A PROJECT REPORT ON

DETECTION AND CLASSIFICATION OF DISEASES IN TOMATO PLANTS

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

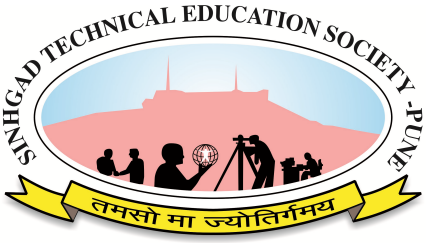
Of

SAVITRIBAI PHULE PUNE UNIVERSITY

By

RUSHI CHAUDHARI	**B150234227**
ROHAN MARATHE	**B150234302**
VEDANG PINGLE	**B150234344**
ROHAN RANE	**B150234351**

Under the guidance of
Prof. G.T. CHAVAN



Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING

SINHGAD COLLEGE OF ENGINEERING, PUNE-41

Accredited by NAAC
2018-19

Sinhgad Technical Education Society,
Department of Computer Engineering
Sinhgad College of Engineering , Pune-41

Sinhgad Institutes

Date:

CERTIFICATE

This is to certify that the project report entitled

"DETECTION AND CLASSIFICATION OF DISEASES IN TOMATO PLANTS"

Submitted by

RUSHI CHAUDHARI	**B150234227**
ROHAN MARATHE	**B150234302**
VEDANG PINGLE	**B150234344**
ROHAN RANE	**B150234351**

is a bonafide work carried out by him/her under the supervision of Prof. G.T. Chavan and it is approved for the partial fulfillment of the requirements of Savitribai Phule Pune University, Pune for the award of the degree of Bachelor of Engineering (Computer Engineering) during the year 2018-19.

Prof. G.T. Chavan
Internal Guide

Prof. M.P. Wankhade
Head
Department of Computer Engineering

Dr. S.D. Lokhande
Principal
Sinhgad college of Engineering

Acknowledgement

It is great pleasure for us to acknowledge the assistance and contribution of number of individuals who helped us in developing this project. First and foremost we wish to express our gratitude and thanks to Prof. G. T. Chavan for his consistent and enthusiastic guidance and help in successful completion of this project. We express our sincere gratitude to our Principal, Dr. S. D. Lokhande, our Head of Department Prof. M. P. Wankhede and our Project Coordinator Prof. U. A. Mande for their valuable guidance.

Rushi Chaudhari
Rohan Marathe
Vedang Pingle
Rohan Rane

Abstract

India is widely considered to be an agrarian nation. It is the second largest producer of wheat and rice which are the world's major food staples. The agricultural system forms the backbone of the Indian economy and the country stands second worldwide in agricultural output and yet, this sector remains largely disorganized, underdeveloped and characterized by a lack of penetration of technology. Crop diseases, in particular is a growing concern faced by farmers these days as the weather and climate is becoming erratic and more unpredictable than ever. There is a lack of proper infrastructure for detection and identification of crop diseases. Farmers face significant losses as a result of destruction of crops due to various such diseases.

Our area of focus was classification of diseases in tomato plants. We intend to design and build a system that identifies tomato plant diseases based on the input image of the plant leaf. Using the novel technology of deep learning, higher accuracy can be achieved for a wider range of diseases and larger datasets. There have been very few attempts to implement such an application in real time although high accuracy has been achieved during training and testing. An interactive image segmentation technique can be employed using Graph Cut algorithm. Processing and storage constraints are eliminated with the help of Cloud platform. We also aim to make the application more comprehensive by providing the farmer with remedies and preventive measures stored on the Cloud, for the detected disease.

List of Figures

List of Tables

Abbreviations

CNN	Convolutional Neural Network

Contents

Chapter 1

INTRODUCTION

1.1 Background and Basics

Agriculture is cultivation of crops for food, fiber, medicinal plants and other products used to sustain human life. India is widely considered to be an agrarian country and ranks second in the world in terms of agricultural output. Despite agriculture being practiced almost everywhere throughout the country and playing a crucial role in the nation's economy, its potential remains relatively under fulfilled due to lack of penetration of technology in this sector. One of the prominent issue in agriculture is crop damage caused by various viral, bacterial, and fungal diseases. Viral diseases such as leaf curl spread rapidly and can cause massive destruction of crops in a matter of days or even hours.

Tomato is one of the most popular and widely produced commercial crop. It is grown in all the states of India. Tomato production is sensitive to temperature; the optimal temperature required for the luxuriant growth of tomato is 23-270C. Temperature below 150C and above 350C during the day time and above 210C during the night is detrimental to the fruit setting. Tomato crop is prone to various types of diseases such as seed borne, soil borne as well as air borne. Hence, it suffers through large number of severe diseases caused by fungi, bacteria, viruses, mycoplasma, nematode etc. Many a times, in order to combat these diseases, farmers use pesticides, fungicides and fertilizers excessively which in some cases may sharply increase yield from cultivation, but at the same time has caused widespread ecological damage and negative human health effects. Also if the farmer misrecognizes the disease or fails to detect it soon enough, it may spread throughout the field and the damages would become irreparable.

Researchers have studied and performed disease detection based on image analysis in the past using various image processing and or machine learning techniques. Learning models such as SVMs and K- Means Clustering have shown decent results when it comes to smaller datasets with lesser parameters, but the technique which has been the most promising is Deep Learning. Deep learning is a subset of machine learning. Deep artificial neural networks are a set of algorithms that have set new records in accuracy for many important problems, such as image recognition, sound recognition, recommender systems, etc. Deep learning neural networks were inspired by our understanding of the biology of our brains – all those interconnections between the neurons. But, unlike a biological brain where any

neuron can connect to any other neuron within a certain physical distance, these artificial neural networks have discrete layers, connections, and directions of data propagation. Each neuron assigns a weighting to its input — how correct or incorrect it is relative to the task being performed. The final output is then determined by the total of those weightings. An artificial neural network deep learning model such as CNN is best suited to this task as it gives high accuracy when the dataset is sufficiently large and also outputs multiple probabilistic outcomes and hence can classify multiple diseases. Two different deep learning network architectures AlexNet and SqueezeNet have been trained and tested previously on image datasets for disease classification. However, such CNNs are heavy and have high processing demands which are difficult to satisfy using a regular computer. It is therefore preferable to train and store the model on a Cloud platform.

1.2 Literature Survey

Title: Halil Durmus, Ece Olcay Günes, Mürvet Kirci, [1] "Disease Detection on the Leaves of the Tomato Plants by Using Deep Learning", September 2017

Description:

- Deep learning network architectures AlexNet and SqueezeNet is used for training and validation on Plant Village dataset. The Plant Village dataset contains 54,309 images for 14 different crops. The deep learning algorithm should run in real time on a robot. Training and validation is done on the Nvidia Jetson TX1.
- Convolutional neural networks are trained on the deep learning frameworks called Caffe. Usually, training is done on the workstations with GPU or GPU clusters on the board.

Limitations:

- Training batch sizes are smaller.
- Training time is longer due to computing resource limitations of on-board computer.
- Lesser accuracy due to smaller batch size.
- Lack of feature extraction over complex backgrounds.

Title: Mrunmayee Dhakate, Ingole A. B., [2] "Diagnosis of Pomegranate Plant Diseases using Neural Network", June 2016

Description:

- The system uses some images for training and some for testing. K-Means clustering is used to perform segmentation and extract features.
- Texture features are extracted using Grey Level Co-occurrence Matrix (GLCM). They are calculated from the distribution of intensity combinations at certain positions relative to others.
- The system applies neural networks through the use of Multilayer Perceptrons (MLP) along with the back propagation algorithm for error correction and learning.

Limitations:

- Relatively less accuracy of 83-87% is achieved for classification of diseases.
- The system is limited to 4 diseases only.

Title: Anand K. Hase, Priyanka S. Aher, Sudeep K. Hase, [3] "Detection, Categorization and suggestion to cure infected plants of Tomato and Grapes by using OpenCV framework for Andriod Environment", December 2017

Description:

- The system involves an android application which serves as an interface for image acquisition and image analysis. Image processing techniques are used to detect and classify the diseases in plants.
- Grabcut algorithm is used to extract the foreground i.e. the leaf through iterated graph cuts. To segment an image, thresholding technique is used. After feature extraction, feature matching is done using FLANN.

Limitations:

- The system is over reliant on image processing and there is very limited learning involved in the model. This affects accuracy of detection and classification.
- Absence of scalable and reliable storage system such as cloud.

Title: Jia Shijie, Jia Peiyi, Hu Siping, Liu Haibo, [4] "Automatic Detection of Tomato Diseases and Pests Based on Leaf Images", January 2018

Description:

- A convolutional neural network model is constructed based on VGG16 and transfer learning. The VGG16 model and SVM are used together where VGG16 acts as image feature extractor and SVM as classifier to detect pests and diseases.
- Fine tuning is employed to construct an end to end classification model based on the original VGG16 model. The fine tuning model performs slightly better than the VGG16+SVM model.
- Data enhancement or augmentation techniques such as rotating, flipping, inverting, scaling are used to increase the dataset size.

Limitations:

- Relatively less accuracy of 89%.
- Accuracy relies on high quality test images.
- Time consuming since training is done on desktop computer.

Title: Monzurul Islam, Anh Dinh, Khan Wahid, Pankaj Bhowmik, [5] "Detection of Potato Diseases Using Image Segmentation and Multiclass Support Vector Machine", June 2017

Description:

- An integrated approach comprising of image processing and machine learning is presented. Image processing includes segmentation to obtain region of interest. Feature extraction is employed next using Grey Level Co-occurrence Matrix (GLCM).
- Multiclass Support Vector Machine is used as a supervised learning model for classification of disease and is applied on 300 leaf images.

Limitations:

- System limited to detection of 2 diseases only.
- Performance does not improve as dataset grows for SVM.

1.3 Project Undertaken

To design and develop a tool for farmers to improve the productivity of tomato crops by performing timely detection of several diseases while recording and pinpointing the location

of the affected crop. To suggest counter measures and preventive measures thereby reducing the excessive expenditure of the farmer on fertilizers and fungicides.

1.3.1 Problem Definition

There have been very few attempts to implement a real time disease detection system through image analysis of leaf images. Majority of the research and implementations have been purely on training and testing datasets. Our aim is to implement leaf image based disease detection in real time and suggest the farmer with solutions regarding the detected disease. The solution will consist of counter measures such as various fertilizers and fungicides. We intend to provide an android application as an interface and make use of cloud platform to remove processing and storage constraints. The system will also be designed to store the location of the diseased crop which can be accessed at any time by the farmer.

1.3.2 Scope Statement

Our project's scope reaches beyond the current systems by overcoming the processing and storage limitations through the use of cloud platform, performing real time segmentation and prediction of disease, recording the location of affected crop, and suggesting preventive and remedial measures for the disease.

1.4 Organization of Report

Chapter 2 elaborates about different system requirements in detail. It also contains system features, cost and effort analysis of project and the timeline of project. Chapter 3 focuses on design and analysis in terms of UML diagrams, feasibility analysis and mathematical model. Chapter 4 contains details about testing carried out on the system.

Chapter 2

PROJECT PLANNING AND MANAGEMENT

2.1 Detail System Requirement Specification (SRS)

2.1.1 System Overview

- The designed system is an android-based stand-alone system and works independently of any other hardware or software. The android module acts as an interface with the user.
- The system involves a cloud platform for processing the image and database storage. The android module is connected to this cloud platform securely and data transfer takes place between the two.
- There is an additional localization module which uses Wi-Fi signal strength of neighboring routers to determine the location of the user.

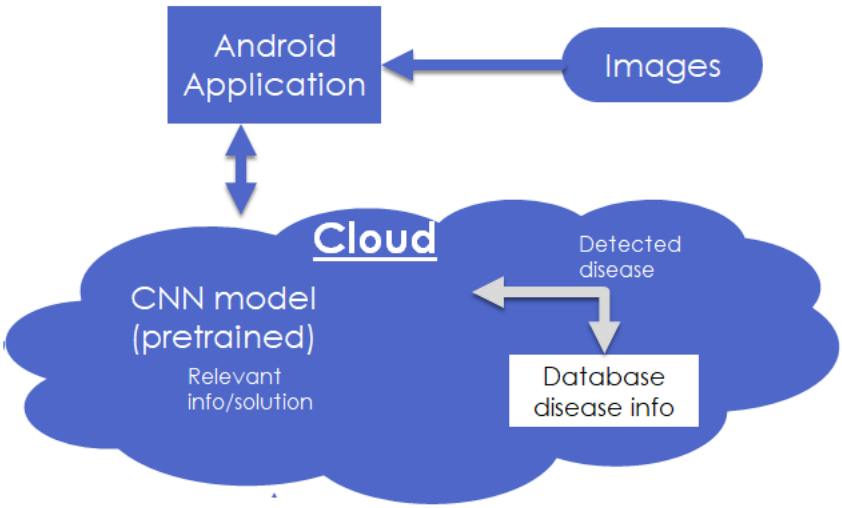


Figure 2.1: System Overview

Figure 2.1 gives an overview of our system. The figure shows communication between its modules.

- Following are the main functionalities provided by the proposed system:
  - Identification of disease based on analysis of input image.

  - Suggestion of solutions including but not limited to a range of fertilizers and fungicides.
  - Performing localization of the diseased crop with a reasonable accuracy.
  - Intuitive and interactive image segmentation via an android application.

- The system requires the user to capture the crop image against a monochromatic plain background which is preprocessed and then subsequent functions are performed.

2.1.2 Functional Requirements

These are the following functional requirements of the system:

1. Leaf Image capture and preprocessing.

   - **Description and Priority**

     This feature acquires the leaf image and performs preprocessing activities such as segmentation, resizing, etc. The feature is vital from the perspective of CNN model, hence of High priority

   - **Stimulus/Response Sequences**

     The camera button captures the image and triggers the preprocessing process. The resultant image is sent to the cloud platform.

   - **Functional Requirements**

     REQ-1: The image should segmented accurately so as to only preserve the leaf

     REQ-2: The image quality should be maintained in the output image. region.

2. **Leaf Image Analysis and Disease Prediction.**

   - **Description and Priority**

     This forms the core feature of the system. It accepts preprocessed image as input and predicts the output class of leaf condition.

   - **Stimulus/Response Sequences**

     The feature is triggered by complete upload of image to cloud.

   - **Functional Requirements**

     REQ-1: The model should accurately make a probabilistic prediction of the disease based on the preprocessed image.

     REQ-2: If the probability of output class is less than the threshold value, it should be discarded and the user should be notified of it.

3. **Solution Recommendation**

- **Description and Priority**

  It provides the user with a cost effective solution in terms of preventive measures as well as cures such as fertilizers and fungicides. User is supplied with a range of reliable options in terms of brands.

- **Stimulus/Response Sequences**

  The database is queried on the basis of the output of CNN model for relevant information which is displayed to the user via the android application.

- **Functional Requirements**

  REQ-1: For the detected disease, the solution should consist of fungicides, fertilizers and other counter measures as well as prevention measures for the future.

4. **Localization**

- **Description and Priority**

  The user has the option to save the location of the affected plant with the help of ESP modules. This will help the user to trace back the location of the affected region.

- **Stimulus/Response Sequences**

  A UI button is used to activate the feature and the response will be displayed via a visual representation of the field.

- **Functional Requirements**

  REQ-1: The location of the user should be determined relative to the ESP nodes.

2.1.3 Non-Functional Requirements

Req1

2.1.4 Deployment Environment

1. **Correctness**

   The system should suggest appropriate disease for the affected crop and the system should suggest the correct solution for the disease.

2. **Reliability**

   System should have minimum system halts or crashes.

3. **Reusability**

   The system should be designed in a modular way which will enable us to reuse the code for future development.

4. **Maintainability**

   The fertilizer database for specific disease should be well maintained updated regularly. Also the training model should be retrained after 1000 user images have been serviced.

5. **Usability**

   Unnecessary details should be kept hidden from the view of the end user. The application UI should be simple and easy to use.

6. **Testability**

   The system should be designed in a modular way so that unit testing becomes easier.

2.1.5 External Interface Requirements

User Interfaces

The home screen will contain of a navigation drawer which consists of multiple options such as Help, Contact Feedback, Settings, etc. It will also contain a camera pane through which image is captured. Succeeding pages will contain image pre-processing interface and results window.

Hardware Interfaces

- ESP modules deployed on the field
- Android Smartphone receiving signal strength of each ESP node

Software Interfaces

- Google Colaboratory cloud platform for development and testing purpose
- DigitalOcean cloud platform for deployment of the model and database management.
- OpenCV's Java API
- Android 6.0 and above
- Python libraries like pandas, scikit-learn, tensorflow, keras with python version Python3.5
- Data input in the form of image is sent to the server. Result is displayed in the form of textual data.

Communications Interfaces

- Accessing the DigitalOcean droplet via SSH

2.1.6 Other Requirements

- The image to be uploaded for analysis should be captured against a monochromatic, plain background in order to aid image preprocessing.
- The user must have a reliable internet connection.
- The user must have a basic understanding of smartphone functionalities and the English language.

2.2 Cost & Efforts Estimates

COCOMO model is a single-valued, static model that computes software development effort (and cost) as a function of program size expressed in estimated lines of code (LOC).

- Development Mode – Semidetached

  A development project can be considered of semidetached type, if the development consists of staff with mixed experience levels who must meet a mix of rigid and less than rigid requirements i.e. it is a combination of organic and embedded modes.

  The co-efficient values in semidetached mode are as follows:

  1. a = 3.0
  2. b = 1.12
  3. c = 2.5
  4. d = 0.35

- Calculations

  1. Development Effort

     Source Lines of Code (SLOC) 8000
     Kilo Delivered Source Instruction (KDSI) = SLOC/1000 = 8000/1000 = 8
     E = a x (KDSI) b
     E = 3.0 x (8)1.12
     E = 30.80

  2. Efforts and Development Time (TDEV)

     TDEV = C x (MM) d
     TDEV = 2.5 x (30.80)0.35
     TDEV = 8.29

  3. Staffing Size (N)

     N = Effort/Duration
     N = E / TDEV
     N = 30.80 / 8.29 = 3.71 4

2.3 Project Scheduling

2.3.1 Time Line Chart

Task Name	Start Date	End Date	Duration
Initiate the Project	**21.06.18**	**22.08.18**	**62d**
Selection of Project Domain	21.06.18	24.06.18	3d
Finalization of Project Domain	24.06.18	30.06.18	6d
Selection of Project topic	30.06.18	20.07.18	20d
Finalization of Topic and Title Submission	20.07.18	30.07.18	10d
Literature Survey	30.07.18	15.08.18	16d
Requirement Analysis	15.08.18	22.08.18	7d
Project Planning and Designing	**05.09.18**	**30.09.18**	**25d**
Polyhouse Visit	05.09.18	05.09.18	1d
Functional requirements and System Architecture	05.09.18	12.09.18	7d
Design of UML Diagrams	12.09.18	17.09.18	5d
Visit to Agro-pharmacy	18.09.18	18.09.18	1d
Finalization of Technologies	18.09.18	30.09.18	12d
Implementation and Coding	**25.12.19**	**10.04.19**	**109d**
Dataset acquisition	25.12.19	05.01.19	12d
Training	09.01.19	11.01.19	3d
Image preprocessing	16.01.19	22.01.19	7d
Dataset acquisition	31.01.19	10.02.19	11d
Solution acquisition	16.02.19	25.02.19	10d
App development	11.03.19	20.03.19	16d
Cloud integration	21.03.19	24.03.19	5d
Optimizations	31.03.19	10.04.19	11d

Figure 2.2: Timeline Chart

Figure 2.2 is a chart which shows the timeline of our project development from 21st June 2018 to 10th April 2019.

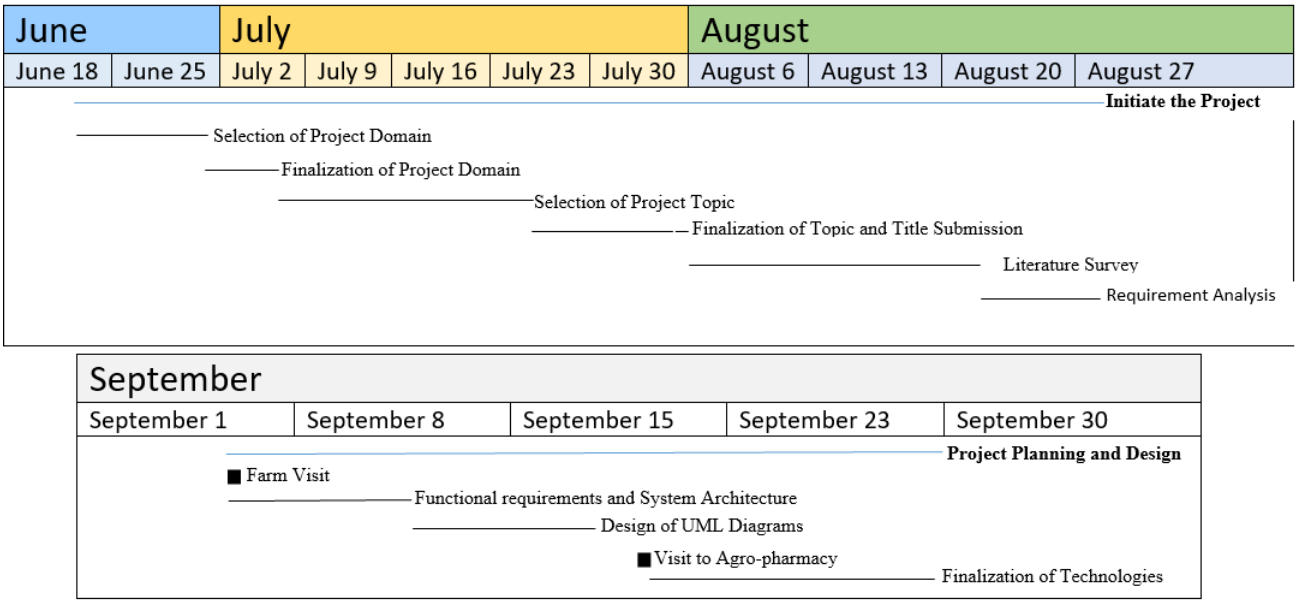


Figure 2.3: Timeline Chart JULY 2018 - SEPTEMBER 2018

Figure 2.3 shows the timeline of project development in first phase of the project.

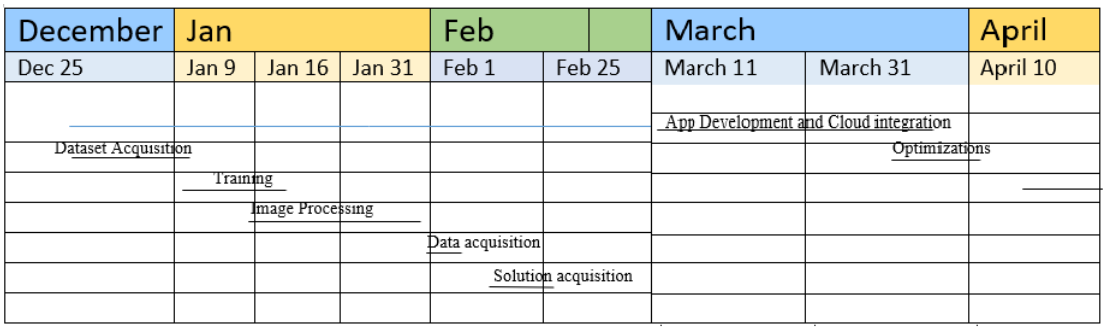


Figure 2.4: Timeline Chart DECEMBER 2018 - APRIL 2019

Figure 2.3 shows the timeline of project development in second phase of the project.

Chapter 3

ANALYSIS & DESIGN

3.1 Mathematical Model

1. Let 'S' be the system

   S = {. . . . . }

2. Identify the inputs as U,Lp

   S = {U,Lp. . . . . } where U are the user inputs, Lp are the identified time and coordinates U = {I} I = {i—'i' is the image of a leaf of the diseased crop} Lp = {X,Y,T} X = {x—'x' is the x coordinate of the location} Y = {y—'y' is the y coordinate of the location} T = {t—'t' is the timestamp}

3. Identift the outputs as O

   S = {U,Lp,O. . . . . } O = {O1,O2} O1 = {disease predicted} O2 = {Rc—Rc is the method/remedy suggested to overcome the disease}

4. Identify the processes as P

   S = {U,Lp,O,P. . . . . } P = {Ic,Dc,Sc}

5. Ic is the Image pre-processing and filtering for relevant feature extraction and compression

   Ic = {Ps,Pc}

   - Ps = {f—'f' is function segmenting the image and extracting foreground}
   - Pc = {f—'f' is function compressing the image}

6. Dc is processing and analysis of data on cloud

   Dc = {Iip,Pi}

   - Iip = {t—'t' is valid input from the smartphone device}
   - Pi = {p—'p' is function that analyses the input and provides a decision based on the said input} Pi(Iip) = IOp
   - IOp = {m—'m' is the probabilistic output of the classifier }

7. Sc is the remedy suggestion

   Sc = {Rip,Rp,Rop}

- Rip = {r—'r' is the input from Dc}
- Rp = {r—'r' is function for association of input reference dataset}
- Rp(Rip) = Rop
- Rop = {r—'r' is the remedy suggested}

8. Identify failure cases as F'

   S = {U,Lp,O,P,F'.....}

   Failure occurs when

   - F' = { }
   - F' = {p—'p' I,p is inconsistent}{Fs—Fs Fc for C}

9. Identify success case (terminating case) as e

   S = {U,Lp,O,P,F',e..... Success is defined as- e = {a—a O1,a is correct}{Fs—Fs Fc for C}

10. Initial conditions as S0

    S = S = {U,Lp,O,P,F',e,S0}

    S0 = {U = { },Lp = { , , },O={ },P={Running}}

3.2 Feasibility Analysis

In order to analyze the project's feasibility, we consider Rs (Remedy Suggestion) module algorithm. In Rs, we suggest the best remedies based on the detected diseases. Now we derive the non-deterministic algorithm for the same.

Algorithm Rs(I) //I is the image of diseased crop S = Fr,Fu,C Solution set Fr = Fertilizer Fu = Fungicides C = Conditions to be maintained. F = Identify(I) //identify class of diseased leaf from image If(F is found) S = Lookup(F) // find the solution set for the given conditions F from the database Success(); Else Failure();

This given algorithm can be proved NP-Hard in the following way A problem L can be considered to be NP-Hard if and only if Satisfiability reduces to L (Satisfiability L) The proportional formula G for the algorithm Rs is: G = I $\hat{P}$ $\hat{R}$ Where, I: is true when the input image from the android device is successfully captured, pre-processed, and uploaded. P: is true when the class of diseased leaf is detected from the uploaded image. R: is true when the relevant remedies are fetched by the system. The algorithm for satisfiability of this propositional formula is as follows.

Algorithm Evaluate() // Satisfiability algorithm is used to determine whether the propositional formula G is satisfiable or not.

{ For i = 1 to n Xi = Remedies(True,False) If G(I,P,R) then success(); Else failure(); }

This problem of Satisfiability can be solved using algorithm Rs as the cases where the Rs results in true and false are the same as that of the Algorithm Evaluate. Or in other words the Statisfiability reduces to problem Rs itself (Satisfiability L). So the above problem of Rs can be regarded as NP-Hard. The verification of Rs algorithm can be done in Polynomial time. This shows that the Rs algorithm is NP. Hence we have proved that the problem of Rs Remedy suggestion as both NP as well as NP-Hard. This means that the problem belongs to an intersection of these two sets called as NP-Complete. Therefore, this analysis shows that the problem is NP-Complete. Hence it can be implemented in polynomial time on a non-deterministic machine.

3.3 Architecture Diagram

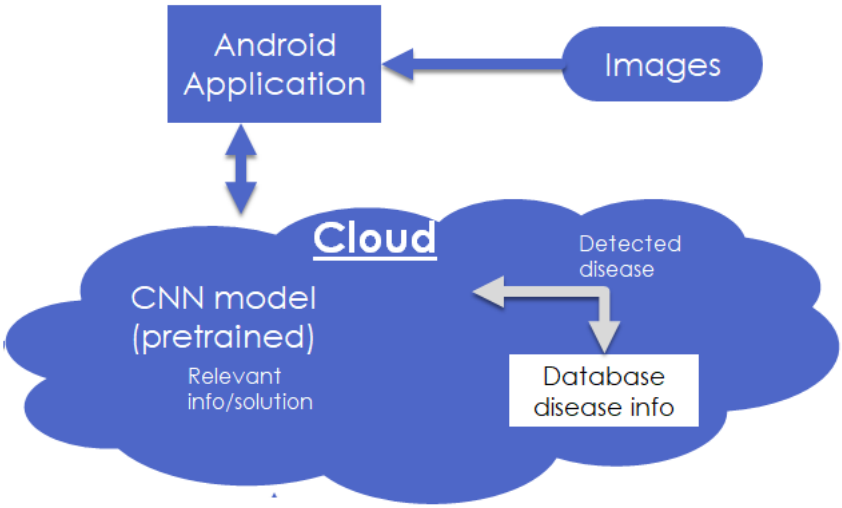


Figure 3.1: Architecture Diagram

Figure 3.1 shows the representation of architecture of our system.

3.4 UML Diagrams

3.4.1 Use Case Diagram

Use case diagrams are usually referred to as behaviour diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system.

3.4.2 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams are intended to model both computational and organizational processes as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

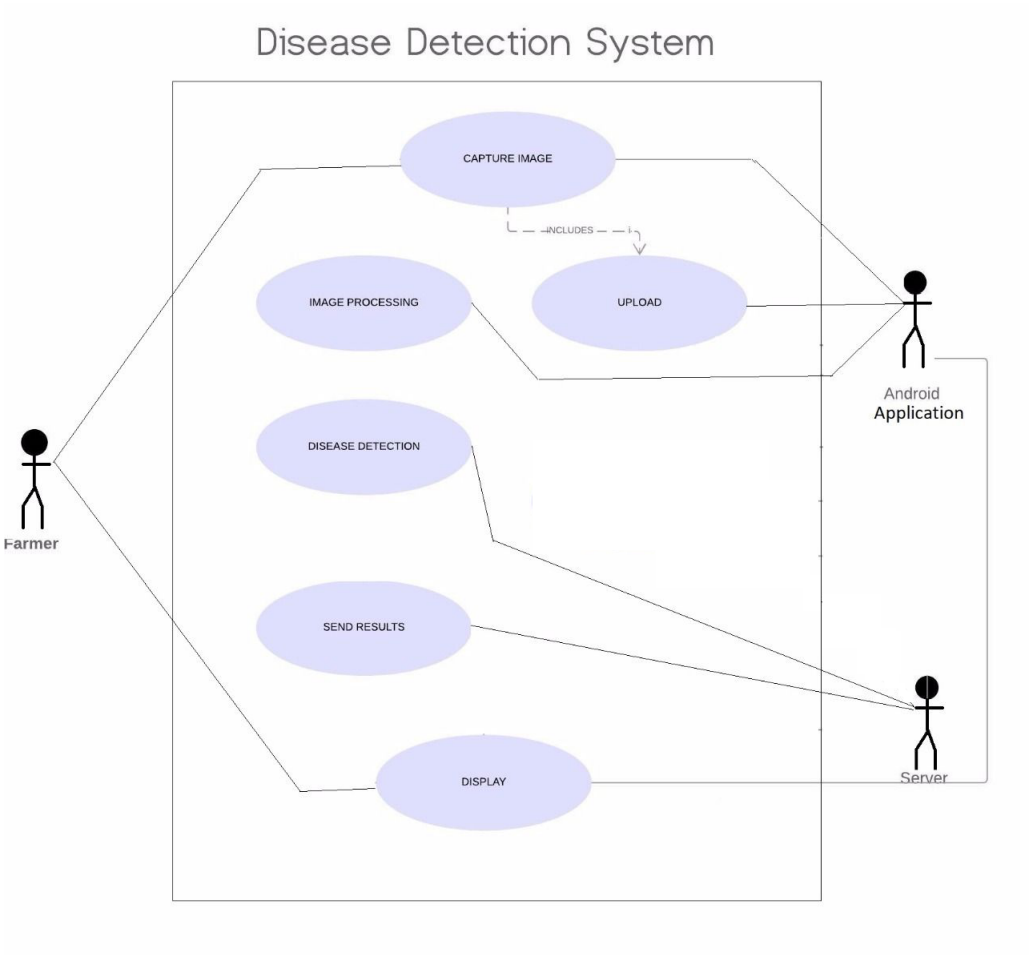


Figure 3.2: Use Case Diagram

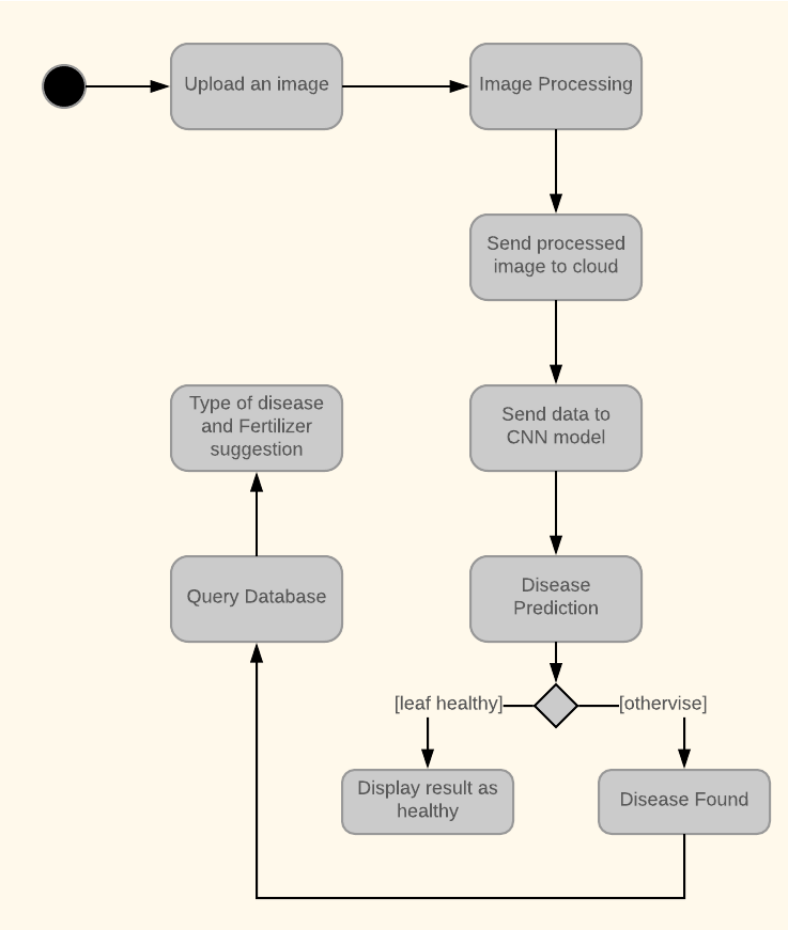


Figure 3.3: Activity Diagram

3.4.3 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

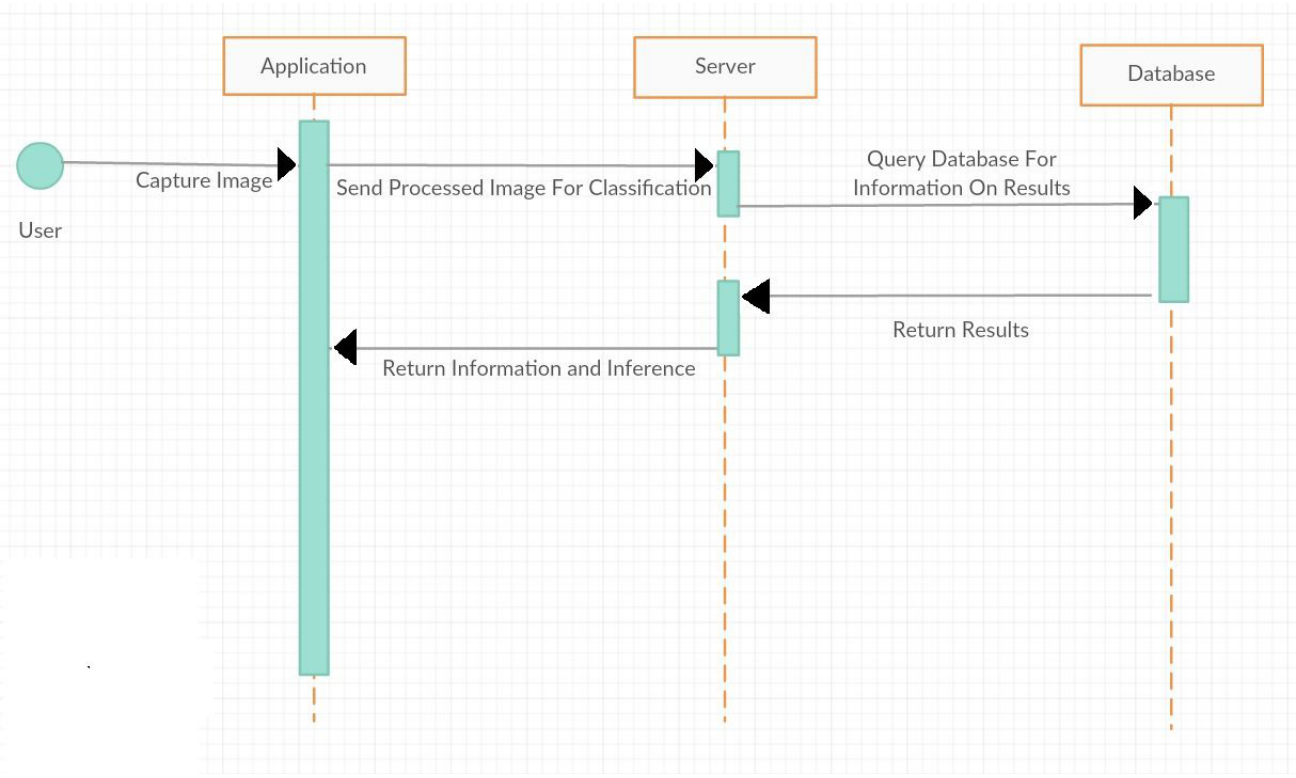


Figure 3.4: Sequence Diagram

3.4.4 Class Diagram

In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects. The class diagram is the main building block of object-oriented modelling. It is used for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code.

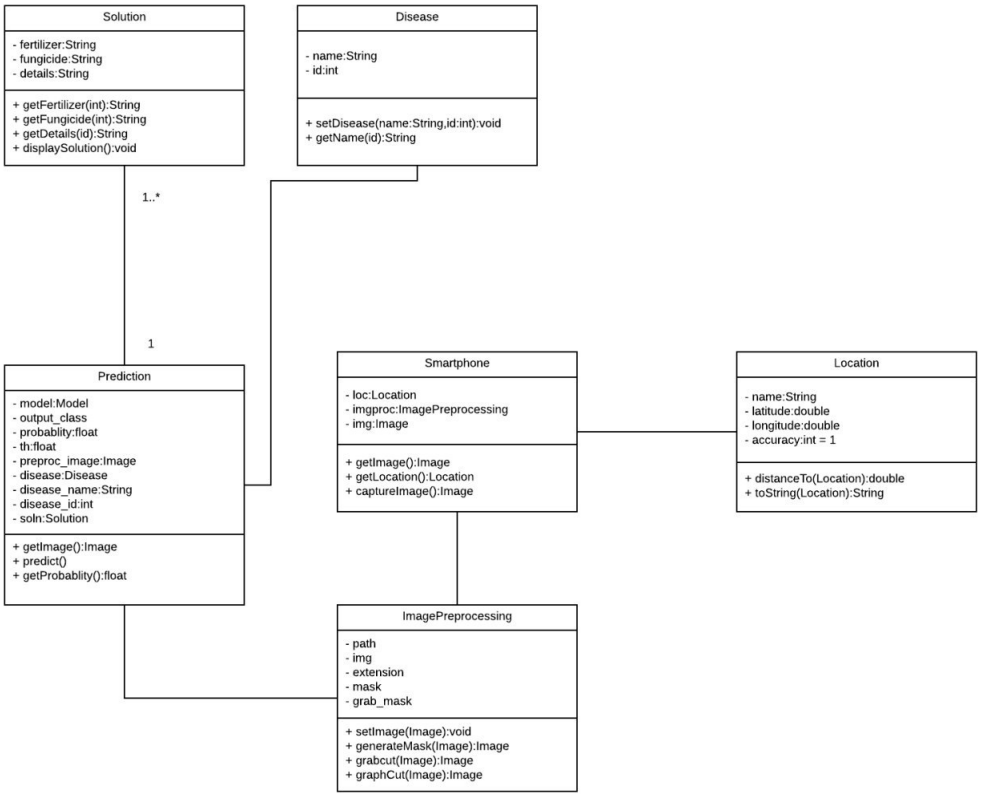


Figure 3.5: Class Diagram

Chapter 4

IMPLEMENTATION AND CODING

4.1 Introduction

This chapter covers the role of various subsystems/modules/classes along with implementation details listing of the code for the major functionalities. Section 4.2.1 outlines the details of various operational modules along with their functionalities. Section 4.2.2 lists the major classes used in the system and specifies their respective responsibilities. In 4.3.3 sample code snippets are provided for the major functional requirements. And finally in the section 4.4, a series of screenshots are displayed which illustrate the working and the flow of the system developed.

4.2 Operational Details

1. Image Acquisition and Output Retrieval
    - This is the first module which kick starts the system. The end user captures an image of the tomato leaf via a smartphone app.
    - This image is uploaded to the server by clicking the upload button.
    - The output, consisting of disease prediction and respective solutions are returned and displayed in this module as well.
    - The history of user queries is also maintained for reference.
    - It has following important classes:
        (a) ShowCaptureActivity: Captures/Loads image and stores it in internal memory
        (b) ShowLoadActivity: Uploads file to server. Displays server response (predicted disease).
        (c) DiseaseResult: Display the appropriate information for the predicted disease consisting of solutions.
2. Image Pre-processing
    - The image received from previous module is pre-processed by this module.

- The foreground is extracted from the image and the resultant image is resized to 256x256 dimensions.

3. Disease Prediction
   - This module takes the pre-processed image as input, which is then processed by the VGG19 CNN model stored on the server.
   - The CNN model predicts the condition of the leaf and returns this result back to the android interface.
4. Solution Suggestion
   - The prescribed treatment for the predicted disease is rendered in the form of web pages in the application.
   - The treatment consists of preventive and control measures which the farmer can take to counter the disease.

4.3 Major Classes

- Android
  1. ShowCaptureActivity - Capture image or load from gallery and store in internal device memory.
  2. ShowLoadActivity – Upload image to server and display response from server i.e the predicted disease.
  3. DiseaseResult – Show the information (HTML page) related to the predicted disease.
- Segmentation.py : Extract foreground from the uploaded image and then resize it.
- CNNmod.ipynb : Input the segmented image to the VGG19 model and obtain the predicted disease for the image.

4.4 Code Listing

Upload file to Server

```
private class UploadFileToServer extends AsyncTask<Void, Integer, String> {

       @Override
       protected String doInBackground(Void... params) {
           return uploadFile();
       }
       @SuppressWarnings("deprecation")
       private String uploadFile() {
           String responseString = null;
           HttpClient httpclient = new DefaultHttpClient();
           HttpPost httppost = new HttpPost(Config.FILE_UPLOAD_URL);
           try {
               AndroidMultiPartEntity entity = new AndroidMultiPartEntity(
                       new AndroidMultiPartEntity.ProgressListener() {

                           @Override
                         public void transferred(long num) {
                         publishProgress((int) ((num / (float) totalSize) * 100));
                           }
                       });
               // image or video path that is captured in previous activity
               filePath = getIntent().getStringExtra("filePath");

               File sourceFile = new File(filePath);
               // Adding file data to http body
               entity.addPart("image", new FileBody(sourceFile));

               totalSize = entity.getContentLength();
               httppost.setEntity(entity);              // Making server call
               HttpResponse response = httpclient.execute(httppost);
               HttpEntity r_entity = response.getEntity();

               int statusCode = response.getStatusLine().getStatusCode();
               if (statusCode == 200) {
                   // Server response
                   responseString = EntityUtils.toString(r_entity);
               } else {
                   responseString = "Error occurred! Http Status Code: "
                           + statusCode;
               }
           } catch (ClientProtocolException e) {
               responseString = e.toString();
           } catch (IOException e) {
               responseString = e.toString();
           }
           return responseString;
       }
       @Override
       protected void onPostExecute(String result) {
           Log.e(TAG, "Response from server: " + result);

           // showing the server response in an alert dialog
           showAlert(result);

           super.onPostExecute(result);
       }
   }
   private void showAlert(String message) {
       AlertDialog.Builder builder = new AlertDialog.Builder(this);
       builder.setMessage(message).setTitle("Response from Servers")
               .setCancelable(false)
               .setPositiveButton("OK", new DialogInterface.OnClickListener() {
                   public void onClick(DialogInterface dialog, int id) {
                       // do nothing
                   }
               });
       AlertDialog alert = builder.create();
       alert.show();
   }
```

Figure 4.1: Upload To Server Code Snippet

Figure 4.1 provides a snippet of the android code for uploading the selected image to the server

```
import cv2 as cv
import numpy as np
img = cv.imread('/content/drive/My Drive/BE PROJECT/hola.jpg')
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)

rect = (50,50,200,200)
res = cv.resize(img,(256,256), interpolation = cv.INTER_CUBIC)
mask = np.zeros(res.shape[:2], np.uint8)
cv.grabCut(res,mask,rect,bgdModel,fgdModel,6,cv.GC_INIT_WITH_RECT)
mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')

res = res*mask2[:,:,np.newaxis]
cv.imwrite('/content/drive/My Drive/BE PROJECT/2.jpg',res)
```

Figure 4.2: Segmentation and Re-size Code Snippet

Figure 4.2 shows the code snippet for the process of segmentation and re-sizing.

```
model = applications.VGG19(weights = "imagenet", include_top=False, input_shape = (img_width, img_height, 3))
for layer in model.layers[:15]:
        layer.trainable = False
x = model.output
x = Flatten()(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.5)(x)
x = Dense(1024, activation="relu")(x)
predictions = Dense(6, activation="softmax")(x)
model_final = Model(input = model.input, output = predictions)
model_final.compile(loss = "categorical_crossentropy", optimizer = optimizers.SGD(lr=0.001, momentum=0.9), metrics=["accuracy"])
train_datagen = ImageDataGenerator(
rescale = 1./255,
horizontal_flip = True,
fill_mode = "nearest",
width_shift_range = 0.2,
rotation_range=20)
test_datagen = ImageDataGenerator(
rescale = 1./255,
horizontal_flip = True,
fill_mode = "nearest",
zoom_range = 0.2,
rotation_range=20)
train_generator = train_datagen.flow_from_directory(
train_data_dir,
target_size = (img_height, img_width),
batch_size = batch_size,
class_mode = "categorical")
validation_generator = test_datagen.flow_from_directory(
validation_data_dir,
target_size = (img_height, img_width),
class_mode = "categorical")
checkpoint = ModelCheckpoint("vgg1922.h5", monitor='val_acc', verbose=1, save_best_only=False, save_weights_only=False, mode='auto', period=1)
early = EarlyStopping(monitor='val_acc', min_delta=0, patience=10, verbose=1, mode='auto')
model_final.fit_generator(
train_generator,
samples_per_epoch = nb_train_samples,
epochs = epochs,
validation_data = validation_generator,
nb_val_samples = nb_validation_samples,
callbacks = [PlotLossesKeras(),checkpoint, early])
```

Figure 4.3: CNN Model Training And Validation Code Snippet

Figure 4.3 shows the deep learning model

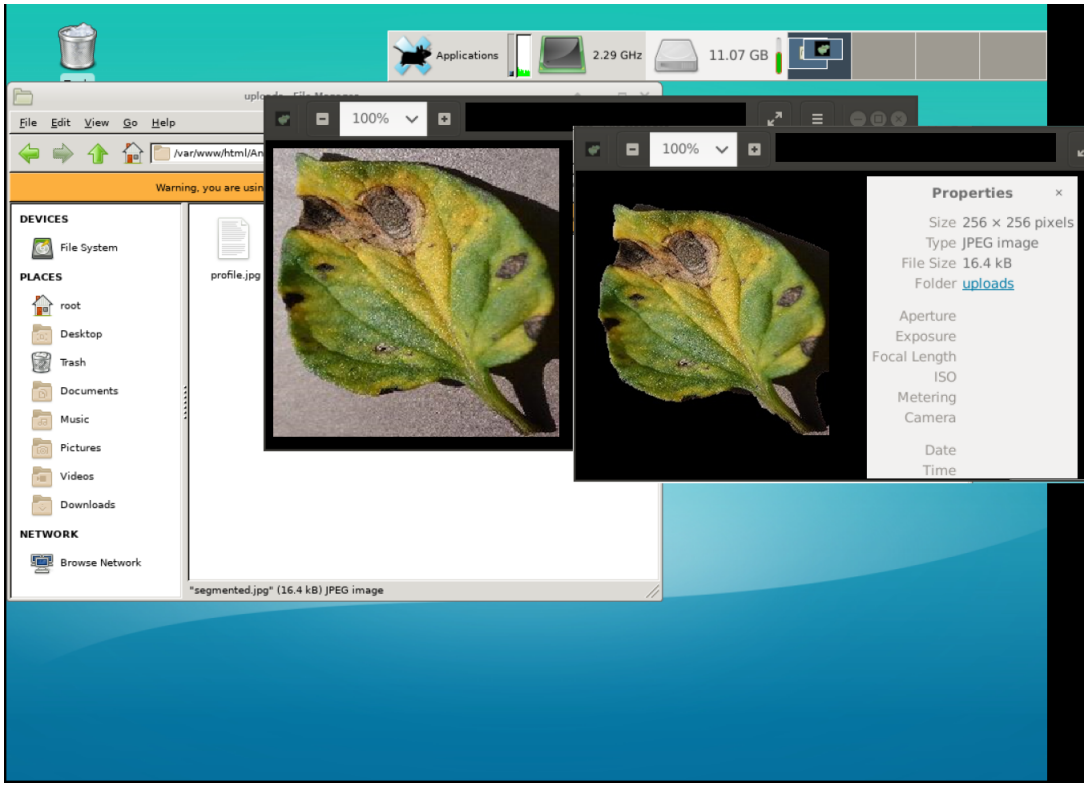


Figure 4.4: Screen-shot of Droplet

Figure 4.4 is a snapshot of the server droplet used to process and classify the uploaded image

Chapter 5

TESTING

Introduction
This chapter covers the testing approach used and the different test cases used for evaluation section 5.1 covers the unit testing with corresponding test cases for individual modules section 5.2 describes the integration testing test cases used nad section 5.3 shows the acceptance testing criterias required to be satisfied

5.1 Unit Testing

The term unit testing refers to the individual testing of separate units of a software system. It is a method of testing the correctness of a particular module of a program code to determine if they are fit to use. A unit is the smallest testable part of the application.

Table 5.1: Unit Testing

Test Item	Input	Output	Result
Segmentation	Raw Image	Segmented Image	Pass
Resizing	Segmented leaf image	Resized image	Pass
Data Item	Image / Textual data	Data transfer	Pass
Disease Prediction	Pre-processed image	Class of outcome	Pass
Solution Recommendation	Detected disease	Set of solutions	Pass

5.2 Integration Testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. It focuses on data communication among different modules. The objective is to take unit tested components and build a program structure that has been dictated by design. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested.

Table 5.2: Integration Testing

Test Item	Input	Output	Result
Segmentation and Resizing	Segmented Image	Resized Image	Pass
Disease Prediction	Pre-processed image	Class of outcome	Pass

5.3 Acceptance Testing

Acceptance testing is often done by the customer to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing. The purpose of this testing is to evaluate the systems compliance with the business requirements and assess whether it is acceptable for delivery. It is the formal testing with respect to user needs, requirements and the business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the customer to determine its acceptability.

Table 5.3: Acceptance Testing

Serial No.	Acceptance Criteria	Critical (Yes/No)
1	Capture, store and open the image	Yes
2	Predict the Disease	Yes
3	Returning appropriate Solution	Yes

Chapter 6

RESULTS AND DISCUSSION

6.1 Main GUI Snapshots

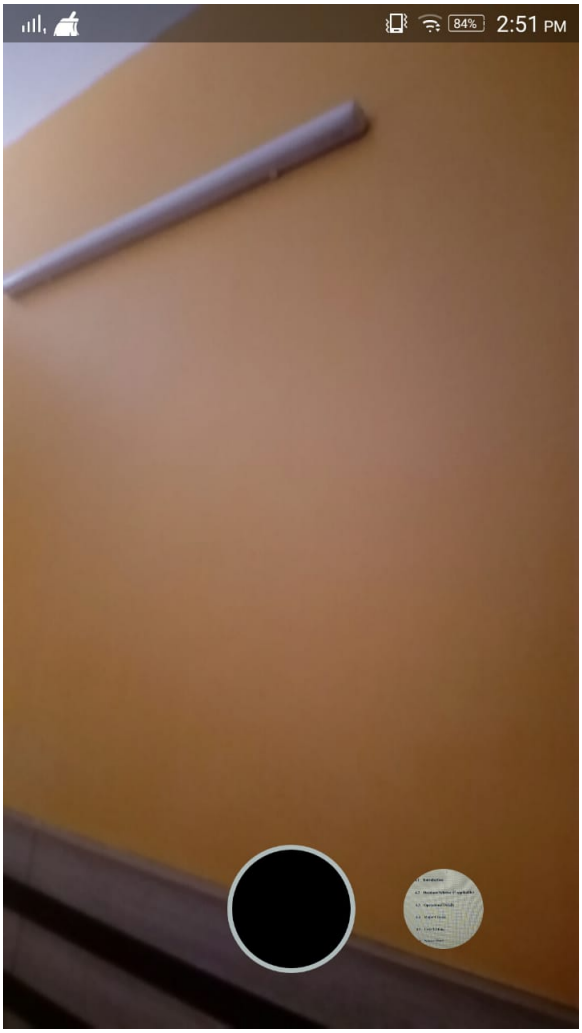


Figure 6.1: Camera Fragment

Figure 6.1 displays the home screen which is displayed when the app is launched. it consists of 2 options namely image capture and select image from gallery

Figure 6.2: Notification Fragment

Figure 6.2 shows the notification fragment which contains the previous queries made by the user

Figure 6.3: Image Selection Screen

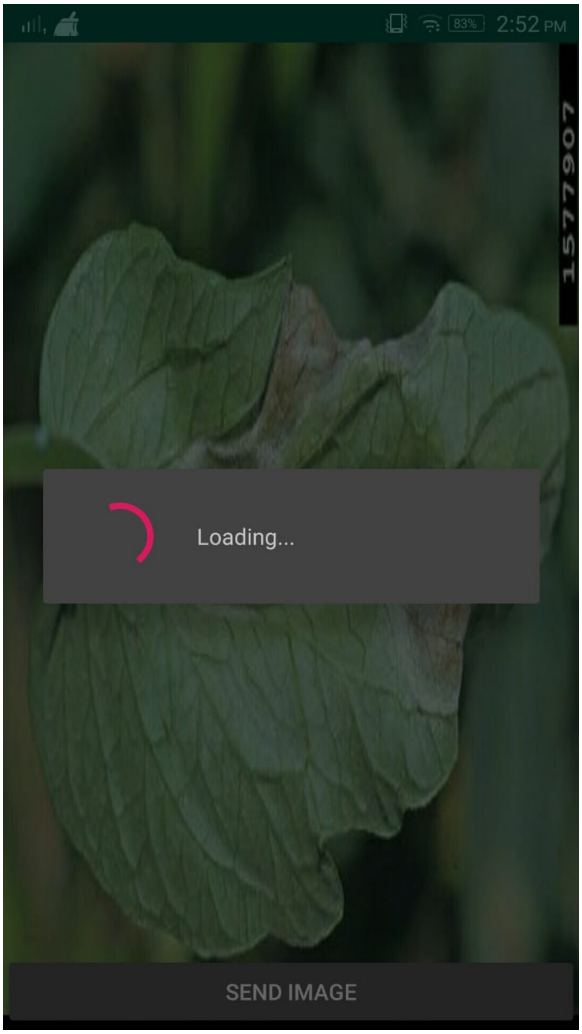


Figure 6.4: Loading Response Prompt

Figures 6.3 and 6.4 display the selected image and sends it to the server when ”send image” button is clicked.

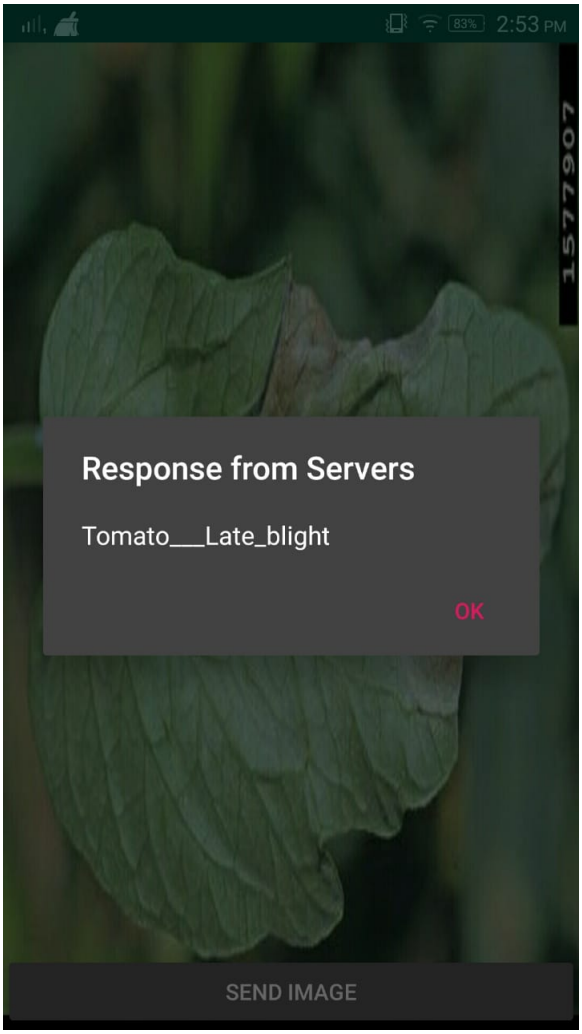


Figure 6.5: Response Prompt

The server response is shown in Figure 6.5 in the form of an alert notification

Figure 6.6: Solution Screen

The solutions and counter measures are displayed in Figure 6.6 for the detected disease. There is an option to go back to the home screen

Figure 6.7: Unsegmented Image

Figure 6.7 shows a sample unsegmented image, as captured by the user.

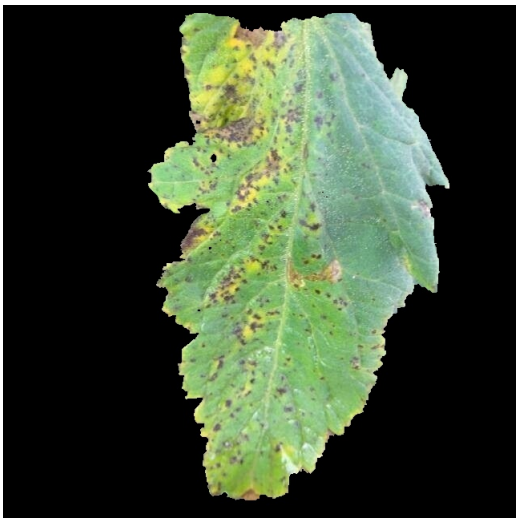


Figure 6.8: Segmented Image

Figure 6.8 shows the Figure 6.7 after segmentation and re-sizing.

6.2 Graph

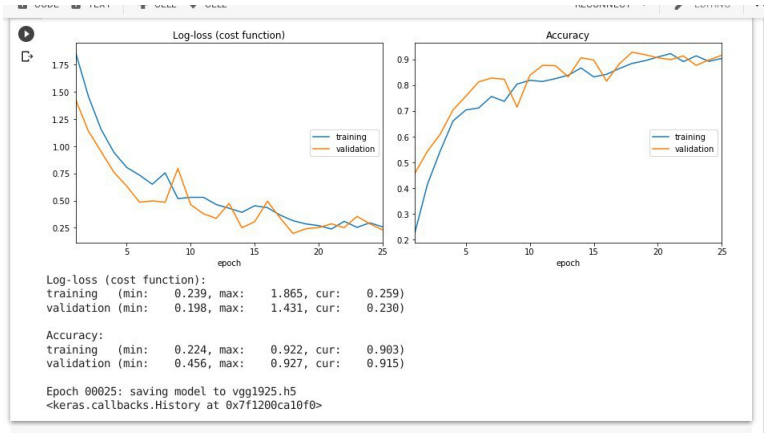


Figure 6.9: Graph of model for 0-25 epochs

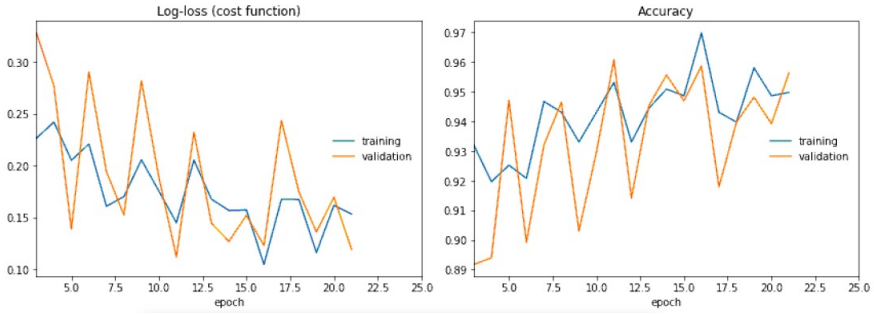


Figure 6.10: Graph of model for 25-50 epochs

Figures 6.9 and 6.10 show the graphs displaying the performance of the model for 0-25 epochs and 25-50 epochs.

6.3 Discussion

AgroAid is a deep learning based android tool. It allows images to be captured, processed, classified. The application runs perfectly on the deployed devices running Android version 5 and above. It allows the user to upload the images which are then processed and classified on the server and the response is returned to the android application. High accuracy is obtained on dataset whereas real time accuracy varies from disease to disease. All features deployed work without crashes on the target devices. The GUI is simple and intuitive.

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

Thus, this report provides an overview of the entire project in sufficient detail.The developed system serves as a tool for the farmer to diagnose a disease in tomato plants and take appropriate measures to counter it. The system has a simple, intuitive interface and high usability so that any person with minimal smartphone knowledge can use it effectively. The results obtained are promising and can further improve through retraining on more image data.Deep learning proves to be the most promising method for image analysis and computer vision. The treatment side of the application makes it more comprehensive, and also proactive rather than simply reactive as it specifies preventive measures as well.

7.2 Future Work

In order to maintain and improve the model's accuracy, it should be periodically trained on new data. A supplementary feature can be added which stores and pinpoints the location of the diseased region on the farm which will help the farmer locate the exact location where the disease has originated. The solution (fertilizers and fungicides) should be provided to the farmer taking into consideration the N-P-K values of the soil so that a proper dosage can be given. Also, forecasting of diseases can be done based on the weather and soil conditions in the future using machine learning.

References

[1] H. Durmuş, E. O. Güneş and M. Kırcı, ”Disease detection on the leaves of the tomato plants by using deep learning,” 2017 6th International Conference on Agro-Geoinformatics, Fairfax, VA, 2017, pp. 1-5.

[2] M. Dhakate and Ingole A. B., ”Diagnosis of pomegranate plant diseases using neural network,” 2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)

[3] A. K. Hase, P. S. Aher and S. K. Hase, ”Detection, categorization and suggestion to cure infected plants of tomato and grapes by using OpenCV framework for andriod environment,” 2017 2nd International Conference for Convergence in Technology (I2CT), Mumbai, 2017,

[4] J. Shijie, J. Peiyi, H. Siping and s. Haibo, ”Automatic detection of tomato diseases and pests based on leaf images,” 2017 Chinese Automation Congress (CAC), Jinan, 2017

[5] M. Islam, Anh Dinh, K. Wahid and P. Bhowmik, ”Detection of potato diseases using image segmentation and multiclass support vector machine,” 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, ON, 2017

[6] S. P. Mohanty, D. Hughes and M. Salathe, ”Using Deep Learning for Image-Based Plant Disease Detection,” eprint arXiv:1604.03169, 2016.

[7] Pawar B.T. “Report of Bacterial Diseases of Tomato from Marathwada Region of Maharashtra, India ” Research Journal of Recent Sciences ISSN 2277-2502 Vol. 3(ISC-2013), 302-304 (2014)

[8] Karthik Praburam. N "Detection And Prevention Of Banana Leaf Diseases From Banana Plant Using Embeeded Linux Board " 2016 Online International Conference on Green Engineering and Technologies (IC-GET)

[9] Pranjali B. Padol Prof Anjali A. Yadav "SVM Classifier Based Grape Leaf Disease Detection" 2016 Conference on Advances in Signal Processing (CASP) Cummins College of Engineering for Women, Pune. Jun 9-11, 2016

[10] Suyash S. Patil Sandeep A. Thorat "Early Detection of Grapes Diseases Using Machine Learning and IoT" 2016 Second International Conference on Cognitive Computing and Information Processing (CCIP).

[11] Manjiri M Deshpande, Santosh T Dhotre amp; DJ Vanmare, Critical survey of fungal diseases on tomato plant in Marathwada region.

[12] D. H. Mitranavar estimation of post harvest losses of major fruits and vegetables in karnataka – a management appraisal

[13] Pawar B.T. Report of Bacterial Diseases of Tomato from Marathwada Region of Maharashtra, India

[14] http://www.saferbrand.com/articles/common-tomato-plant-problems-how-to-fix-them
[15] http://vikaspedia.in/agriculture/crop- production/integrated-pest-managment/ipm-for-vegetables/ipm-strategies-for-tomato/tomato-diseases-and-symptoms
[16] `https://keras.io/applications/`