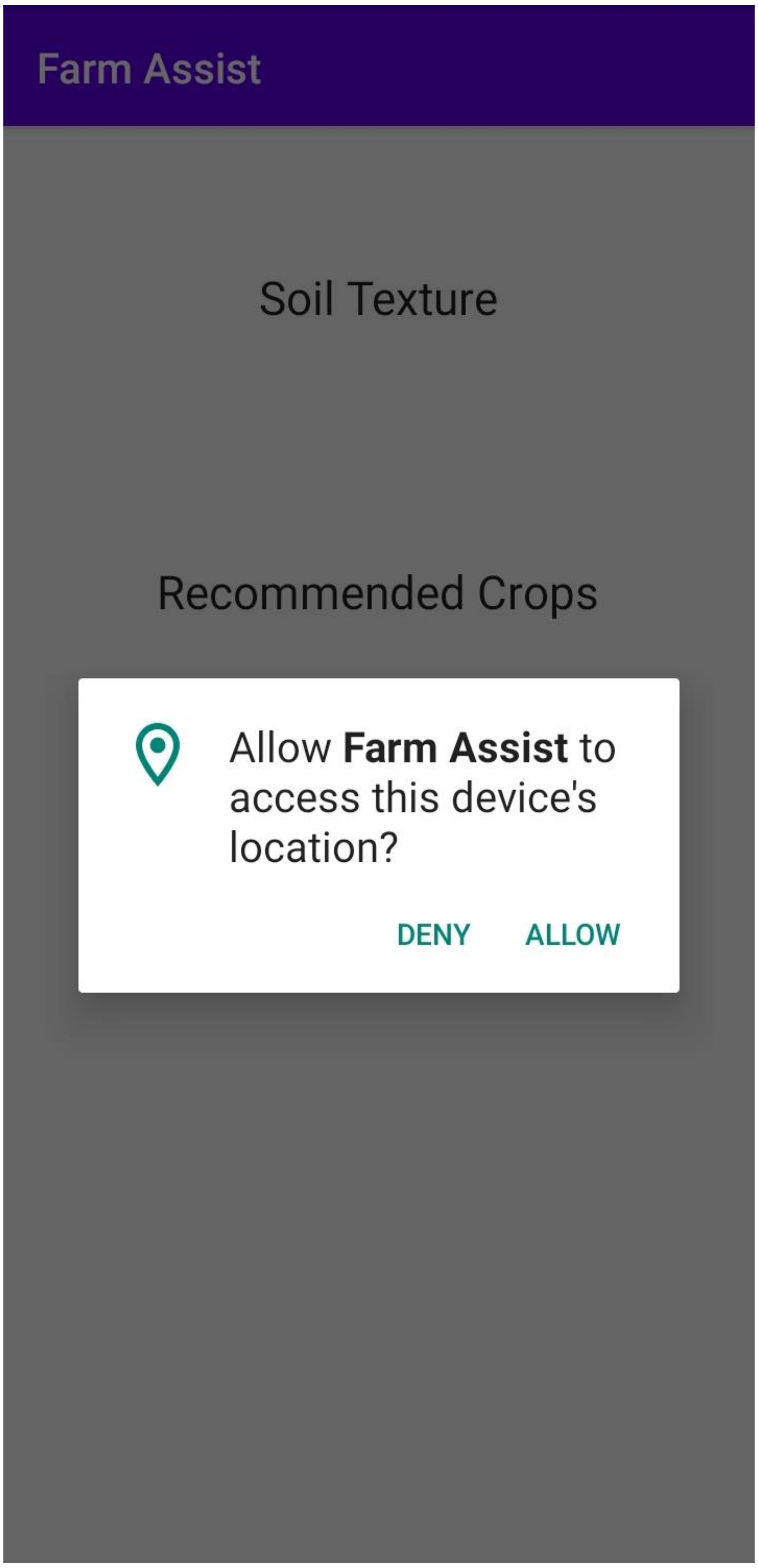
8.1 Home Screen (UI)

This is the Home Screen for a farmer immediately after opening the App with two image buttons, one for Crop Recommendation and another for Disease Detection.



8.2 Requesting Location Access

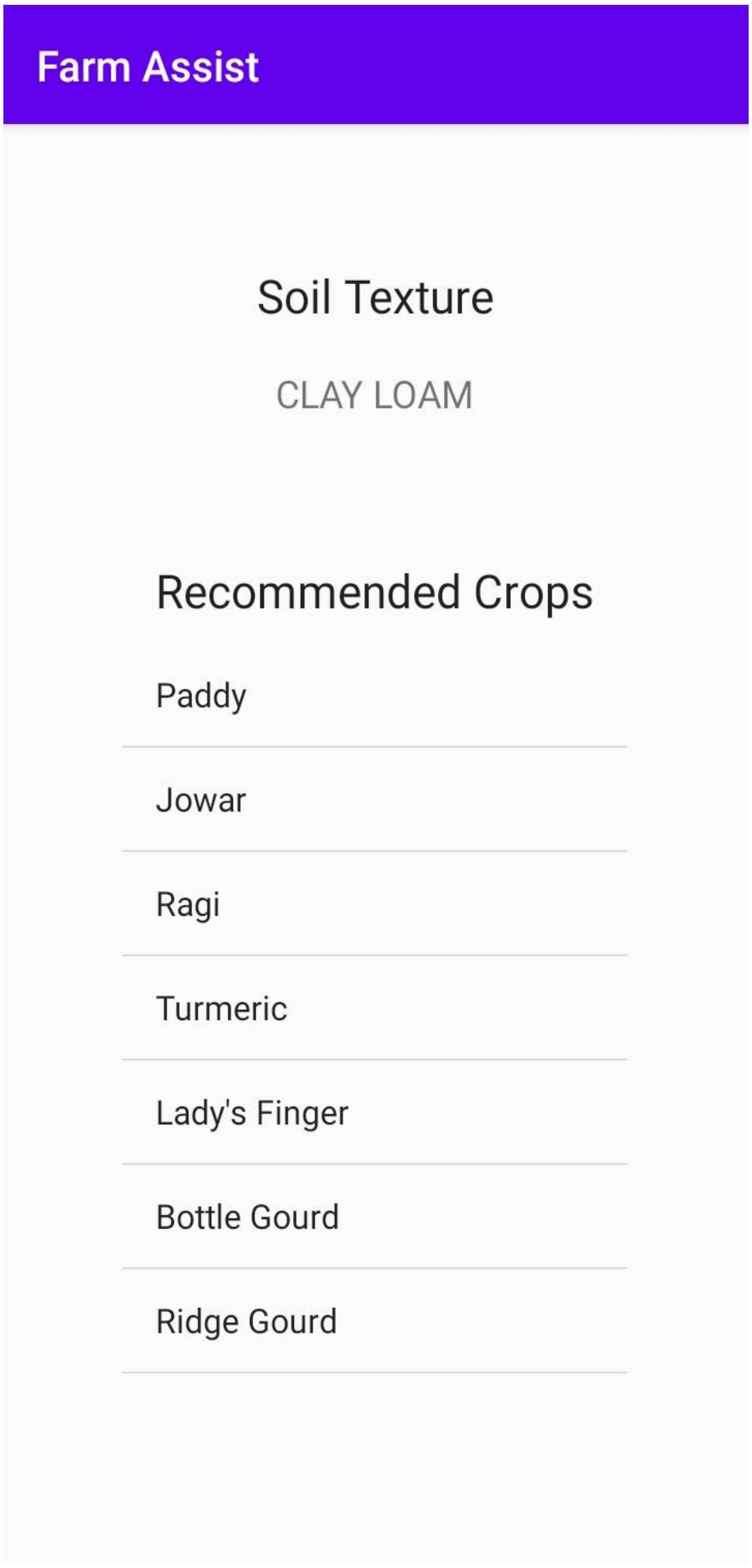
Requesting location access for Crop Recommendation.



Allow **Farm Assist** to access this device's location?

8.3 Crop Recommendations

These are the list of crops that are obtained in June for our college location.


Farm Assist

Soil Texture

CLAY LOAM

Recommended Crops

Paddy

Jowar

Ragi

Turmeric

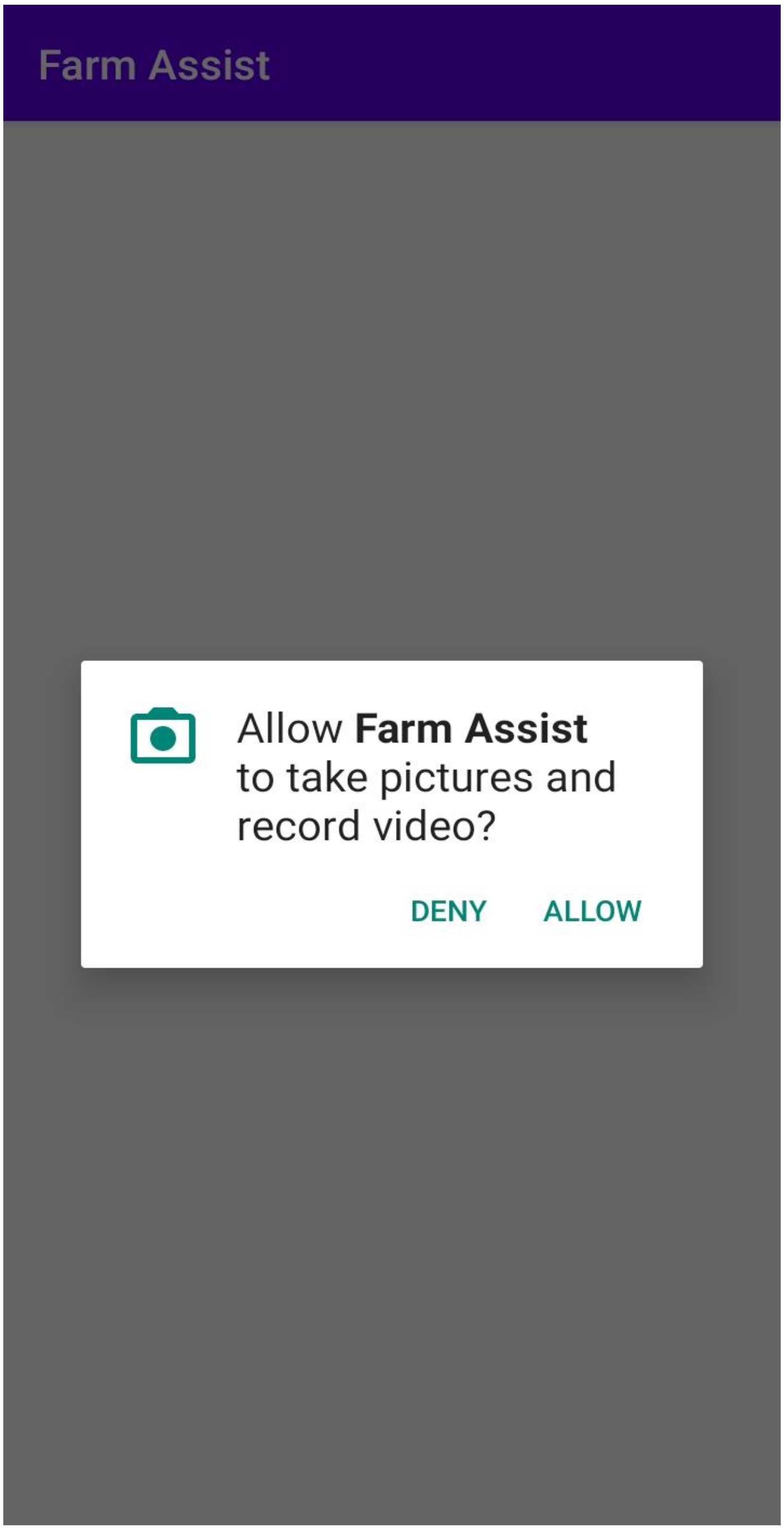
Lady's Finger

Bottle Gourd

Ridge Gourd

8.4 Requesting Camera Access

Requesting Camera access for Disease Detection

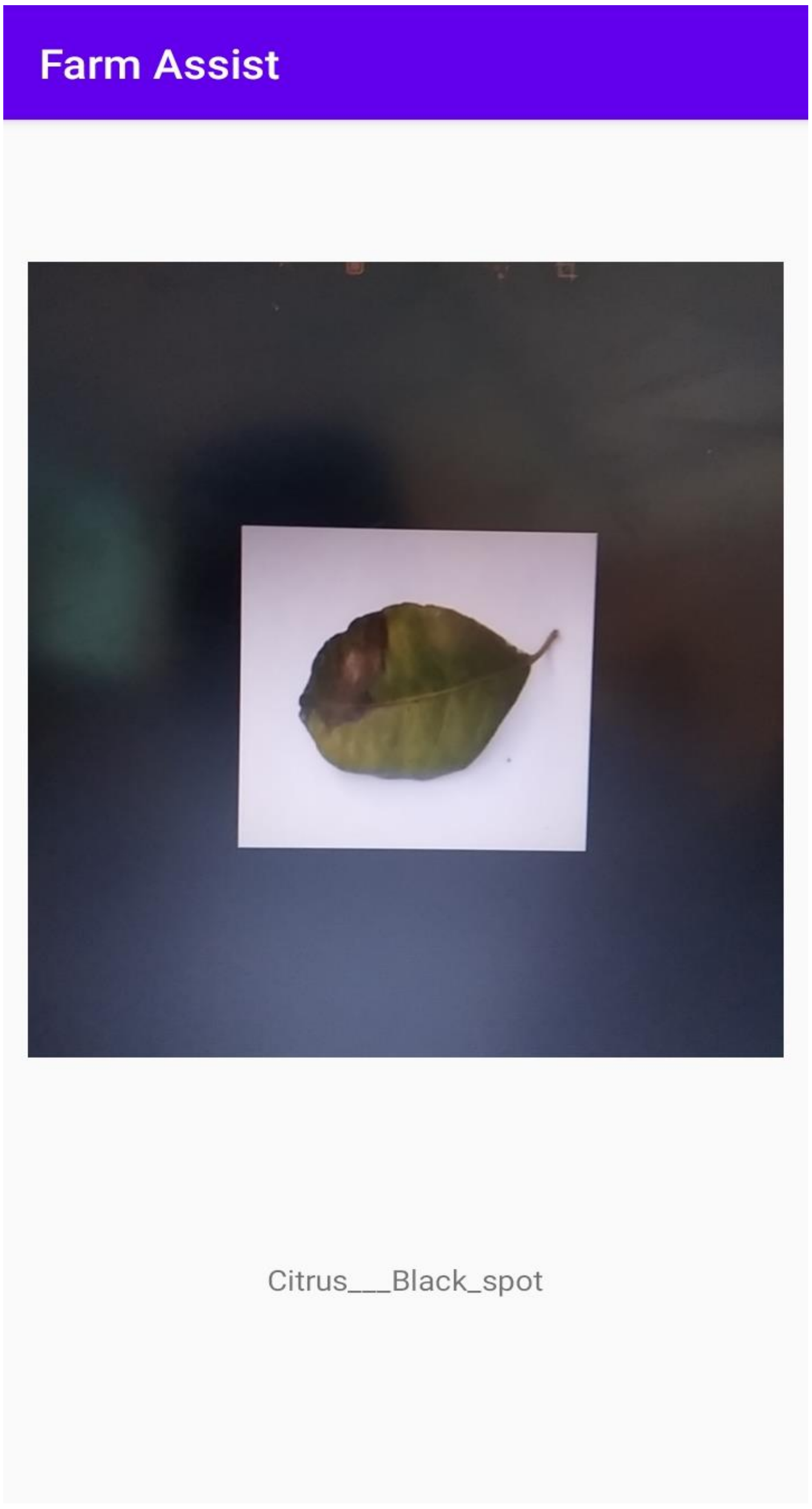


Farm Assist

Allow **Farm Assist** to take pictures and record video?

8.5 Disease Detection

Output obtained while scanning Citrus leaf with black rot disease accurately.



Citrus___Black_spot

CHAPTER 9

TESTING

Testing is a process or method of finding error/s in a software application or program so that the application functions according to the end user's requirement. The process involves verifying a system with the purpose of identifying any errors, gaps or missing requirement versus the actual requirement.

The following are the some of test cases performed with this application

- Test Case -1

  The Disease Classifier has achieved validation accuracy of 98.8% and on further testing with Test dataset, achieved an accuracy of 97.2%.

- Test Case -2

  The Crop Recommendations are compared with Agricultural University suggestions and were proven to be accurate.

- Test Case -3

  The Application is tested for its performance and it is found to be giving accurate results within no time.

- Test Case -4

  The Application is tested for usability and it is found to be clear and user friendly

- Test Case - 5

  The Application is tested for its functionality and it is found that it works accordingly on Android versions above 5.0.

CHAPTER 10
FUTURE SCOPE & APPLICATION

- Design and development of Decision Support System and implementation of Machine Learning, computer vision platforms in agriculture in this thesis signify the initial steps in this field. Efforts to develop a sustainable technology assisted systems for crop prediction and diseases detection have been reported in this thesis.
- Integrated farming systems seem to be the possible solution to the continuous increase of demand for food and nutrition, income stability and livelihood upliftment particularly for farmers with less resources.
- As farming relies more on complex decisions and equipment's, further research and enhancement in this fields of technology will play an increasingly larger role in farm management.
- This system can be deployed after enhancing its disease detection capabilities by collecting images of various disease types so that it would be much more scalable.
- This system can further be improvised to other aspects of farming like fertiliser/pesticides recommendations which may also give rise to new, easy and reliable farming methodologies.
- With unpredicted growing trends, this system might be handy for self-decisions rather than depending on official announcements or half knowledge which might cause greater risks
- To tackle India-specific agricultural problems, induction of Artificial Intelligence in this particular field can create unimaginable results such as seed improvisation, crop protection etc.

- Moving a step further, we can also use satellite data with ML techniques to assess imageries of farms and predict monetary prospects of their future crop yield. This may lead to scraping off age old tradition of dependence and faith and can ultimately create a platform where farmer can be the sole operators and decision makers of their own farm.

CHAPTER 11
CONCLUSION

Agriculture is a diverse and vital industry. In India it can rather be called as livelihood for more than 2/3rd population. It is undergoing a process of transition to a market economy, with substantial changes in the productivity and supply with all other sectors of the economy. So, to setup the standards of agriculture to the required point the whole burden and expectation are solely on the farming community.

Technology plays an important role to modernise the existing farming system, in other terms it has been a boon to farming where there are many platforms specific to agriculture and farmers. This project is a work that mainly address some of the main constraints like crop recommendation and leaf disease detection etc. With crop being very specific with locational connections and prone to certain diseases it becomes difficult to address these issues particularly in large scales. Thus the application plays an important role in minimising the decision making effort to fraction of seconds with the help of technologies mentioned above. The system has been greatly simplified to a point for ease of understanding and enhancing the utility among the farmers. In conclusion, this Decision Support System is easily adaptable and cost free system which gives reliable results in any sort of conditions so that the farmer can blind trust this application.

Finally, Agriculture tends to be taken for granted. Many people need to understand that agricultural land is a non-renewable resource requiring appropriate management techniques before allowing land for urban expansion. If this situation continues neither the technologies nor any government schemes can meet the demand for long term sustainability and supply.

BIBLIOGRAPHY

1. Omkar Kulkarni **“Crop Disease Detection Using Deep Learning”** , 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA).

2. Muhammad Hammad Saleem, Johan Potgieter, Khalid Mahmood Arif **“Plant Disease Detection and Classification by Deep Learning”**, MDPI.

3. https://www.farmer.gov.in

4. https://www.data.gov

5. https://agropedia.iitk.ac.in

6. https://wikifarmer.com

7. https://www.ers.uda.gov

8. https://fao.org