Plant Disease Diagnostics using UAV and Android APP (MedCrafter)

Authors

Somia Imdad (UET/14-CS-50)

Abdul Haseeb (UET/14-CS-03)

Muhammad Usama Bin Islam (UET/14-CS-66)

Supervised By

Mr. Muhammad Wakeel Ahmad

Department of Computer Science

UNIVERSITY OF ENGINEERING AND TECHNOLOGY TAXILA

July, 2018

DEDICATION

Dedicated to

“The Teacher of the Universe”

(Peace be Upon Him)

With whose existence
and
by having the charity of His knowledge
the cosmos got illuminated with the light of
insight and wisdom
and
the journey of human enlightenment was made possible.

ACKNOWLEDGEMENT

Praise to Allah Almighty, Lord of the worlds, the Merciful and the Beneficent, who gave us strength, thoughts and co-operative people to enable us to accomplish this goal and fulfill the required functionalities.

This was all not possible without the guidance, continuous appreciation and moral support by our honorable Supervisor **Mr. Muhammad Wakeel Ahmad.** He was always there whenever we need his help and ideas. We are really thankful to him who made our concepts clearer. We are also thankful to Chairman of Computer Science department, **Dr. Syed Aun Irtaza** and **Dr. Syed Adnan** for helping us.

At last, we would like to acknowledge all of the assistance and contributions of UNIVERSITY OF ENGINEERING AND TECHNOLOGY, TAXILA for supporting us with all that is needed starting from the books, and ending with the full care that it is providing us with, to help us to be professionals in the field of Information Technology.

DECLARATION

We hereby declare that we have developed this application and accompanied report entirely on the basis of our personal efforts. Not any of the portion of the application work presented has been submitted of any application for any other qualification or degree of this or any other university or institute of learning.

Students Name & Signature

Student 1	Student 2	Student 3
Somia Imdad	Muhammad Usama Bin Islam	Abdul Haseeb
_____	_____	_____

CERTIFICATE OF APPROVAL

It is to certify that the final year project of BS (CS) "**Plant Disease Diagnostics using UAV and Android App (MedCrafter AGRIDRONE)"** was developed by **Muhammad Usama Bin Islam, Somia Imdad** and **Abdul Haseeb** under the supervision of "**Mr. Muhammad Wakeel Ahmad**" and that in his opinion, it is in scope, fully adequacy and quality of the degree of Bachelors of Science in Computer Sciences.

External Examiner

Supervisor

Mr. Muhammad Wakeel Ahmad Rao

Lecturer, Computer Science Department

UET, Taxila

Head of Department

Dr. Syed Aun Irtiza

Department of Computer Science

UET, Taxila

ABSTRACT

Plant diseases causes many significant damages and losses in crops around the world. Some appropriate measures should be introduced on identification of plant disease to prevent damages and minimize losses. Early Detection of Disease helps in increasing the productivity of crop as well as in minimizing expense. Technical approaches using machine learning and computer vision are actively researched to achieve intelligence farming by early detection on plant disease. The accuracy of object detection and recognition systems has been drastically improved by the recent development in Deep Neural Networks. By using these systems and implementation of computer vision and machine learning techniques, plant diseases can easily be detected. Here we have used transfer learning based approach to diagnose diseases of different plants using its images captured by camera devices either drone or smartphone. Our goal is to build a market oriented product for Plant Disease Detection, a smartphone app compatible with both smartphone camera and drone camera. The target group of the user is those who request a quick diagnosis on common leaf disease at any time of the day i.e. Farmers, agricultural industries, agricultural consultants and Government Agencies & Departments.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER # 1

INTRODUCTION

1 INTRODUCTION

In this chapter, we will introduce this application, software tools, problem statement, objectives, scope of this application, proposed application, motivation of this proposed solution, relevance to courses and tools & techniques which are used to implement this application.

1.1 Introduction & History

In agriculture sector, plant diseases and pests are a major challenge. An accurate and fast detection of plants diseases could help to develop an early treatment method while at the same time significantly reducing economic losses.

Techniques based on image processing are used to detect plant diseases without causing any secondary impact in plant. However, the accuracy is still a challenge in Computer Vision. In intelligent systems, most important issue is the use of proper dataset. Our technique is applicable on all kinds of plants but for testing, we focus on the recognition and identification of diseases that affect tomato plants (we are not working on pest detection due to lack of dataset). Economically, tomato is the most important vegetable crop around the world, and its production has been significantly increased through the years. Several diseases that affect tomato plants are the main part of this study. Testing is done on some samples of our collected dataset which contains different diseases of tomato plants. The tomato diseases on which testing is done are late blight and bacterial spot.

It is a machine learning based approach in order to detect diseases in tomato plants using drone or smartphone camera. It will help the Farmers, agricultural industries, agricultural consultants and Government Agencies & Departments to perform quick diagnosis on common leaf disease at any time. Most probably this is the first time someone doing in the UET Taxila.

Our system uses images of diseased plants taken from different tomato farms and fields, thus we don't need to analyze samples in laboratory. We collected dataset of images gathered from fields and greenhouses using drone and android mobile camera. Techniques like image annotation, image augmentation are applied on images in order to increase the dataset and accuracy of images. For training those images, an open-source software library called Tensor Flow is used. Farmers will be able to take advantage from this application whenever they want early detection of plants diseases, they can do it easily by using android app.

MedCrafter AGRI-DRONE has following features:

1. Detect plant disease more precisely.
2. Save time required for infield scouting & actions.
3. Help the farmer to identify crop's progress accurately and in replanting decisions.
4. Help in preventing unnecessary waste of financial resources.
5. Smart phone diagnostic app will keep farmer updated about crop at a single click.

Since this project is android based also, so users can access this tool from multiple android devices to recognize different image targets. This application just requires input images captured by various camera devices with different resolutions, such as cell phone and other digital cameras. The goal of this project is to develop a market-oriented product for Plant Disease Diagnosis, a smartphone app compatible with both smartphone camera and drone camera. Moreover, it can efficiently deal with different size of objects, different illumination conditions and background variations contained in the surrounding areas of diseased plant. This is not much time consuming as compared to older ways of plant disease detection techniques. The idea is productive and as well as innovative. Application like this is never created in the history of UET Taxila.

The MedCrafter AGRI-DRONE provides a practical and real time market oriented product that can be used in different fields without employing any complex and expensive technology. It also considers the possibility that a plant can be simultaneously affected by more than one disease in the same plant.

1.2 Statement of the problem

The economy of agricultural countries heavily depends upon number of crops grown in the country. Moreover, the rapidly increasing population (especially those residing in rural areas) also depends upon agriculture for subsistence. In Pakistan, agriculture contributes to 24 percent of national GDP. It feeds whole rural and urban population.

But crops diseases have been a continuous threat to farmer's livings and equally endanger the food security of the world. Diseases on plants leads to the greater number of decrease in both quality and quantity of agricultural products. The reason behind this failure is, the naked eye observation of the expert/farmer is in practice, which is time consuming and not feasible for larger fields.

Machine learning based detection and recognition of plant diseases provide hints to detect the diseases in early stages. Comparatively, identifying the plant diseases visually is expensive, difficult and inefficient. Also, it requires the higher level of expertise of trained agricultural experts and botanists.

1.3 Introduction to the Software tools and Technologies

In order to develop this final year project, following software tools are used:

1. Ubuntu 16.04 LTS cloud server
2. Python
3. LableImg
4. Tensorflow
5. Android Studio
6. Pycharm
7. Flask

1.3.1 Why Ubuntu 16.04 LTS cloud server is used?

Ubuntu is a free operating system and Linux distribution based on debian. It is widely used by developers across the world for its low system requirements and open source basis.

From development to deployment we are using Ubunto 16.04 LTS cloud and on local environment. One main reason is that TensorFlow is basically for linux based systems, although it is also supported

on windows but support, installation and computing power on linux is far better than Windows and MAC. Moreover, Ubuntu is secure, totally free with strong community support. After a lot of thinking and discussion we choose Ubuntu for our development. Our deployment is on Ubuntu Cloud managed by Aruba Cloud.

1.3.2 Why Pycharm is used?

PyCharm, an integrated development environment specifically designed for python programming provides every aspect of code management, code analysis, unit testing and integration with version control systems. It also provides special support for **Flask**.

As we were going to develop our website using Flask and machine learning part on Tensorflow which is also python based so we decided to use PyCharm as an IDE.

1.3.3 Why Android Studio is used?

Android Studio is the official integrated development environment for android development. It provides code management, one-click solution for many problems faced by android developers and it also provides support for version controlling. So, we have chosen android studio for developing our android application.

1.3.4 Why TensorFlow is used?

TensorFlow is an open source and fast machine learning system that operates at large scale, having a lot of community support. It is also python based and as we decided to develop our website using python so using TensorFlow for Machine Learning was the best choice to gain more exposure of the language. [1] [1-6]

1.3.5 Why LabelImg is used?

LabelImg is a tool used for annotation and labeling of images. It outputs the data in the form of PASCAL VOC. For feeding our values to F-RCNN for transfer learning we needed TFRecords of dataset. So, by using ready-made script provided by TensorFlow community we easily converted PASCAL VOC dataset to TFRecord.[7]

1.3.6 Why Python is used?

Python is easy to understand and its packages are free to use under GNU license. Many machine learning engineers use python. So we have chosen python for it.

1.3.7 Why OpenCV is used?

OpenCV is a python library aimed at real-time computer vision. We used OpenCV for capturing IP stream and extracting frames from it. We also used OpenCV for changing dimensions of frames.[8]

1.4 Objectives

The objectives of any project are important to achieve its goal. The main objectives of our product oriented project are:

1. To develop a practical, reliable and inexpensive real time application that can be used in fields.
2. To develop a market-oriented product for Plant Disease Detection, a smartphone app compatible with both smartphone camera and drone camera.
3. It considers the possibility that there can be more than one disease same plant is affected.
4. Moreover, android application will update the user about weather forecast daily and hourly, so that user can prepare work plan carefully.

The diseases in the plants may show various physical characteristics i.e. difference in colors, shapes and forms etc. We are using a machine learning based approach using Object Detection API and Faster RCNN pre-trained modal for distinguishing different diseases. Based on these facts [5, 7, 9, 10], following characteristics are analyzed:

1. **Locations symptom:** Diseases affect not only leaves of the plant but other parts like stems and fruits also.
2. **Disease patterns:** Visible variations are shown on either front or back side of leaves by the symptoms of diseases.
3. **Fungus type**: Some diseases can easily be differentiated by the identification of type of fungus.

1.5 Proposed Solution

In this project we are proposed to develop UAV drones and smart phone-based system which would help the farmers to diagnose diseases of crops earlier. Machine learning based technology will assist farmers and agricultural agencies to scout areas of fields. Farmers can use either drone or smartphone camera to monitor their fields in less time using MedCrafter AGRI-DRONE android application. This will also save the cost of labor associated with manual infield scouting. An early identification of affected areas will also reduce pesticide application rates and cost. This application will increase the overall growth of crops and no need for identifying the diseases in long time. This application has user friendly interface. All communications between all the users is done through an android device, UAV drone and a remote controller. An android app is the cost-effective solution without expensive hardware or software requirement. The images acquired by UAV Drone/ Smart phone will be processed on an online server using Tensorflow, OpenCV and flask.

A transfer learning based approach with the help of TensorFlow and Object Detection API is used for this purpose[4]. A visual representation of it is shown below in **Figure 1.1**

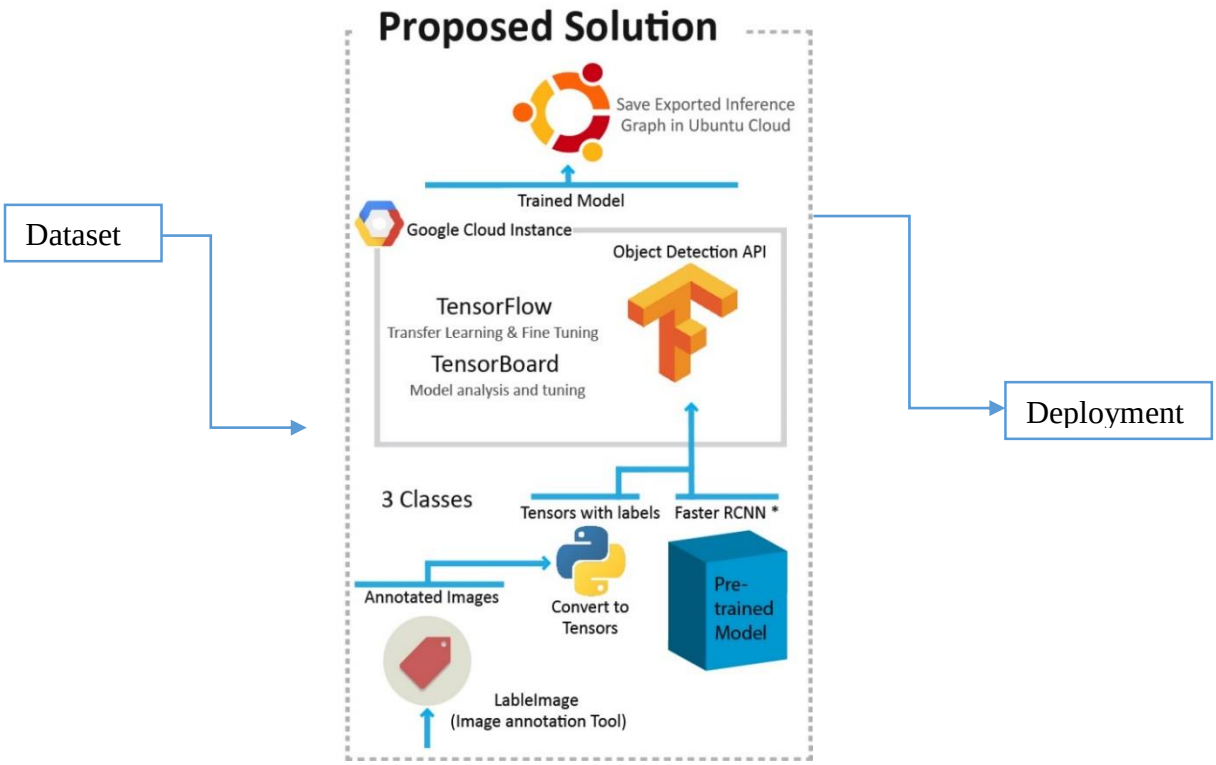


Figure 1.1 Proposed Solution

1.6 Motivation

MedCrafter AGRI-DRONE saves significant amount of time and money of farmers and agricultural industries. Despite the existence of many learning Applications, the following features are the motivational source of our project:

1. Cost optimization in terms of developing and deploying the overall application.
2. This application is really very helpful and easy to use for any layman farmer to detect diseases of plants.
3. We are developing a market-oriented product i.e. Agri-drone, an android application and a web application.
4. Android application also show weather forecasting details.
5. All the update is given to the farmer on one click via mobile app.

1.7 Scope of Proposed Solution

This project is designed to assist individuals (farmers and agricultural experts) to get updates about their fields by using machine learning and artificial intelligence technology. This is the first time someone is doing this kind of project in the history UET Taxila. Farmers will be able to take advantage

from this product whenever they have shortage of time and they get all the updates about their fields using our product. Our system uses the video stream of plants, thus avoiding the process of collecting and capturing individual samples manually and then analyzing. It is a market oriented real time product that is less costly and easy to implement. It can work with any video streamed through various camera devices (e.g. drone or smartphone) with different resolutions. Moreover, it can effectively handle background variation and different illumination conditions. It considers the possibility that there can be more than one disease same plant is affected.

1.8 Relevance to Courses

1.8.1 Software Design Project-I (CS-400)

This course helped us to evaluate our development methodologies which are used in our project.

1.8.1 Introduction to Database Systems (CS-203)

This course helped us in database management.

1.8.2 Smart Application Development (CS-311)

This course helped us in developing android smart application.

1.8.3 Web Technologies (CSC271)

This course helped us in developing web-based application.

1.8.4 CS Elective-VI (System Programming)

This course helped us gaining better understanding of how python works and implementing the advanced concepts of Flask.

1.9 Tools & Techniques

"MedCrafter AGRI-DRONE" in an Android and web-based application and a market-oriented product consists of software and hardware tools and technologies. Hardware details are given in section 1.9.1, Software details are given in section 1.9.2.

1.9.1 Hardware Details

1. DJI Phantom Drone
2. 4 GB RAM
3. 5 GB HHD GPU (not necessary) Tested on Corei3
4. Minimum Android API 25
5. DJI Drone with RC (Any Modal)
6. Android device

1.9.2 Software Details

1. Ubuntu 16.04 cloud server
2. Pycharm
3. Android Studio
4. Tensorflow
5. LableImg
6. Python3

7. OpenCV

8. Cloud

9. Cython

10. Android SDK

11. Android NDK

12. JupyterHub + Jupyter Notebook

13. DJI SDK for drone

CHAPTER # 2
STATE OF THE ART

2 STATE OF THE ART

In this chapter, we will discuss what is TensorFlow? What is Machine Learning? What is Artificial Intelligence? How we'll implement this application in real world? Why we need it, if other applications exist? What is the pros and cons of existing applications and proposed application?

2.1 What is TensorFlow?

According to official website of tensorflow

> ***TensorFlow*** *is an open source software library for high performance numerical computation. Its flexible architecture allows easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices. Originally developed by researchers and engineers from the Google Brain team within Google's AI organization, it comes with strong support for machine learning and deep learning and the flexible numerical computation core is used across many other scientific domains.*

2.2 What is Machine Learning?

According to medium.com

> ***Machine learning is a subfield of artificial intelligence. Its goal is to enable computers to learn on their own.*** *A machine's learning algorithm enables it to identify patterns in observed data, build models that explain the world, and predict things without having explicit pre-programmed rules and models.*

2.3 What is Artificial Intelligence?

According to medium.com

> ***Artificial intelligence is the study of agents that perceive the world around them, form plans, and make decisions to achieve their goals.*** *Its foundations include mathematics, logic, philosophy, probability, linguistics, neuroscience, and decision theory. Many fields fall under the umbrella of AI, such as computer vision, robotics, machine learning, and natural language processing.*

2.4 Implementation in real world

This application can be implemented by Farmers, agricultural industries, agricultural consultants and Government Agencies & Departments to perform quick diagnosis on common leaf disease at any time. Anyone will be able to download the android version of this application from Play Store after some time. Users can easily use all the three versions (android app, web app and UAV Drone) of this application by having friendly user interface. User will find it efficient and less time consuming.

2.5 Existing Applications

2.5.1 Assess

There is an expensive and commercial software, called Assess, which can quickly and easily tell the quantity of disease's symptoms of plants.

2.5.2 Leafsnap

There is an excellent iPhone app, called Leafsnap, which help the user to identify plant species from images of leaves using visual recognition.

2.5.3 Leaf Doctor

This is an android app, which performs quantitative assessments for plant diseases on diseased leaves. User collect snaps of diseases plants and calculate the percentage of diseased tissue. The algorithm employs user-specific values for up to 8 colors of healthy tissues in picture. The color of each pixel is then evaluated for its distance from the healthy colors and assign a status of healthy or diseased leaf. The assessment data may be sent by email to the recipients.

2.6 Drawbacks of Existing Applications

Drawbacks in above mentioned existing application are as follows:

1. Low accuracy.
2. First capture image and upload it.
3. Hardware dependent requires a computer system, which is inconvenient for the user to use any time.
4. Weather Forecast is not available.
5. Just android application, no web application or UAV drone.

2.7 The Necessity

There are some necessities for this market-oriented product, android and web-based application to conduct meetings are as follows:

2.7.1 Dataset

Must need to have images as the developer has used in project. We have collected a large number images of tomato plants, which contain the diseases depending on the farm where they were taken from. Some farms presented more infected plants by canker, early blight, late blight and plague, and others by leaf mold, bacterial spot and gray mold.[7, 11-13] The rest of diseases were found in each

Figure 2.1 Tomato Late Blight

farm but in less presence. Our technique is applicable on all kinds of plats and its diseases, but testing is done on the tomato plants affected by late blight and bacterial spot. The infections of Late blight produce firm lesions, dark brown as shown in **Figure 2.1** which may destroy the entire tomato fruit.

2.7.2 Accurate Results

MedCrafter AGRI-DRONE must show accurate results.

2.7.3 Better and quick Analysis

MedCrafter AGRI-DRONE must provide the facility of quick and efficient analysis and detection of diseased plants as compared to the previous conventional ways of analysis.

2.8 Application Architecture

The problems which are occurred in the existing application are overcome in proposed application. In this application we are implemented all of the functionality by using any mobile device, UAV drone and a web application.

The proposed application supports fast and efficient detection and analysis of diseased images as compared to existing applications.

2.8.1 Application Architecture

MedCrafter AGRI-DRONE consists of five features:

1 Machine vision based technology using transfer learning and fine tuning of Faster-RCNN.
2 A smartphone based low cost android application for monitoring small fields.
3 A drone based solution for covering large area.
4 Android application also forecasts weather daily & hourly and predicts accurate weather information.
5 Reduction in the time and work load.

How we achieved these is explained in **Figure 2.2** below.

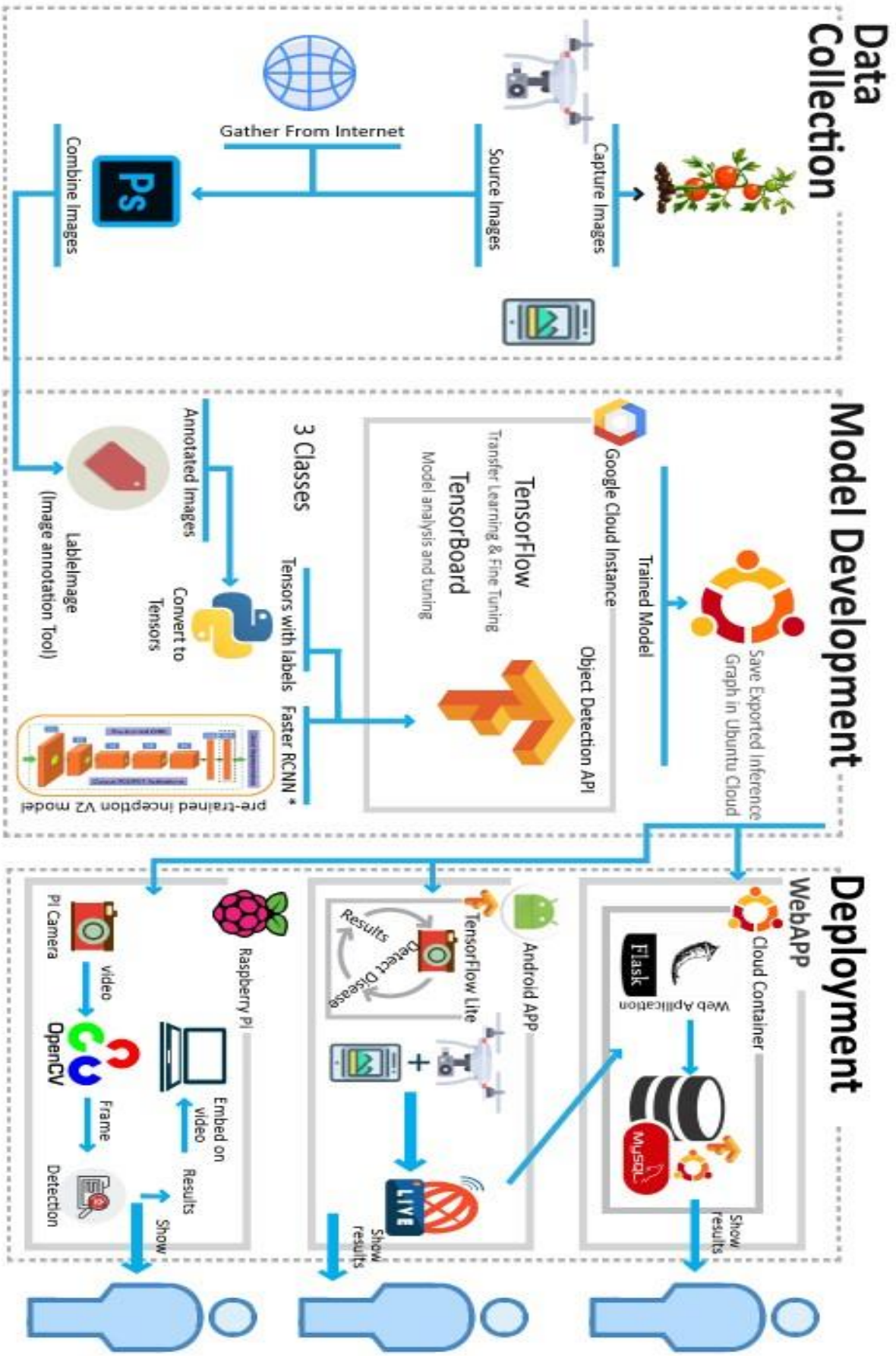


Figure 2.2 Application Architecture

CHAPTER # 3

METHODOLOGY

&

WORKPLAN

3 METHODOLOGY & WORKPLAN

In this chapter, we will discuss that what are the existing methodologies and which one we have chosen for implementation of this project in an effective way, also we will discuss advantages of adopted methodology [14, 15].

What is methodology and why we need it?

Whenever a small or large project have started to develop, first thing all of programmers required is methodology. Methodology is a way of developing a project, in which all of the programmers gather the user's requirements, design the project, implement it, and after all this testing and maintenance of the project, in a satisfaction of user and according to the project requirements.

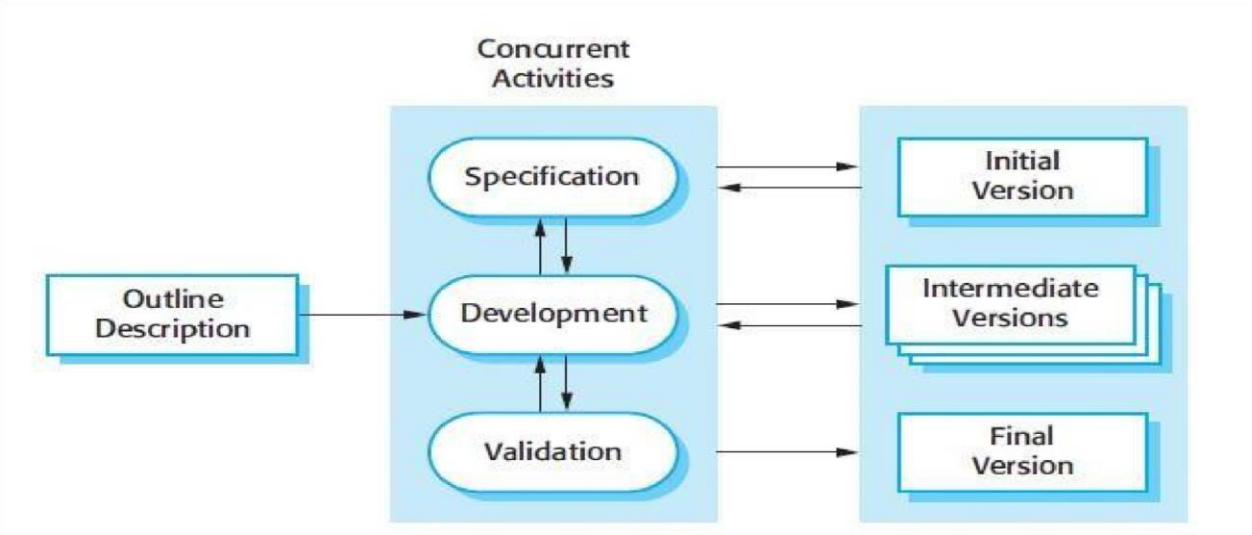


Figure 3.1 Adopted Methodology

3.1 Adopted Methodology

Incremental model is used to develop this project, in which we divided our work in multiple modules. All these modules are further divided into more easily managed modules which made up the actual implementation of the requirements.

Reason behind using this model is:

- It is easy to test and debug the product during iterations.
- We incrementally developed our project and handled efficiently any changes made by supervisor.

3.2 Project Time Allocation

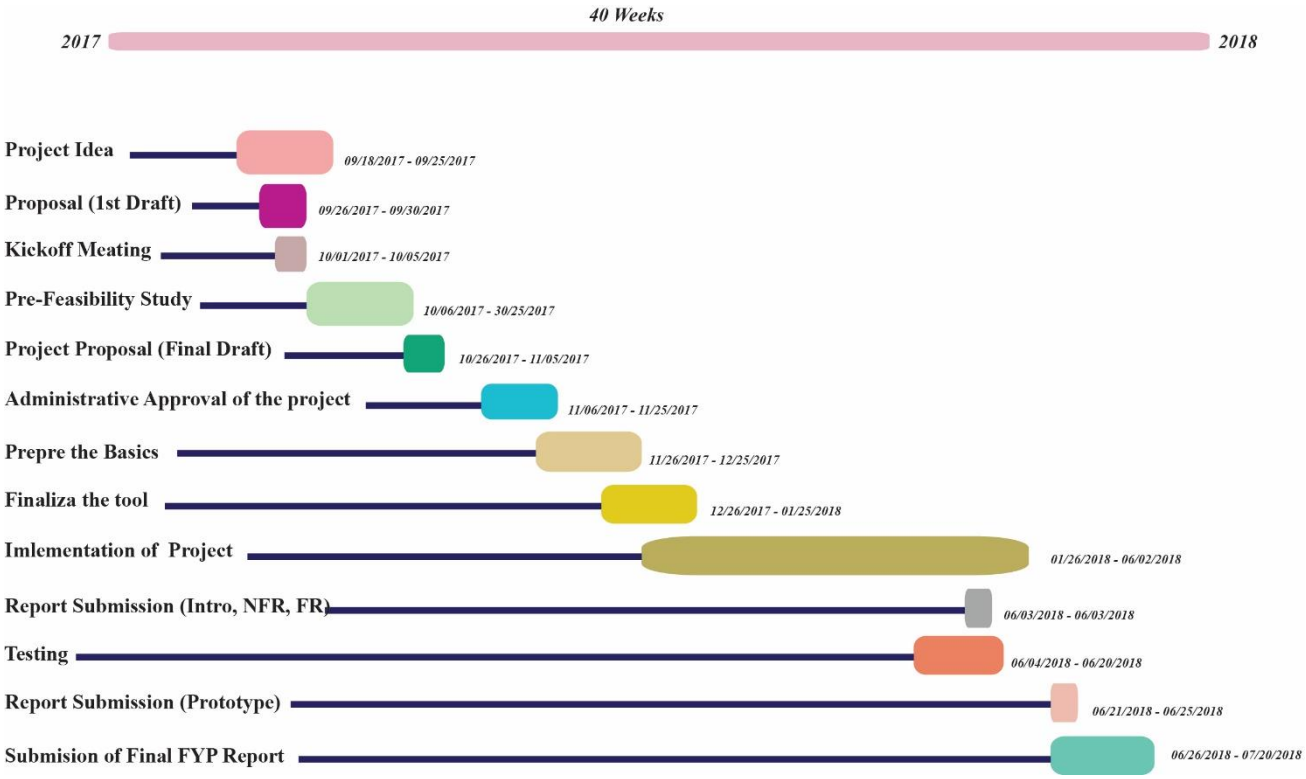


Figure 3.2 Project Time Allocation

3.3 Key Milestones

Table 3.1 Key Milestones with dates

Key Milestones			
Sr. No	Time Spent	Milestones	Deliverables
1.	1 Months (September 18,2017- October 18,2017)	Analysis & Requirements	Research Proposal
2.	5 Days (October 19,2017 –October 24, 2017)	Planning, scheduling	Report
3.	1 Month (October 25, 2017 -November 25, 2017)	Modeling & design	Report
4.	6 & half Months (December 20, 2017-June 25, 2018)	Coding & testing	Software System
5.	7 Days (June 26, 2018-July 3, 2018)	Deployment	Software System

3.4 Roles & Responsibilities

Project development team is consisting of three members. To accomplish a goal, documentation and development is equally distributed among them and each member work on parallel to avoid wastage of time.

CHAPTER # 4

SYSTEM ANALYSIS & DESIGN

4 SYSTEM ANALYSIS & DESIGN

In this chapter requirements analysis, feasibility study, planning, forecasting, modeling, scheduling and design of the project is discussed. For developing any project, the major problem is requirement gathering. We will also focus on functional and non-functional requirements.

The procedure for gathering requirements has its own defined procedure according to the complexity of the application. To define project schedule and processing, different models and techniques also focused on this chapter.

4.1 Requirements Gathering Techniques

A requirement can be defined as a condition or capability that must be processed by a product or an application. Techniques that can be used for collecting requirements are as follows:

- By analysis and observations
- Using software tools
- Using techniques of Artificial Intelligence and Machine Learning

The techniques we have used to collect requirements are observations and knowing about AI new techniques.

4.2 Requirement Analysis

Requirements analysis is the process of planning, forecasting and studying the overall former needs of the application requirements. Requirements analysis is further divided into two parts based on our project:

1. Functional Requirements
2. Non-Functional Requirements

4.2.1 Functional Requirements

Following are the functional requirements of our project:

REQ-F1: Must provide ability to connect with DJI drone.

REQ-F2: Must show IP address and port number of android device for capturing stream.

REQ-F3: Must stream video to server.

REQ-F4: Must provide disease diagnosis feature natively using smartphone camera.

REQ-F5: Must show detailed 7 days forecast.

REQ-F6: Ability for the user to read about different diseases and counter measures.

4.2.2 Non-Functional Requirements

Non-functional requirements are the constraints or checks on the services and functions provided by an application such as constraints on the development standards/process and constraints of time etc.

Non-Functional requirements of Med-Crafter are as follows:

REQ-NF1: Application shall provide better response and performance.

REQ-NF2: Application shall provide good quality results

REQ-NF3: Application must be efficient.

REQ-NF4: Application must be user interactive.

REQ-NF5: Application shall detect plant diseases more precisely.

REQ-NF6: Application must be less time consuming.

4.3 Application Quality Attribute

4.3.1 Availability

Application must be responsive and available at every time.

4.3.2 Maintainability

Making modifications or upgrade-potential in the utility will not be so much difficult. A little knowledge about machine learning, transfer learning, python and Tensorflow will be enough to make changes or upgrading the software/utility.

4.3.3 Consistency

When a developer is updating information, consistency must hold there.

4.3.4 Portability

MedCrafter AGRI-DRONE is a market-oriented product, portable UAV drone, an android based mobile application and a web application that is why there is no problem in portability to use it cross the android versions and systems for processing.

4.3.5 Database Requirements

In this section, we are using MySQL and SQLite for storing the user's data and classification results.

4.4 Use Cases

We draw several use cases for analysis of our project. Some of the use cases are as follows

4.4.1 UC1: Starting of Application

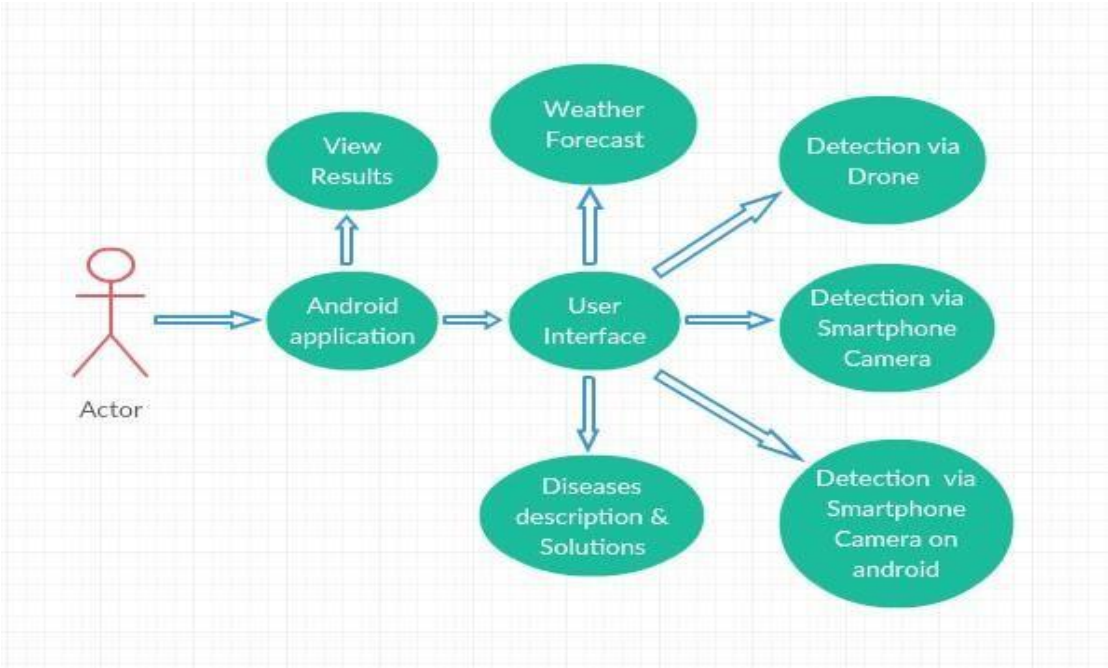


Figure 4.1 Main Application

4.5 Data Flow Diagram

A DFD shows the flow of data across the application. The information about process timing and whether it is background process or not, doesn't defined in DFD.

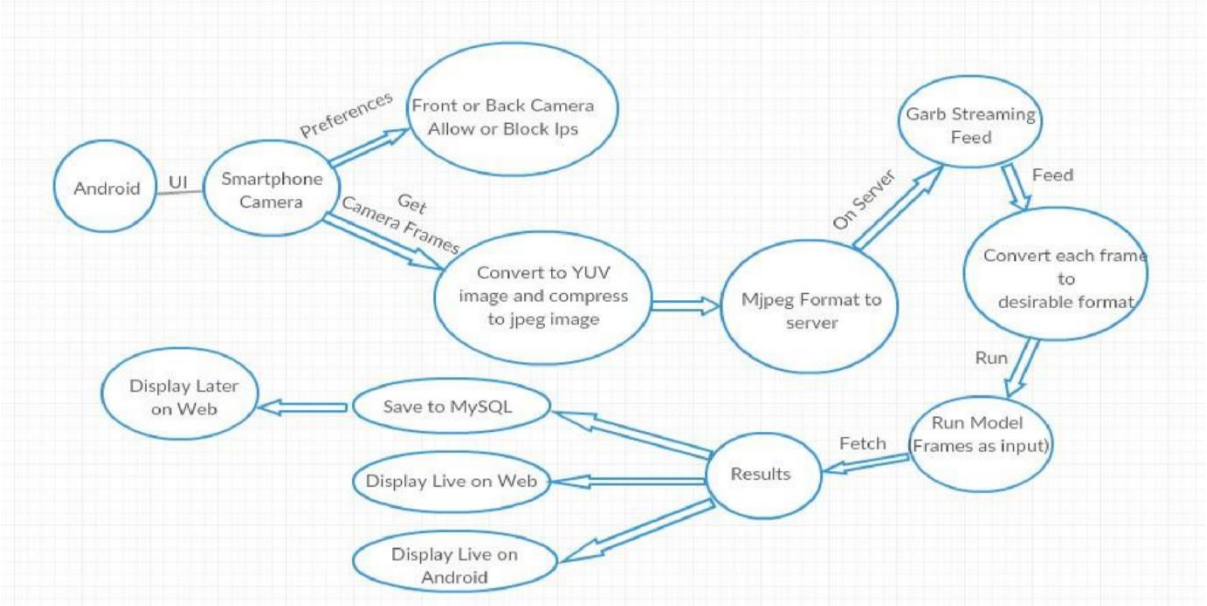


Figure 4.2 Stream from Smartphone Camera: Detection on Server

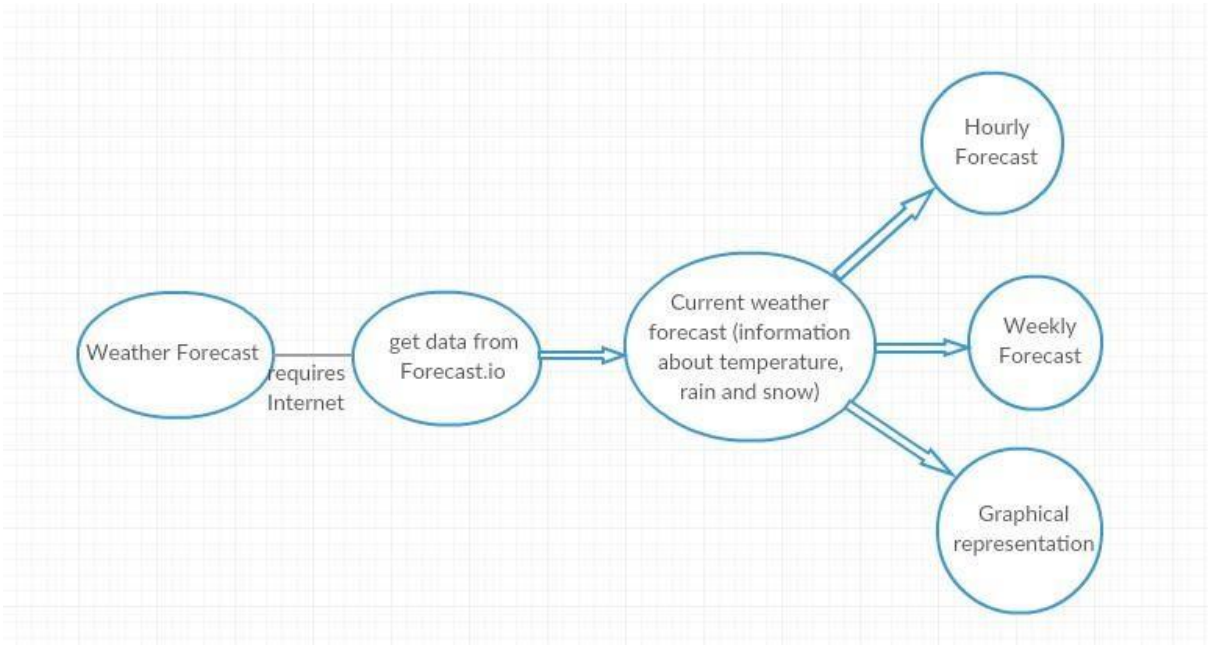


Figure 4.3 Weather forecast

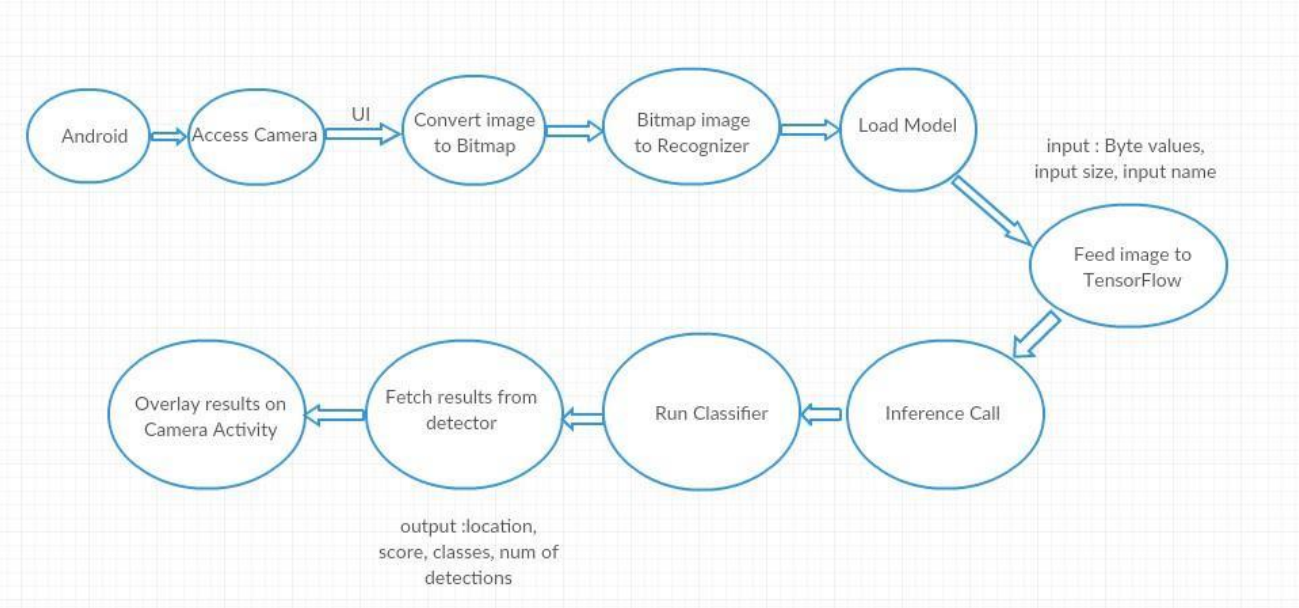


Figure 4.5 Stream from Smartphone Camera: Detection on Server

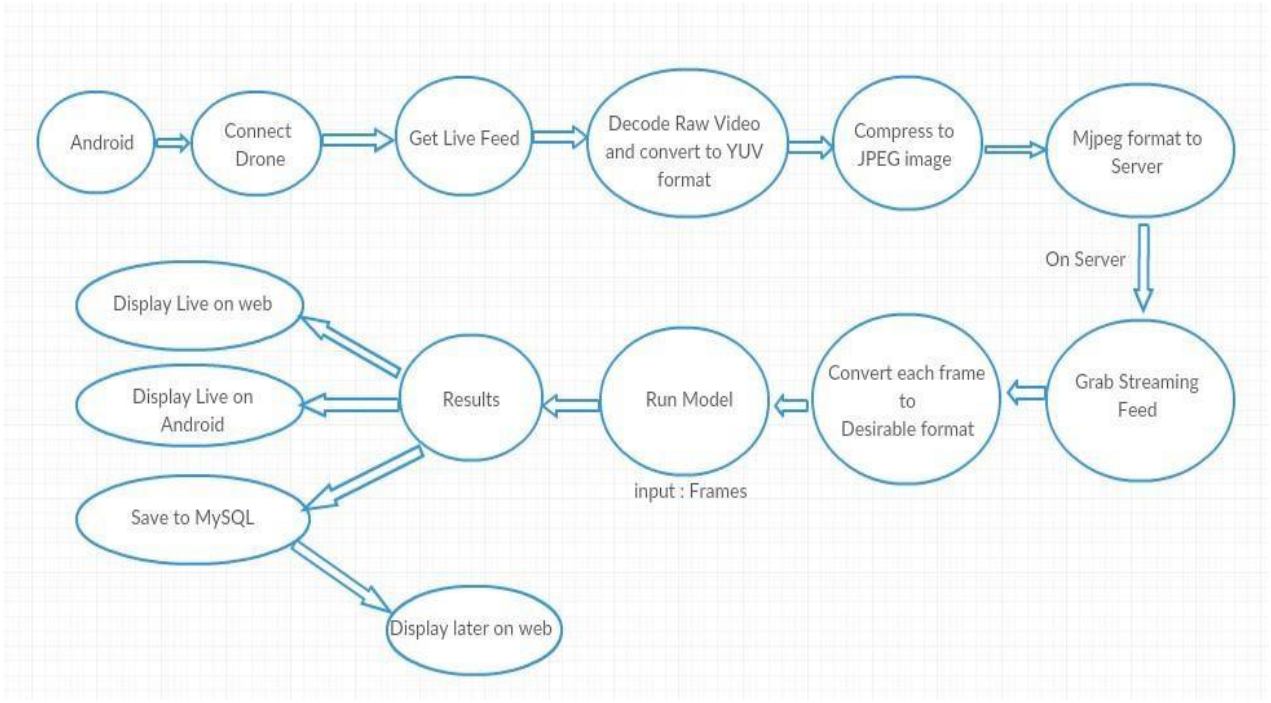


Figure 4.4 Stream from Drone Camera: Detection on Server

4.6 Activity Diagram

An activity diagram is similar to a flowchart or a dataflow diagram showing a series of actions or flow of control in a system. A description use case step can also be drawn. It can consist of sequential and concurrent activity models .

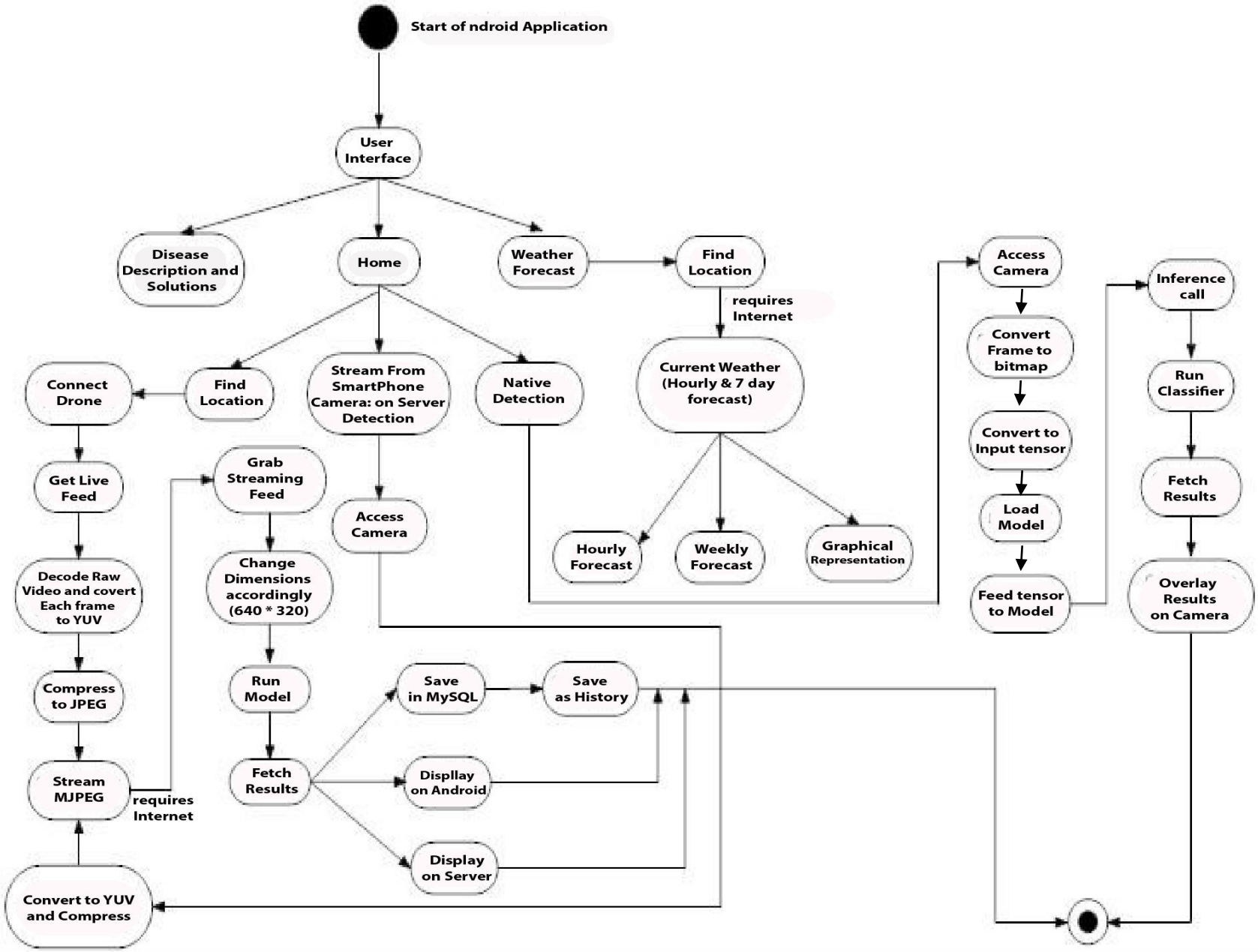


Figure 4.6 Android Activity Diagram

CHAPTER # 5

SYSTEM IMPLEMENTATION

5 SYSTEM IMPLEMENTATION

In this chapter, we'll focus on an implementation of "MedCrafter Agri-DRONE" application. Where user can perform activities on android application.

5.1 Introduction

This section discusses the phase in which the application is developed. Work to be done in the phase is much straightforward and stepwise due to the planning and designing that went into the project before this phase. The design requirements and specifications will be implemented in an executable program. Because designing has been completed, developers now have an idea of how the application should be implemented and what should be its look and feel. Now all that remains is for the developers to put together all the components and designs, so that the project can be easily implemented and understood.

5.2 Training

Object detection generally is a term used to label computer vision techniques for locating objects and labeling them. Object detection techniques can be applied both to static images and moving images.

Open-source libraries such as OpenCV's DNN library and TensorFlow Object detection API offers easy-to- use, open source frameworks where pre-trained models for object detection reach high accuracy in detecting various objects from humans to tv monitors. However, the video footage used in this project was shot with cameras of an android phone and a drone. Which means that in the footage taken from the drone camera was from a very high angle and the objects appear very small and partially occluded. This made it important that we train the model to work with specific data. For this project, we chose Tensor Flow object detection API due to its popularity in object detection sand for its easy-to-approach

In the project, TensorFlow object detection API was used and tested for detection of Tomato plants and two diseases **"Late Blight"** and **"Bacterial Spot"**. The model used was Faster RCNN model with Inceptionv2.

We already knew that

1. For working with given data a pre-trained model needs to be fine-tuned.
2. After fine tuning a per-trained model can and will work efficiently with given data.
3. Fine-tuned data can have some problems with drone footage as it cannot provide the detailed image of a plant at such distance.
4. Using a training data taken at different times will definitely improve the results with the new images.

5.2.1 Other models

Popular models in object detection API are Faster-RCNN, Multibox Single Shot Detector (SSD) and YOLO (You Only Look Once).

Originally **R-CNN** method worked by running a neural net classifier on samples cropped from images using externally computed box proposals. This approach is very computationally expensive due to many crops. **Fast RCNN** reduced the computation by doing the feature extraction only once to the whole image and using cropping on the lower layer i.e. feature extraction is done only once to the whole image and samples are cropped with externally computed box proposals. **Faster-RCNN** goes beyond and used extracted features to create class-agnostic box proposals i.e. no externally computed box proposals. R-FCN is like Faster RCNN, but the feature cropping is done on a separate layer to increase efficiency [2, 7, 10, 16].

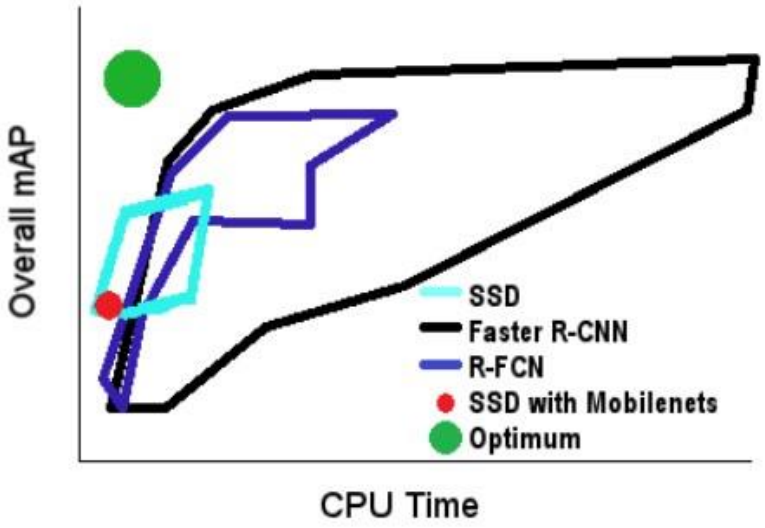


Figure 5.1 Comparison of TensorFlow models

YOLO runs a single convolutional network on the whole input image once to predict bounding box with confidence score for each class. This means that this model is faster than SSD and Fast RCNN. It learns only the general representation of the objects. This increases the localization loss.

SSD is different than F-RCNN and YOLO as it does not require a second stage pre-proposal classification operation.

As we can see that all these models have some tradeoffs. Some work faster and some are relatively slower to train and run. The models that run faster are mostly less accurate and vice versa. Such as Faster RCNN slightly more expensive to run but are more accurate.
The image below illustrates the accuracy of models with larger images (600x600) on different models.

5.2.2 An explanation of layers used

The feature extractor used with our model **Faster-RCNN** is **Inceptionv2.**

In the case of Inceptionv2, it has the following layers[7-9, 16-19].

Table 5.1 Inception v2 layers
https://arxiv.org/pdf/1512.00567v3.pdf

Type	Patch Size/Stride	Input Size
conv	3x3/2	299x299x3
conv	3x3/1	149x149x32
conv padded	3x3/1	147x147x32
Pool	3x3/2	147x147x64
conv	3x3/1	73x73x64
conv	3x3/2	71x71x80
conv	3x3/1	35x35x192
3xInception	As in Figure 5.2	35x35x288
5xInception	As in Figure 5.3	17x17x768
2xInception	As in Figure 5.4	8x8x1280
pool	8x8	1x1x2048
Linear	Logits	1x1x2048
softmax	classifier	1x1x1000

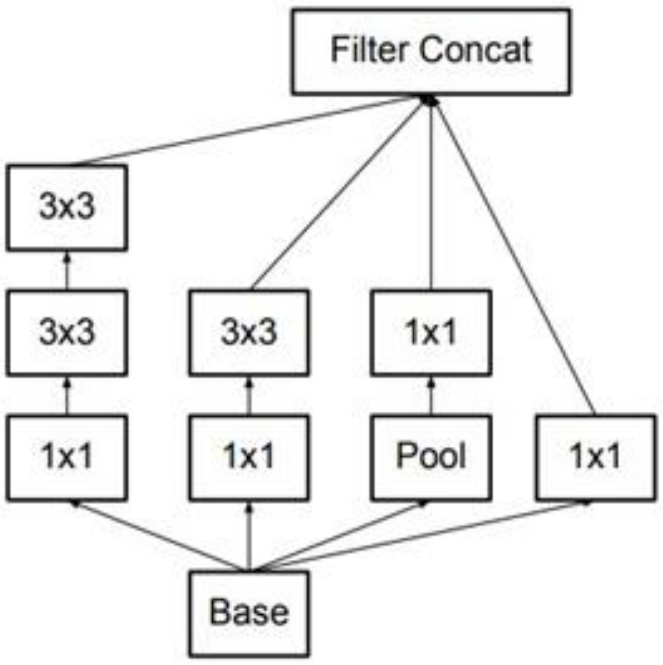


Figure 5.2 A 3xInception Layer
Inception modules where each 5 x 5 convolution is replaced by two 3 x 3 convolutions.
https://arxiv.org/pdf/1512.00567v3.pdf

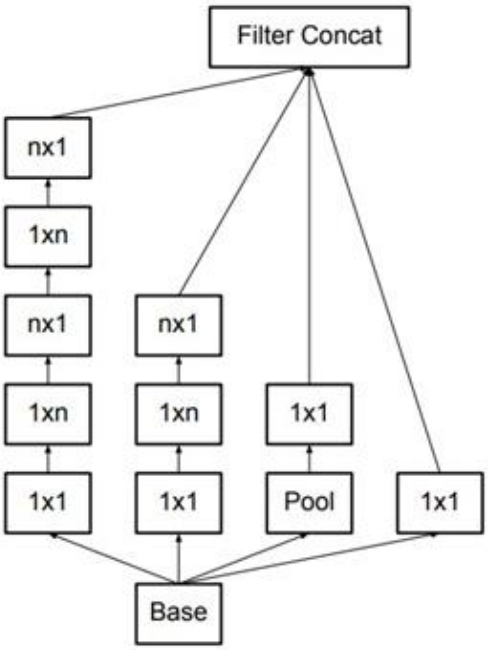


Figure 5.4 A 5xInception Layer
Inception modules after the factorization of the n x n convolutions.
https://arxiv.org/pdf/1512.00567v3.pdf

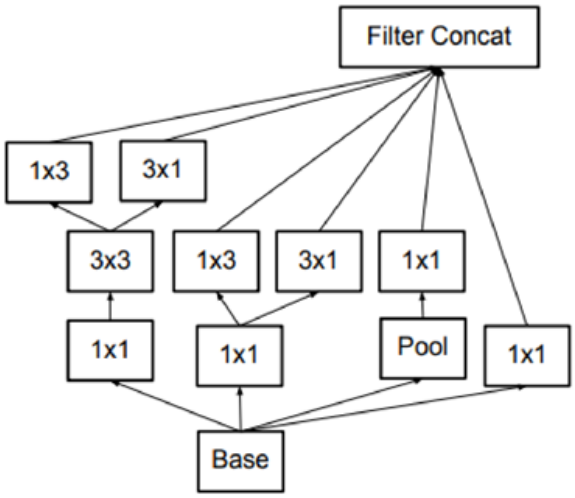


Figure 5.3 A 2xInception Layer
Inception modules with expanded filter bank outputs. This architecture is used in coarsest (8 x 8) grids to promote high dimensional representations.
https://arxiv.org/pdf/1512.00567v3.pdf

5.2.3 Training Method and Dataset

Raw data consisted of images from the android phone and some drone shots. In total, there were 168 images. From which 150 images were chosen to be used for training and 18 images were used as test images. All the images were labeled. The training data base was created by tagging the objects i.e. diseases and plants manually in the images using LabelImg as shown in Figure. LabelImg saves the annotations as PASCAL VOC format which is an xml-format. A readymade script was available for reading TFRecords (Tensor Flow Record Format). All the images were resized to a size of 320x180 to increase model training efficiency [7].

5.2.4 Object Detection API

The object detection API is a framework built on top of TensorFlow that helps to make it easy to 'develop, train and deploy object detection models'. To make this possible, the TensorFlow Object Detection API makes available to the user multiple pre-trained object detection models. In this work, we used Faster RCNN model with Inception v2. The faster_rcnn_inception_v2_coco-model reported to have mean Average Precision of 28% on COCO dataset [20].

The COCO dataset has 330k images out of which more than 200k images are labeled.

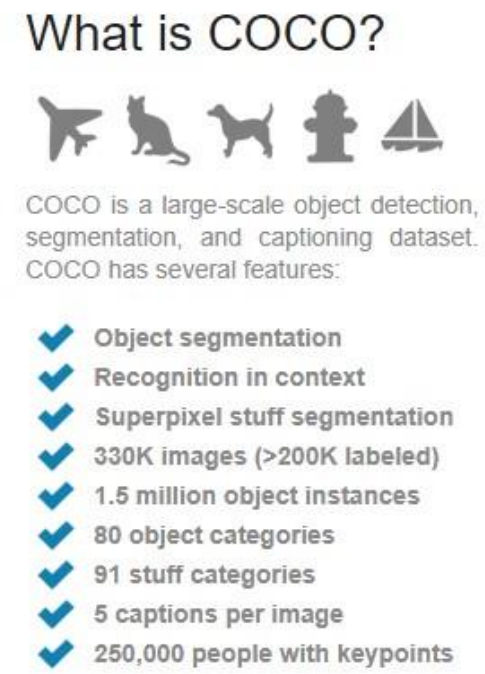


Figure 5.5 coco dataset
http://cocodataset.org/

The pre-trained F-RCNN model was fine-tuned for our dataset manually. A provided faster_rcnn_inception_v2_coco.config file was used as a basis for the model configuration. The provided checkpoint for faster_rcnn_inception_v2_coco was used as a starting point for the fine-tuning process. The training was stopped after 177.5k steps. During the process the loss function eventually evened out to a very small value after that many steps.

The training took over 72 hours on server with CPU with 8 cores. The total loss was greatly decreased over time from the pre-trained model.

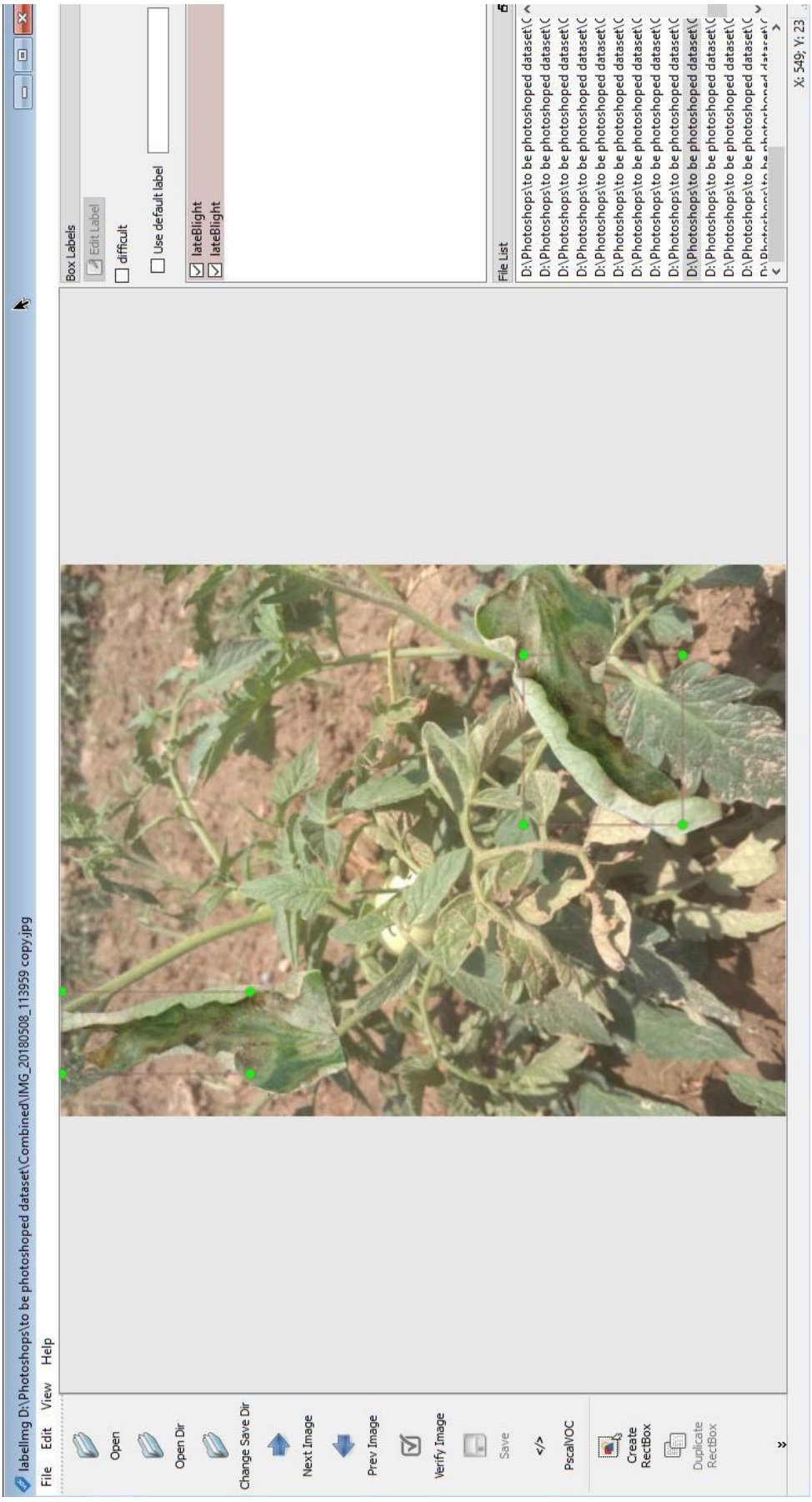


Figure 5.6 LabelImg

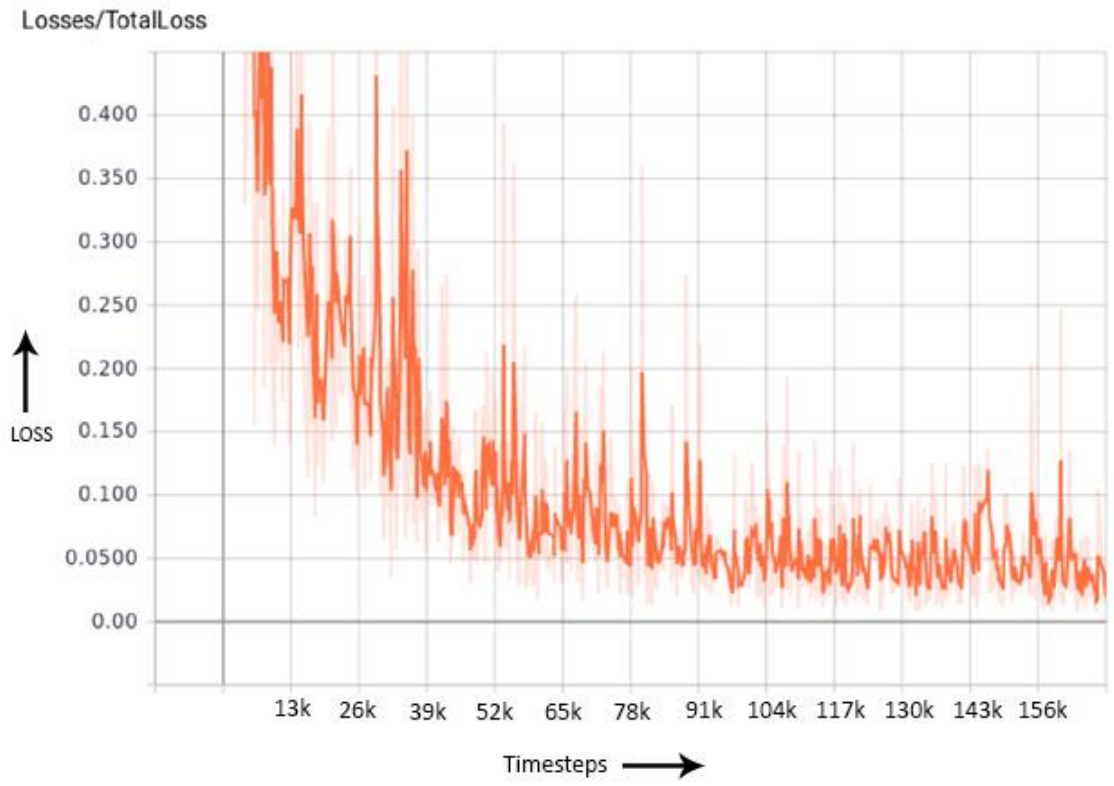


Figure 5.7 Loss progression during training

5.3 Web Application

5.3.1 Introduction

This section provides an overview of the working of the web app of MedCrafter project. Some the problems encountered during the development and deployment will also be discussed in this section.

5.3.2 Description

The web app aims to provide a platform for the users where they can log in, upload a picture or a video of the plants that they need to get diagnosed. The web app detects if there are any infected plants from the footage that is provided by the user. It also acts as a server for plant disease detection for live video being streamed directly from a MedCrafter drone. The application is a web-based tool that has been implemented using the following [21, 22]:

1. Python	**2.** JavaScript	**3.** Bootstrap
4. MySQL	**5.** JQuery	**6.** TensorFlow
7. Flask	**8.** OpenCV	**9.** TensorFlow js

5.3.3 Features of Web App

5.3.3.1 Registration

The user inputs their information, we verify that the information is correct, insert it into the database and return back with a success message. If the information is not valid, redirect to the registration page and show the following requirements:

5.3.3.2 Validations and Fields to Include

1. First Name - letters only, at least 2 characters and that it was submitted
2. Last Name - letters only, at least 2 characters and that it was submitted
3. Email - Valid Email format, and that it was submitted
4. Password - at least 8 characters, and that it was submitted
5. Password Confirmation - matches password

5.3.3.3 Login

When the user initially registers we would log them in automatically, but the process of "logging in" is simply just verifying that the email and password the user is providing matches up with one of the records that we have in our database table for users.

5.3.3.4 But how do we keep track of them once they've logged in?

It's using session! We can create a session variable that holds the user's id. From our study in Database Design, we know that if we have the id of any table we can gather the rest of the information that is associated with that id. Storing a single session variable with the user's id is all we need to access all the information associated with that user.

Once we have already identified the places on our site that we wish to be dynamic for users that are logged in, then we just need to check to see if that session variable has been set and display the content accordingly.

What’s happening!

1. A basic login and registration form
2. Redirect user to a Home page on successful login and register
3. Display error messages if either login or registration validations fail We used md5 to hash passwords before inserting them into the database

5.3.3.5 Static Disease Detection

The main functionality of the tool is the detection of a disease in

- An image of a plant.
- or a video of the plant

The website provides the user with an interface where user can upload a picture or video and the app detects disease in plant or plant leaf. Complete process of detection is done on the server. Once the detection script has run, the user is directed to a new page that displays the details about detected diseases in the plants and its remedy. Results are automatically stored in a mySQL database, so that the user can see the previous detections and history. The same procedure is followed with video feed, user can upload video and the results are displayed to the user.

5.3.3.6 Live Disease Detection

The app can live stream footage from a drone and run detection on the images received. The app will show the results to the user live on the video. It can also save the results in a database. So, the user can also see the results of the detection later[22].

5.3.3.7 History

The app provides a history tab to the user where they can see the results of the previous detections and see the progress of the crop at different time steps.

5.3.3.8 Validation

The app has authentication-based access. It allows only the registered users to access all the features. If the user has no account for the app, they can sign up for it in the signup section provided.

5.3.4 How does it work?

The web app is implemented in Python web programing environment called flask. It also implements MySQL database server for storing user credentials and diseases detection history. The website is styled using Bootstrap4 classes on top of HTML5. [21]

The detection of diseases is done by running the images and video frames against the trained detection models that have been trained on the dataset containing images from different sources of tomato plants that are healthy as well as infected with diseases.

```
with detection_graph.as_default():
  with tf.Session(graph=detection_graph) as sess:
    # Definite input and output Tensors for detection_graph
    image_tensor = detection_graph.get_tensor_by_name('image_tensor:0')
    # Each box represents a part of the image where a particular object was detected.
    detection_boxes = detection_graph.get_tensor_by_name('detection_boxes:0')
    # Each score represent how level of confidence for each of the objects.
    # Score is shown on the result image, together with the class label.
    detection_scores = detection_graph.get_tensor_by_name('detection_scores:0')
    detection_classes = detection_graph.get_tensor_by_name('detection_classes:0')
    num_detections = detection_graph.get_tensor_by_name('num_detections:0')
```

Figure 5.8 code initializing all tensors

```
def load_image_into_numpy_array(image):
  (im_width, im_height) = image.size
  return np.array(image.getdata()).reshape(
      (im_height, im_width, 3)).astype(np.uint8)
```

Figure 5.9 code converting an image to numpy array

The stored images and videos are processed using Faster-RCNN model which provides more accurate results but takes slightly more duration to detect the diseases. The live video feed is processed against MobileNet model which provides results quicker than Faster RCNN but is slightly less accurate than Faster-RCNN. The models have been trained using TensorFlow's Object detection API in Python.

In first step tensors are generated (a tensor is a multi-dimensional array). Each tensor generated has a different function. The tensors include ***(Figure 5.8)***

Input Tensor

1. **image_tensor** tensor to store image

Output Tensors

1. **detection_boxes** tensor to store area of image with a disease detected.
2. **detection_scores** detection score is the probability of a disease being in a detection box i.e >70% bacterialSpot means that "we can say with 70% accuracy that the disease detected in a detection box is bacterial spot".
3. **detection_classes** detection classes are all the disease that the app can detect.
4. **num_detections** number of detection per image or frame store in it.

The images are converted into numpy array ***(Figure 5.9)***, which means that an image is only a collection of different values of red, green and blue pixels. These values are stored in an array.

```
output = []
obj_above_thresh = np.asscalar(obj_above_thresh)
# Add some metadata to the output
item = Object()
item.version = "0.0.1"
item.numObjects = obj_above_thresh
item.threshold = threshold
output.append(item)

for c in range(0, len(classes)):
    class_name = category_index[classes[c]]['name']
    if scores[c] >= threshold:      # only return confidences equal or greater than the threshold
        print(" object %s - score: %s, coordinates: %s" % (class_name, scores[c], boxes[c]))
        item = Object()
        item.name = 'Object'
        item.class_name = class_name
        item.score = np.asscalar(scores[c])
        item.y = np.asscalar(boxes[c][0])
        item.x = np.asscalar(boxes[c][1])
        item.height = np.asscalar(boxes[c][2])
        item.width = np.asscalar(boxes[c][3])

        output.append(item)

outputJson = json.dumps([ob.__dict__ for ob in output])
```

Figure 5.10 classifying diseases from the numpy data

The numpy array is run against the model now and the values like detected disease classes, total number of classes detected, and their accuracies are stored in their respective variables after detection.

Now the app generates coordinates on the image where the disease was detected and stores the value in jsonObject. This jsonObject is then returned to client browser in form of XMLHTTP request. Browser takes this request and draw boxes and labels on the video frame by implementing an HTML5 canvas.
The same process happens when a video is uploaded, the individual frames are run against the model and results are obtained that get appended to the frames of the video on run time.

```
(boxes, scores, classes, num) = sess.run(
    [detection_boxes, detection_scores, detection_classes, num_detections],
    feed_dict={image_tensor: image_np_expanded})

classes = np.squeeze(classes).astype(np.int32)
scores = np.squeeze(scores)
boxes = np.squeeze(boxes)

obj_above_thresh = sum(n > threshold for n in scores)
print("detected %s objects in image above a %s score" % (obj_above_thresh, threshold))
```

Figure 5.11 drawing detection scores over the image

For the connection of the app with mySQL database, Flask module called Alchemy was used. For the user authentication process, Flask module called bCrypt is user that encrypts the passwords and stores them in the database. When the user logs in, bCrypt module decrypts and matches it with the database entry.

5.4 Android App

5.4.1 Introduction

The working and functions of android app for MedCrafter project are discussed here in this section.

5.4.2 Description

MedCrafter android app is a native android application that makes the process of authentication and diagnosis of diseases a matter of mere seconds with great ease and speed.

The app is created using the following resources:

1. Android Studio
2. ProtoBuf
3. ffmpeg parsing
4. VideoWriter library
5. DJI media codec
6. DJI SDK
7. TensorFlow Lite

5.4.3 Features of Android App

The features that the android application provides are as follow

5.4.4 Diseases Information

The app displays all the information about the diseases it can detect in its home screen. The app also shows what kind of prevention measures be applied to avoid the plants from catching these diseases. Also, the actions that can be taken of the plants get infected by these diseases.

5.4.4.1 Disease diagnosis on a live video from the mobile phone camera

The android app can detect diseases on video feed live from the phone camera and display the results to the user on the video in form of detection boxes with name of the detected diseases.

5.4.4.2 Disease recognition on a live video from the drone

The live video from the drone is streamed to the mobile phone and it is processed frame by frame. The result is displayed live on the video.

5.4.4.3 Disease recognition on a live video from the drone

The live video from the drone is streamed to the mobile phone and it is processed frame by frame. The result is displayed live on the video.

5.4.4.4 Disease recognition from an image in the gallery

The android app allows the user to pick an image from the gallery and run the detection locally. Then the results are displayed to the user.

5.4.4.5 Disease recognition from a server based on the internet or the android device locally

Android app has detection model stored locally that can make the results available to the user readily. The user can also request the online resources to detect the anomalies.

5.4.4.6 Weather forecast integration

A weather forecast activity has been added to the app so that the farmer can plan to spray the fields on a clear day

5.4.4.7 Live Log

The user can either wait for image to be processed or they can choose to see the log of steps that are being carried out in the background for the detection of diseases.

5.4.4.8 Bridge between drone and webapp

The Android app can also act as a bridge between the drone and the web app. This is achieved by using Java ServerSocket Library and DataOutputStream.

The individual frames received by the android app from the drone are compressed from YUV format to RGB using ffmpeg library. Then it is converted to a stream of jpegs with library called mjpeg. The final mjpeg result is put into DataOutputStream and passed to the network through ServerSocket.

5.4.4.9 History and previous detections

The app provides the user with information regarding previous detections.

5.4.5 How does it work

5.4.5.1 On Image

For detection on images, the app lets the user choose an image from the gallery. The image is converted to an array of pixels and the array is passed to the detection function. The tensors are created that will be updated as the function progresses. The detection boxes are generated by the detection function and then drawn over the image. Which is then displayed to the user.

5.4.5.2 On video

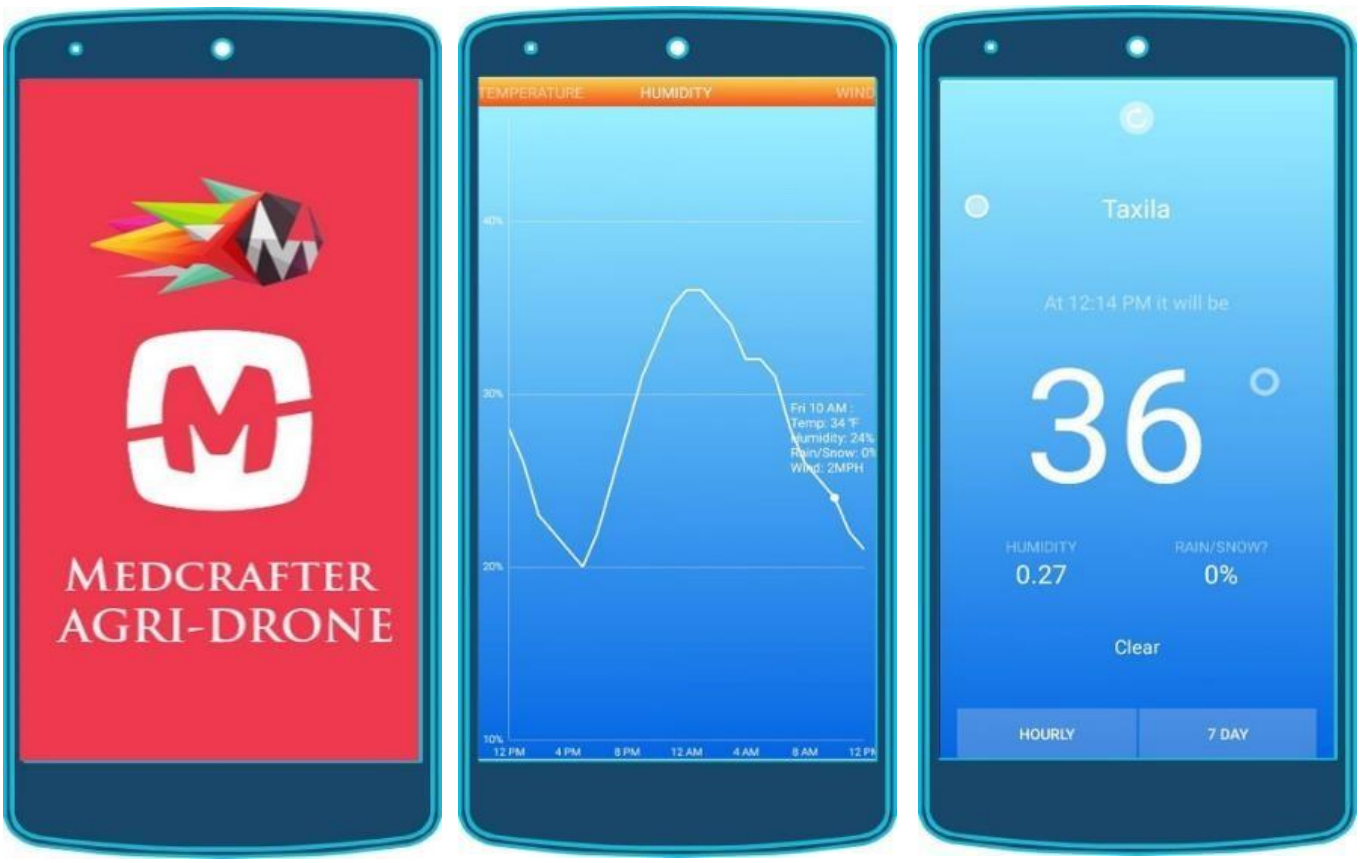
For a video to be processed and tested against the trained model. We first need to separate individual frames and convert them from YUV color space to RGB color space. In the next step tensors are generated that are filled with data after and during the detection of diseases by the model. After the detection process has been completed. The VideoWriter Library creates a canvas in the video and produces detection boxes which are then merged with the original video to produce a final video that displays the results of the detection as it plays.

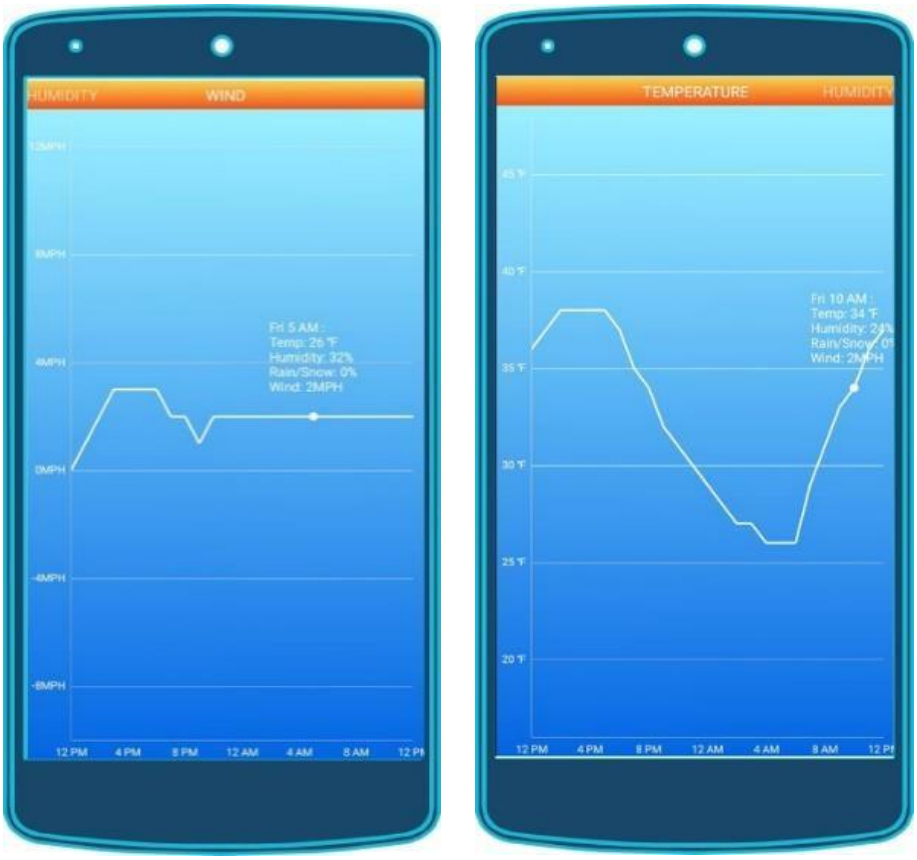
5.4.5.3 On Live feed

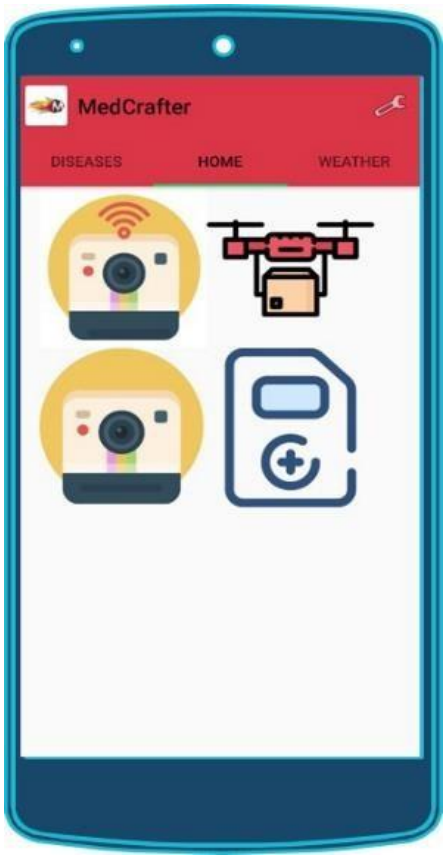
For live feed video same process is followed. The only obstacle is to live stream the video from the drone to the mobile phone. Which is achieved by following these steps [23]

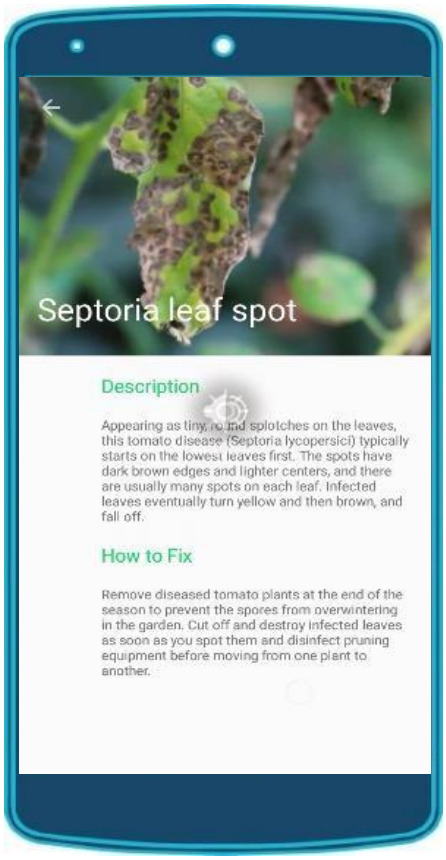
1. First, we initialize and set the instance as NativeDataListener to receive the frame data.
2. Then we send the data from camera to the ffmpeg library for parsing of individual frames.
3. Then we get the parsed data from ffmpeg and cache the data to a frameQueue
4. Now we initialize MediaCodec as decoder and then check whether there is any iframe in the MediaCodec. If not, then we get the default i-frame from the SDK resources and insert it at the head of the frame queue. Then dequeue the framed data from frameQueue and then feed it into the MediaCodec.
5. Get the output byte buffer from MediaCodec, if a surface is available for viewing the video and is configured by the MediaCodec, the output byte buffer only needs to be released. If not, the output YUV data should invoke the callback and pass it out to external listener, it should also be released.
6. Release the ffmpeg and the Media Codec, Stop the decoding thread.
7. The YUV data is then compressed into JPEG image and streamed to server in form of MJPEG (motion JPEG)

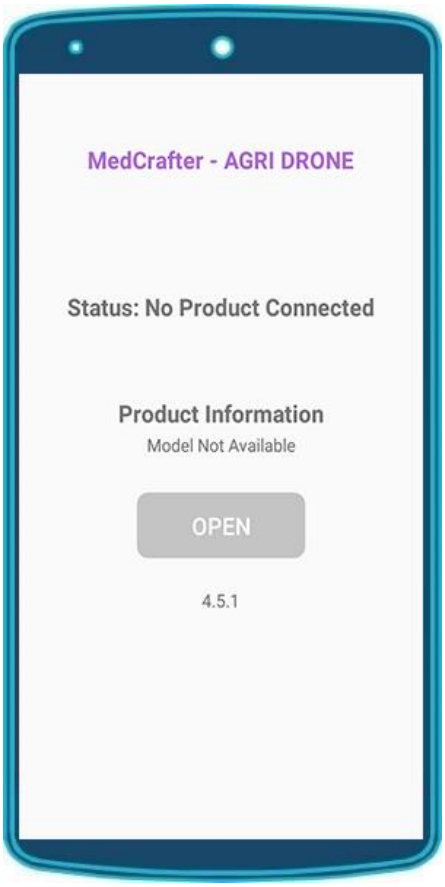
5.5 Screenshots of Android Application





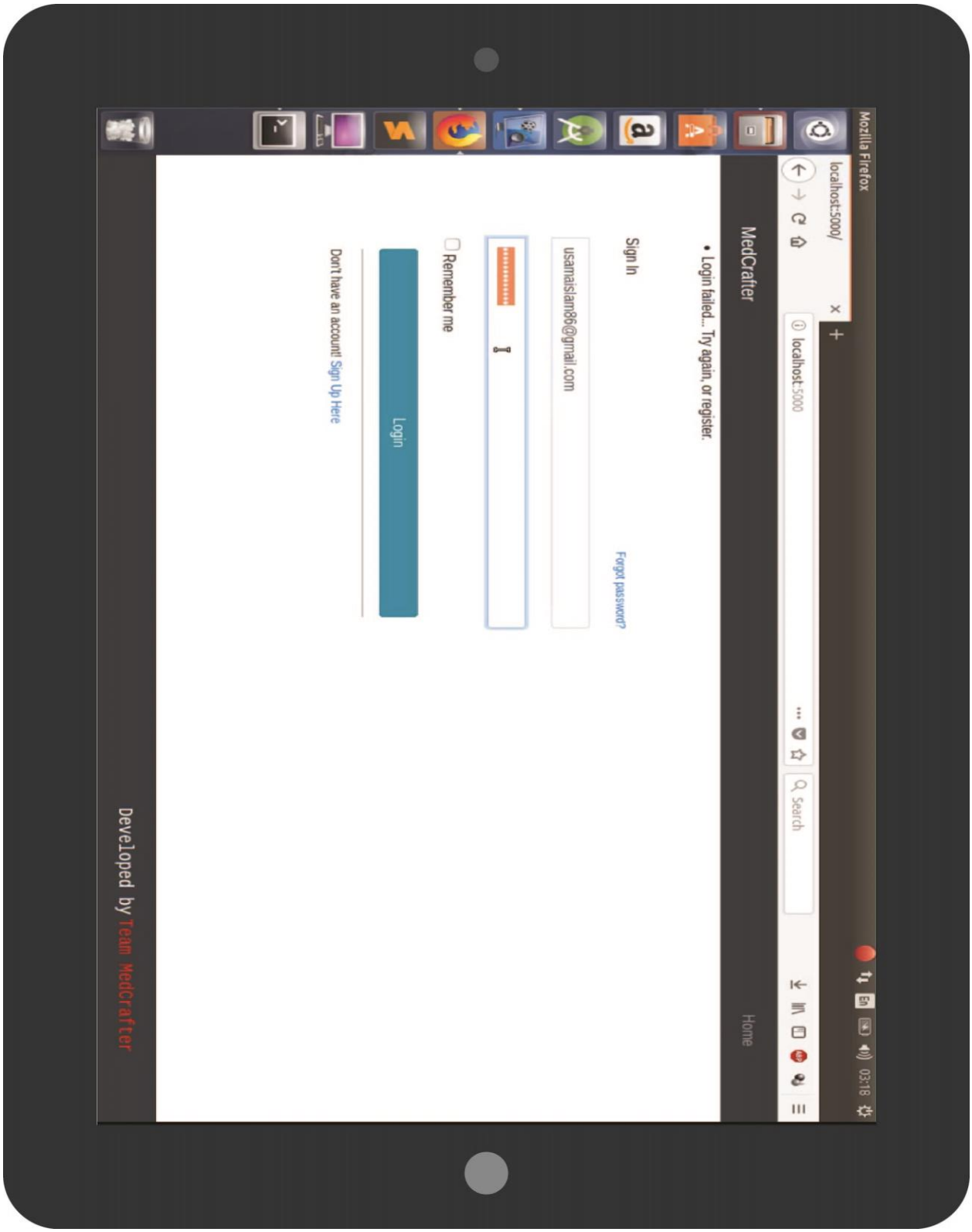

MedCrafter
DISEASES
HOME
WEATHER

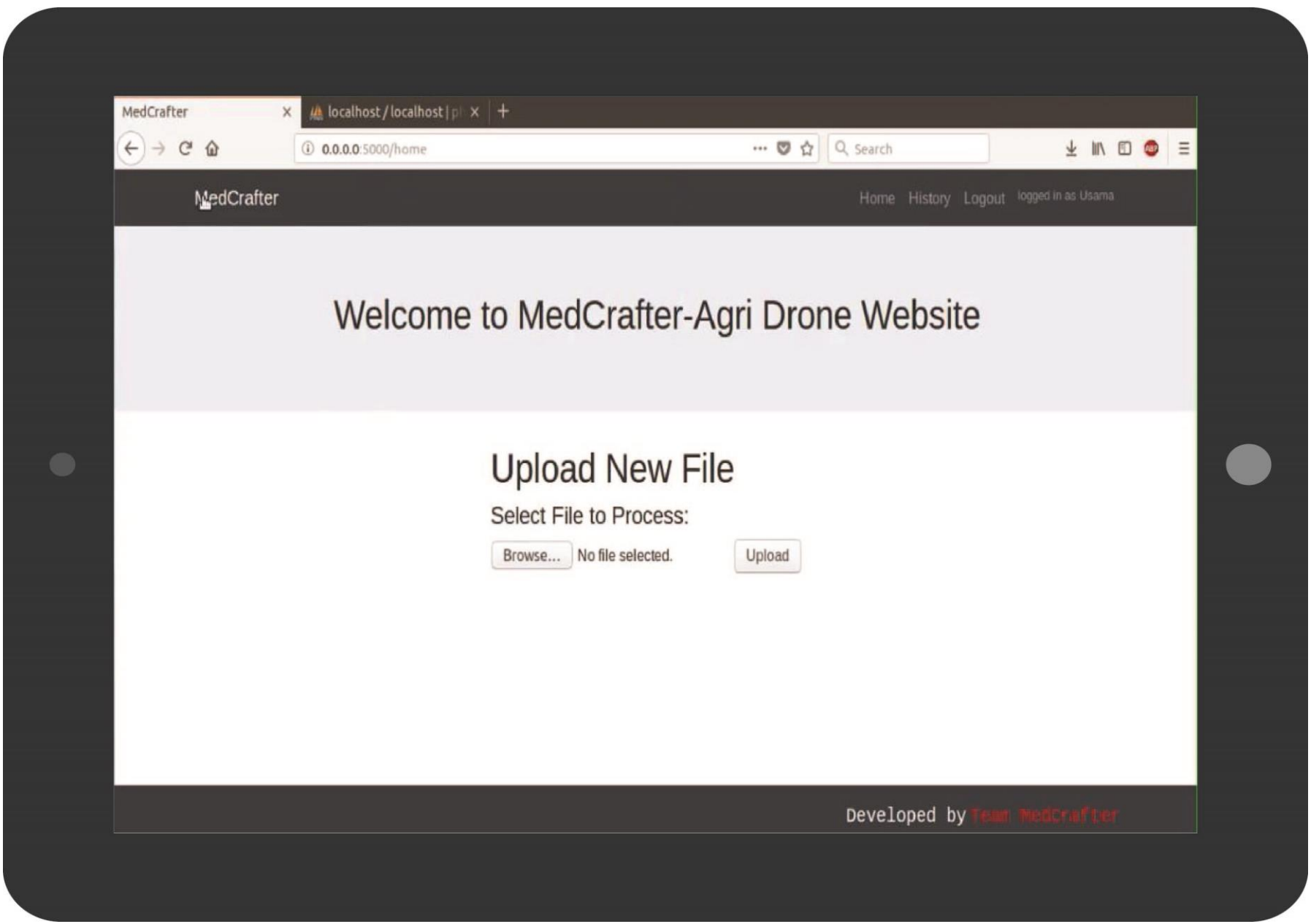

Septoria leaf spot
Description
Appearing as tiny, round splotches on the leaves, this tomato disease (Septoria lycopersici) typically starts on the lowest leaves first. The spots have dark brown edges and lighter centers, and there are usually many spots on each leaf. Infected leaves eventually turn yellow and then brown, and fall off.
How to Fix
Remove diseased tomato plants at the end of the season to prevent the spores from overwintering in the garden. Cut off and destroy infected leaves as soon as you spot them and disinfect pruning equipment before moving from one plant to another.


MedCrafter - AGRI DRONE
Status: No Product Connected
Product Information
Model Not Available
OPEN
4.5.1


lateBlight 1.00
lateBlight 1.00
lateBlight 1.00

5.6 Screenshots of Web App




MedCrafter
localhost / localhost | pl
0.0.0.0:5000/home
Search
MedCrafter
Home History Logout logged in as Usama
Welcome to MedCrafter-Agri Drone Website
Upload New File
Select File to Process:
Browse... No file selected.
Upload
Developed by Team MedCrafter

MedCrafter

Home History Logout logged in as Usama

Plant Doctor

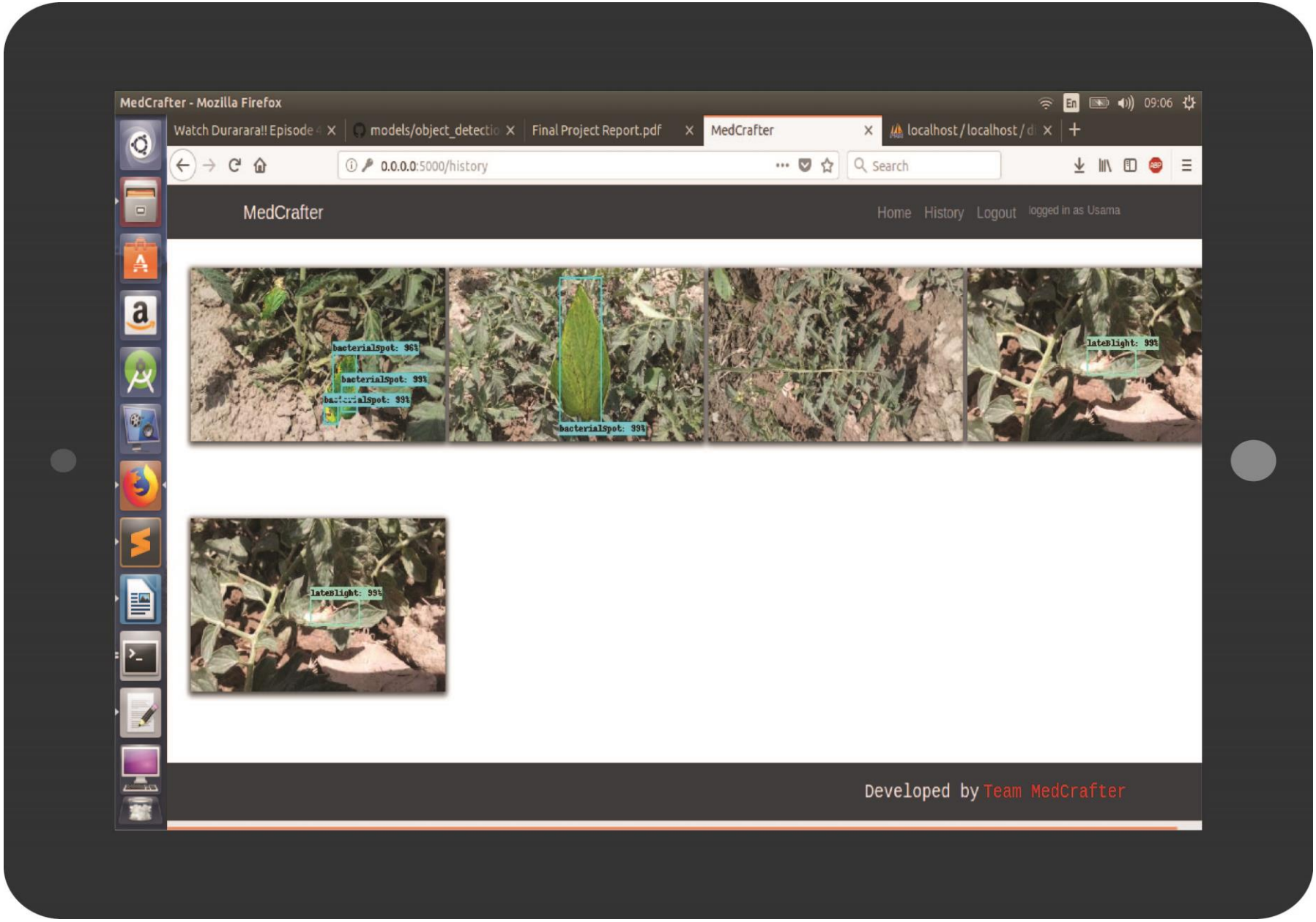
TensorFlow Output:

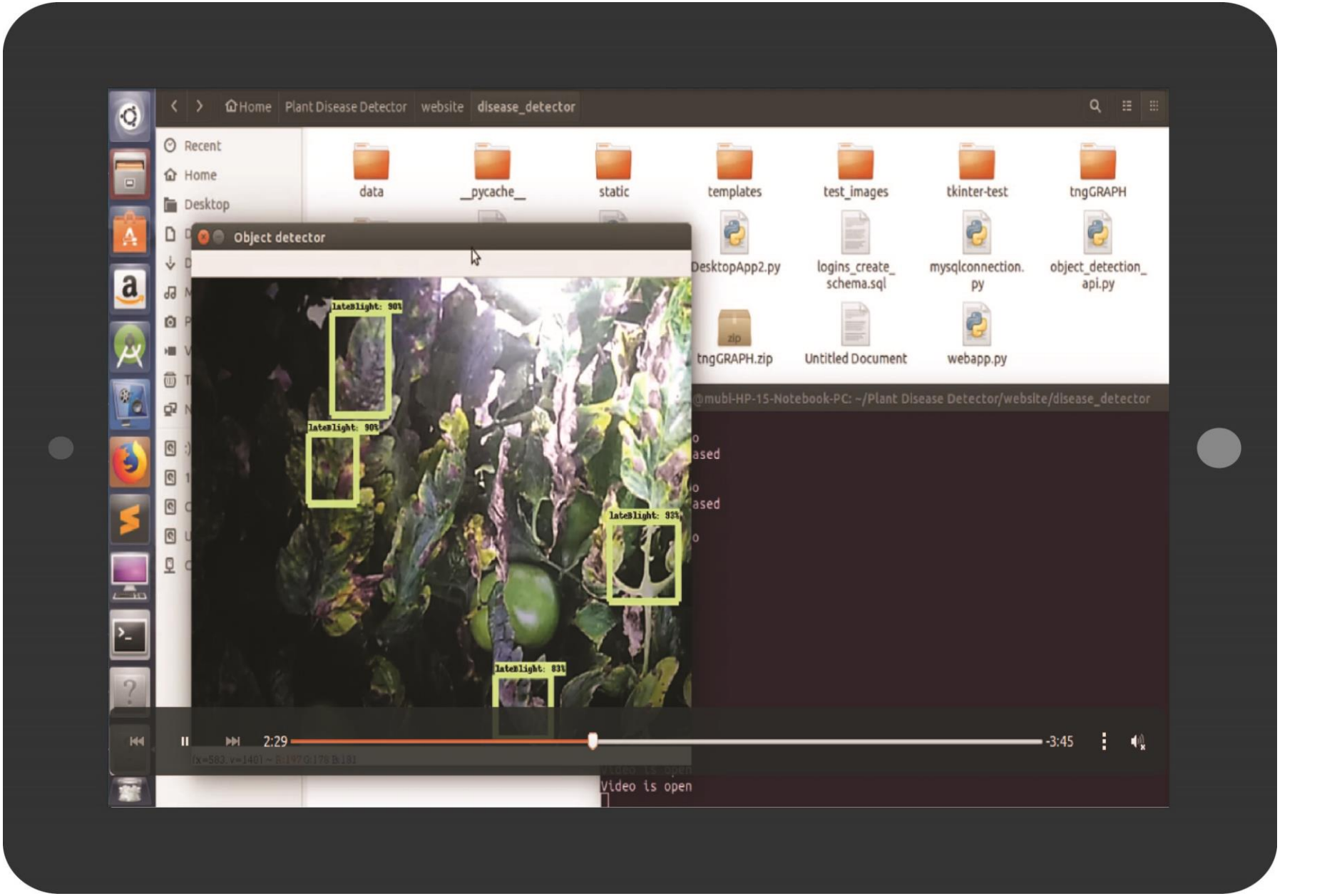
Unique Detected Disease

Detected Diseases with Scores > 50%

What should I do?

Developed by Team MedCrafter


MedCrafter - Mozilla Firefox
09:06
Watch Durarara!! Episode
models/object_detectio
Final Project Report.pdf
MedCrafter
localhost / localhost / d
0.0.0.0:5000/history
Search
MedCrafter
Home
History
Logout
logged in as Usama
bacterialSpot: 96%
bacterialSpot: 99%
bacterialSpot: 99%
bacterialSpot: 99%
lateBlight: 99%
lateBlight: 99%
Developed by Team MedCrafter


Home
Plant Disease Detector
website
disease_detector
Recent
Home
Desktop
data
__pycache__
static
templates
test_images
tkinter-test
tngGRAPH
DesktopApp2.py
logins_create_schema.sql
mysqlconnection.py
object_detection_api.py
tngGRAPH.zip
Untitled Document
webapp.py
Object detector
lateBlight: 90%
lateBlight: 90%
lateBlight: 93%
lateBlight: 83%
2:29
-3:45
@mubi-HP-15-Notebook-PC: ~/Plant Disease Detector/website/disease_detector
Video is open

CHAPTER # 6

SYSTEM TESTING

6 SYSTEM TESTING

In this chapter, we will discuss the testing phase of developed application **"MedCrafter AGRI-DRONE"** in different manner to know that how much efficient and effective application is.

6.1 Introduction

System testing is a procedure of executing an application or program with the expectation of discovering mistakes and whether the application is satisfying client needs. It can also be described as the capacity of a program in meeting the required or wanted outcomes.

Generally in the field of software engineering, a separate phase is carried out which is called testing phase it is done after the implementation phase has been completed. This approach has a benefit that it is difficult for an individual to look for his mistakes, but a different perspective and fresh eye can readily find the observable errors much quicker than the individual that has written or read the said material several times.

6.2 Testing Plan

A process of performing as application or program with the intention of finding errors and whether the application is fulfilling user needs.

6.2.1 Unit Testing

The software units in an application are modules and subroutines that are incorporated and integrated to play a particular function. Unit testing first of all focuses around modules, irrespective of one another's functionalities, to find errors. This enables the developers to recognize errors in the code and the logic within each individual module. This testing includes performing practically. All the feature controls are tested to make sure that each performs its action as required.

Commonly used method is White-Box Testing method. Every time a component of the program is changed, it can be run for testing that is the biggest and famous benefit of this testing phase. Issues that are arises during this phase, allowing to be resolved as quickly as possible. Unit testing is familiar by software developers. It allows them to test their application units before moving them to testers for formal testing.

6.2.2 System Testing

To test the complete "MedCrafter AGRI-DRONE", system testing has been used. It is beneficial to check whether the application meets its requirements and fulfill Quality Standards.

6.2.3 Integration Testing

Integration testing allows the software developers to integrate all the components of the "MedCrafter AGRI-DRONE" within one program and then test them in a group. Basically, this testing level is used

to catch the defects in the user interface between the functions/ modules. It is useful to determine how logically and efficiently all the units/ components are running together.

Here the android application, Web Application and dji UAV drone are integrated and tested. This testing helps to make sure that the application is integrated efficiently and is a functional unit that has a smooth transition of run time working.

6.2.4 User Acceptance Testing

User acceptance of "MedCrafter AGRI-DRONE" is the key factor for the success of our application. The application under consideration has been tested for user compliance by frequently keeping in touch with the application users during developing period and making required changes.

6.3 Test Cases

Table 6.1 Objectives -Test Case

Test Cases	Objectives
1	To make sure that application is easy to understand and is in working order.
2	To make sure that the user can work on application with ease.
3	To make sure that user can view data correctly.
4	To make sure that Detection of Disease is done on run time on Android App.
5	To make sure that drone is streaming video to server
6	To make sure that all the labels are according to disease
7	To make sure that server is picking the video and showing detection
8	To make sure that all modules are working properly.
9	To make sure that the application run at cross android versions successfully.
10	To make sure that android camera is streaming the video to server

6.4 Testing Results

Table 6.2 Testing results

CRITERIA	Test Status	REMARKS
All the graphical user interface options display successfully.	Test successful	OK
Disease Detection is working on run time.	Test successful	OK
Detections are working properly according to their targeted images.	Test successful	OK
Label showing exact data.	Test successful	OK
Drone is streaming video to server	Test successful	OK
Android Camera is streaming video to server	Test successful	OK

CHAPTER # 7

CONCLUSION
&
FUTURE WORK

7 CONCLUSION & FUTURE WORK

In this chapter, we will discuss the results and discussions of this framework "**MedCrafter AGRI-DRONE**" with conclude remarks and will also discuss related future work of this application.

7.1 Conclusion

"MedCrafter AGRI-DRONE" is developed for multiple DJI versions. Application and product are specially designed for farmers, agricultural industries, agricultural consultants and Government Agencies & Departments. The main objective of this application is to provide the facility of quick disease detection with modern techniques with the development of a practical, reliable and inexpensive real time application that can be used in fields. It is a market-oriented product for Plant Disease Detection, a smartphone app compatible with both smartphone camera and drone camera. User just needs to install android version of it once and then use it one click away. It will improve the performance agricultural crops.

7.2 Future Work

In next our first preference is to enhance this application by providing more new features that are as follows:

1. Synchronization of Android and Web application.
2. Integration of Raspberry PI.
3. Launching across customized drones (no DJI drone involved).

8 REFERENCES

1. Rampasek, L. and A. Goldenberg, *TensorFlow: Biology's Gateway to Deep Learning?* Cell Syst, 2016. **2**(1): p. 12-4.
2. Alzantot, M., et al., *RSTensorFlow: GPU Enabled TensorFlow for Deep Learning on Commodity Android Devices.* MobiSys, 2017. **2017**: p. 7-12.
3. Lite, T., *Android to launch tensorflow lite for mobile machine learning*. 2017.
4. Mulfari, D., A. Palla, and L. Fanucci, *Embedded Systems and TensorFlow Frameworks as Assistive Technology Solutions.* Stud Health Technol Inform, 2017. **242**: p. 396-400.
5. Tran, D., *How to train your own Object Detector with TensorFlow's Object Detector API.* URI: https://medium. com/towards-data-science/how-to-train-yourown-object-detector-with-TensorFlows-object-detector-api-bec72ecfe1d9. Cited November, 2017. **5**: p. 2017.
6. Wongsuphasawat, K., et al., *Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow.* IEEE Trans Vis Comput Graph, 2018. **24**(1): p. 1-12.
7. Fuentes, A., et al., *A Robust Deep-Learning-Based Detector for Real-Time Tomato Plant Diseases and Pests Recognition.* Sensors (Basel), 2017. **17**(9).
8. Bradski, G. and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. 2008: " O'Reilly Media, Inc.".
9. Huang, J., et al. *Speed/accuracy trade-offs for modern convolutional object detectors*. in *IEEE CVPR*. 2017.
10. Ren, S., et al. *Faster r-cnn: Towards real-time object detection with region proposal networks*. in *Advances in neural information processing systems*. 2015.
11. Lu, J., et al., *Detection of multi-tomato leaf diseases (late blight, target and bacterial spots) in different stages by using a spectral-based sensor.* Sci Rep, 2018. **8**(1): p. 2793.
12. Schaefer, S.C., et al., *Enhanced resistance to early blight in transgenic tomato lines expressing heterologous plant defense genes.* Planta, 2005. **222**(5): p. 858-66.
13. Xie, C., et al., *Detection of early blight and late blight diseases on tomato leaves using hyperspectral imaging.* Sci Rep, 2015. **5**: p. 16564.
14. Behmann, J., et al., *Specim IQ: Evaluation of a New, Miniaturized Handheld Hyperspectral Camera and Its Application for Plant Phenotyping and Disease Detection.* Sensors (Basel), 2018. **18**(2).
15. Bergenti, F., M.-P. Gleizes, and F. Zambonelli, *Methodologies and software engineering for agent systems: the agent-oriented software engineering handbook*. Vol. 11. 2006: Springer Science & Business Media.
16. Howard, A.G., et al., *Mobilenets: Efficient convolutional neural networks for mobile vision applications.* arXiv preprint arXiv:1704.04861, 2017.
17. Hoo-Chang, S., et al., *Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning.* IEEE transactions on medical imaging, 2016. **35**(5): p. 1285.
18. Krizhevsky, A., I. Sutskever, and G.E. Hinton. *Imagenet classification with deep convolutional neural networks*. in *Advances in neural information processing systems*. 2012.
19. Szegedy, C., et al. *Rethinking the inception architecture for computer vision*. in *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

20. http://cocodataset.org/
21. Greenberg, M., *Development of web applications using Flask in Python.* Moscow: DMK, 2014.
22. Grinberg, M., *Designing a RESTful API with Python and Flask*. 2013.
23. https://developer.dji.com/mobile-sdk/documentation/sample-code/index.html