CROP DISEASE DETECTION USING CNN

A Project Report Submitted in Partial Fulfilment of Requirement

for the award of the degree of

Bachelor of Technology

In

COMPUTER SCIENCE ENGINEERING

SESSION 2019-2020

Submitted by

Aman Neekhara (1604310007)

Utsav Kumar Datta (1604310053)

Arjun Gaud(1604310012)

Vishvdeep Gautam(1604310057)

Under the Supervision of

Ass. Prof. S.K Gupta

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

BUNDELKHAND INSTITUTE OF ENGINEERING AND TECHNOLOGY, JHANSI 284128

BUNDELKHAND INSTITUTE OF ENGINEERING AND TECHNOLOGY

Department of Information Technology

CERTIFICATE

This is to certify that the project entitled "**CROP DISEASE DETECTION USING CNN"** submitted by **Aman Neekhara (1604310007), Utsav Kumar Datta (1604310053), Arjun Gaud (1604310012)** and **Vishvdeep Gautam (1604310057)** in the partial fulfilment of the requirement for the award of Bachelor of Technology degree in Information Technology at **Bundelkhand Institute of Engineering and Technology**, **Jhansi** is an authentic work carried out by them under my supervision and guidance.

To the best of my knowledge, the matter embodied in the project has not been submitted to any other University/Institute for the award of any Degree or Diploma.

Project Guide:

……..

Ass. Prof. S.K. Gupta

Computer Science Engineering

Date:

Place:

BUNDELKHAND INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(Affiliated to Dr. A.P.J. Abdul Kalam Technical university, Lucknow, UP)

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published and written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in this text.

Submitted By: **Date:**

Aman Neekhara (1604310007)
Utsav kumar Datta (1604310053)
Arjun Gaud (1604310012)
Visvdeep Gautam (1604310057)

BUNDELKHAND INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(Affiliated to Dr. A.P.J. Abdul Kalam Technical university, Lucknow, UP)

ACKNOWLEDGEMENT

I would like to express my special thanks of gratitude to my guide "Ass. Prof. S.k. Gupta Sir" for there able guidance and support in completing my project.
I would also like to extend my gratitude to our Honorable Director "Prof. V.K. Tyagi Sir" for providing me with all facility that was required.

Group Members:

Aman Neekhara……..(1604310007)

Utsav Kumar Datta …….(1604310053)

Arjun Gaud……..(1604310012)

Vishvdeep Gautam……...……..(1604310057)

Table of Contents

LIST OF FIGURES

ABSTRACT

Plant disease causes many significant damages and losses in crops around the world. Appropriate measures on disease identification should be introduced to prevent the problem and minimise the losses. Technical approaches using machine learning and computer vision are actively researched to achieve intelligence farming by early detection on plant disease. An mobile application is obviously desirable to aid the farmers or garden enthusiasts in diagnosing what sorts of diseases a plant has. Although some similar applications exist, most of them achieve the function by submitting the image to a team of plant pathologists or expert garden advisors to get possible identification results and some advice. This dissertation presents the research, design and implementation of a mobile application which can automatically identify the plant's diseases based on its leaf appearance with some computer vision and machine learning technique. Many experiments and evaluations on different segmentations, feature extractions and classification methods were done to find the most effective approach. The target group of the user is those who request a free and quick diagnosis from a picture of a leaf at any time of the day.

1. INTRODUCTION

India is an agricultural country. Today, India ranks second worldwide in farm output. Nowadays, a new concept of smart farming has been introduced where the field conditions are controlled and monitored using the self operating systems. The self recognition of the disease is based on the identification of the symptoms of disease. So that information about the disease occurrence could be quickly and accurately provided to the farmers, experts and researchers. This in turn reduces the monitoring of large fields by human beings. In disease recognition from image the key is to extract the characteristic feature of the diseased region. According to the disease the features may vary. The features that are extracted from the image are color , shape, texture etc. Sometimes for detection of the disease more features are extracted and these extracted features would increase the hardware as well as software cost. This further causes increase in the complexity and the computation time. Hence it is necessary to reduce the feature data.

The occurrence of the disease on the plant may result in significant loss in both quality as well as the quantity of agricultural product. This can produce the negative impact on the countries whose economies are primarily dependent on the agriculture. Hence the detection of the disease in the earlier stages is very important to avoid the loss in terms of quality, quantity and finance. Usually the methods that are adopted for monitoring and management of plant leaf disease are manual. One such major approach is naked eye observation. But the requirement of this method is continuous monitoring of the field by a person having superior knowledge about the plants and its corresponding diseases. Moreover, appointing such a person would prove costly. Another approach is seeking advice from the expert which may add the cost. Also, the expert must be available in time otherwise it may result in loss. Diagnosis of disease on plants can also be done in laboratory testing. But this method requires satisfactory laboratory conditions along with professional knowledge. The pathogen detection methods can provide more accurate results. As the tests are carried out off the field the cost may be high and could be time consuming.

This software suggests a system which can provide more accurate results related to the identification and classification of disease and the best solution of that disease. It tries to replace the need of the experts to a certain extent.
Here, the captured image is first preprocessed and then converted to HSI color space format by using segmentation. The features such as major axis, minor axis, eccentricity are extracted from the image. In the last step, these features are given to the classifier to classify the disease occurred on the leaf.All the above processing is done by inception V3.

1.1 PROBLEM STATEMENT

The main purpose is to detect the diseased part of the plant. Using MATLAB convolutional neural networks are implemented in order to classify the diseased part. Aim is to detect the diseased part by finding the optimum way with minimum cost. In this problem we have considered fundamental five categories of the plant leaf disease which are Alternaria Alternata, Anthracnose, Bacterial Blight, Cercospora leaf spot and Healthy Leaves. All of these disease belongs to fungal, viral or bacterial type of the diseases. In our proposed solution we identify the percentage of the affected area and identify the disease. Our approach provide the result in minimum time span with maximum precision and accuracy in comparison to other existing approaches.

1.2 PROJECT OVERVIEW/SPECIFICATIONS

1.2.1 OBJECTIVE

- To help farmers by identifying the disease present in their plants leafs and to provide the best possible solution for that disease.
- To enhance the productivity of farming and to develop the interest towards farming.
- To reduce the effort of farmer in terms of time and money .
- To apply and implement the new algorithm to monitor the growth of the crop and detect the diseased part in the plant.
- To incorporate the global optimization strategy for enhancing the optimal solution.

- To modify the existing solution more efficiently to solve this problem is a more genuine way.

1.2.2 GOAL

The main goal of this software is to identify the disease present in plant leaf with best possible accuracy. This software will also give the solution to the disease which is present in the plant leaf. It will reduce the effort of farmers and will increase the productivity of the farmers.

1.2.3 FUNCTIONALITY

This project Farming Assistant will make farmers smart. It will assist farmers by taking care of every type of crops with the help of **new coming technologies like transfer learning and image processing**, it will detect if any disease is spreading in the crops along with this it will also tell the farmers that what treatment should be given to protect the crops.
Farming is not strictly about the primary production, but plays a major role in improving the efficiency of the entire supply chain and alleviating food security concerns.
It will manage the crop failure risk and boost feed efficiency in livestock production.
This software will take pictures of diseased leaf as input which is sent by farmers for the disease identification.

Fig 1.1: Input to software.

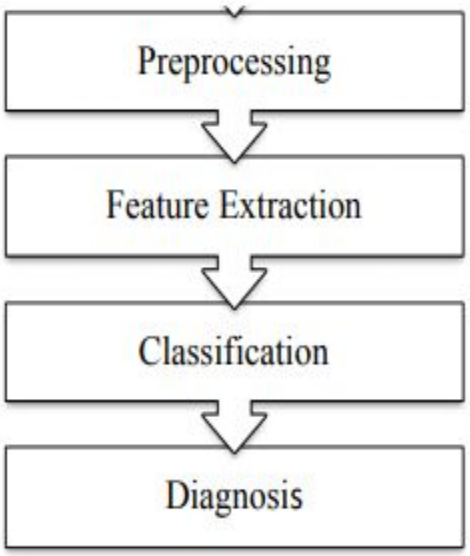

Fig 1.2 : Input flow processing

1.3 HARDWARE SPECIFICATION

- **Desktop :**
    1. **Processor :** Intel i5 (2.5-3.1GHz)
    2. **RAM** : 4 GB RAM
    3. **Storage :** 1TB HardDisk
- **Mobile :**
    1. **Ram**: 2GB,
    2. **API version:** 19.0 (Android Version 4.4)
    3. **Camera:** Yes

1.4 SOFTWARE SPECIFICATION

- **Desktop :**
    1. IDE: Android Studio (library required = tensor flow)
    2. Python IDE: Anaconda or any other web IDE

- **Mobile :**
    1. **API Version**: 19 (Android 4.4)
    2. **Minimum Ram Required**: 2GB
    3. **Camera**: Yes

2. LITERATURE REVIEW

Various techniques of image processing and pattern recognition have been developed for detection of diseases occurring on plant leaves, stems, lesions etc. by the researchers. The sooner disease appears on the leaf it should be detected, identified and corresponding measures should be taken to avoid loss. Hence a fast, accurate and less expensive system should be developed. The researchers have adopted various methods for detection and identification of disease accurately. One such system uses a thresholding and back propagation network. Input is a grape leaf image on which thresholding is performed to mask green pixels. Using K-means clustering segmented disease portion is obtained. Then ANN is used for classification [1].The other method uses PCA and ANN.PCA is used to reduce the dimensions of the feature data. to reduce the no. of neurons in the input layer and to increase speed of NN[2].Sometimes threshold cannot be fixed and objects in the spot image cannot be located. Hence authors proposed LTSRG-algorithm for segmentation of image [3]. In cucumber leaf disease diagnosis, spectrum based algorithms are used [4].In the classification of rubber tree disease a device called spectrometer is used that measures the light intensity in the electromagnetic spectrum. For the analysis SPSS is used [5].In citrus canker disease detection uses three level system. Global descriptor detects diseased lesion. To identify disease from similar disease based regions zone based local descriptor is used In last stage two level hierarchical detection structure identifies canker lesion [6]. For identification of disease on plant and stems first segmentation is carried using K-means clustering. Feature extraction is done by CCM method. Identification is done by using BPNN[7].With relevance to grapes, the fruit mostly suffer with tree types of diseases viz Powdery Mildew, Downy Mildew and Anthracnose. The two diseases are considered Powdery Mildew and Downy Mildew.

Powdery Mildew: Powdery mildew can infect all green parts of the grapevine. This disease is most easily recognized by the dusty appearance or white powdery growth occurring in patches on fruit or leaves. The white patches of powdery mildew produce millions of spores (conidia) which are spread by wind to cause more infections. Free moisture is not needed for secondary infection; temperature is the most important environmental factor. The image of grape leaf affected with late blight is shown below in Figure.

Fig (2.1) Diseased leaf

Downy Mildew: Early in the season, infected leaves develop yellowish-green lesions on their upper surfaces. As lesions expand, the affected areas turn brown, necrotic, or mottled. Severely infected leaves may curl and drop from the vine. The disease also attacks older leaves in late summer and autumn, producing a mosaic of small, angular, yellow to red-brown spots on the upper leaf surface. Downy mildew is favoured by warm, wet growing seasons.The image of leaf infected with early blight is shown below in Figure.

Fig (2.2) Diseased leaf

2.1 EXISTING METHOD OR SYSTEM

Earlier papers are describing to detection of the plant leaves diseases using various approaches are discussed below,
In [1], they discussed automatic detection and classification of diseases. Plant disease spots are different in color but not in intensity. Thus color transform of RGB image is used for better segmentation of disease spots. Median filter is used for image smoothing and the Otsu method is used to calculate threshold values to detect the disease spot. It doesn't give accurate result for Dicot family plant.

P. Revathi and M. Hemalatha [4] investigated advance computing technology to assists the farmer in plant development process. This approach used mobile to capture infected cotton leaf images. RGB color feature segmentation is carried out to get disease spots. Edge detection technique is used for extraction of image features of spots to detect diseases. Neural network is used to categorize the diseases. The segmentation process is not suitable for Monocot family plant.
S. Dubey and R. Jalal [5] explored the concept of detection and classification of apple fruit diseases. The proposed approach is composed of three steps such as segmentation, feature extraction and classification. K-means clustering technique is used for the image segmentation. The features are extracted from the segmented image and images are classified based on a Multiclass Support Vector Machine (SVM). The proposed approach is specific to apple fruit diseases and cannot be extended to other fruit diseases.

In [6], the approach focused on Cercospora leaf spot detection in sugar beet using hybrid algorithms of template matching and support vector machine. The approach adopts three stages; first, a plant segmentation index of G-R is introduced to distinguish leaf parts from soil contained background for automatic selection of initial sub-templates. Second is robust template matching method adopted for continuous observation of disease development, foliar translation and dynamic object searching. Then, Support Vector Machine (SVM) is used to disease classification by a color features named two dimensional, xy color histogram. The segmentation process is not suitable for other Dicot family plant.

Yan, Han and Ming [7] proposed to select features of cotton disease leaf image by introducing fuzzy selection approach, fuzzy curves and fuzzy

surfaces. The features which are extracted from fuzzy selection approach are used for diagnosing and identifying diseases. This approach removes the dependent features of image so as to reduce the number of features for classification.

Sannakki, Rajpurohit, Nargund and Kulkarni [8] proposed an approach to diagnose the disease using image processing and artificial intelligence techniques on images of grape plant leaf. The input image of grape leaf is complex at background. The thresholding is used to mask green pixels and image is processed to remove noise using anisotropic diffusion. Then, segmentation is done using K-means clustering technique. The diseased portion from segmented images is identified. The results were classified using back propagation neural network.

In [9], they investigated approach for automatic detection of chilies plant diseases. For that, CIELab color transformation model is used to extract color feature from infected image.

Compare the color feature for detection of disease. There is no effective work done in feature extraction. But it could yield more result accuracy if appropriate work would have been done.

Next paper [10] discussed the monitoring of grapes and apples plant diseases. It suggests a solution to farmers for healthy yield and productivity. K-means clustering is used for segmentation and artificial neural network is used for classification of features. Also back propagation concept is used for counting the weight of mango. Morphology, color and texture features are extracted for classification.

2.2 PROPOSED SYSTEM

We have created the system based upon image processing and machine learning with the help of tensor flow library.

Fig 2.3 : Tensorflow

TensorFlow is a multipurpose machine learning framework. TensorFlow can be used anywhere from training huge models across clusters in the cloud, to running models locally on an embedded system like your phone.

Tensorflow Architecture
Tensorflow architecture works in three parts:

- Preprocessing the data
- Build the model
- Train and estimate the model

It is called Tensorflow because it takes input as a multi-dimensional array, also known as **tensors**. You can construct a sort of **flowchart** of operations (called a Graph) that you want to perform on that input. The input goes in at one end, and then it flows through this system of multiple operations and comes out the other end as output.

This is why it is called TensorFlow because the tensor goes in it flows through a list of operations, and then it comes out the other side.

List of Prominent Algorithms supported by TensorFlow

- Linear regression: tf.estimator.LinearRegressor
- Classification:tf.estimator.LinearClassifier
- Deep learning classification: tf.estimator.DNNClassifier
- Booster tree regression: tf.estimator.BoostedTreesRegressor

- Boosted tree classification: tf.estimator.BoostedTreesClassifier

Install TensorFlow

Before we can begin the tutorial you need to install TensorFlow version 1.7, and PILLOW.

If you have a working python environment you can install them with:

```
pip install --upgrade  "tensorflow==1.7.*"

pip install PILLOW
```

If this doesn't work, follow the instructions here.

If you have the git repository from the first codelab

This codelab uses files generated during the TensorFlow for Poets 1codelab. If you have not completed that codelab we recommend you go do it now. If you prefer not to, instructions for downloading the missing files are given in the next subsection.

In TensorFlow for Poets 1, you also cloned the relevant files for this codelab. Ensure that it is your current working directory, checkout the branch and check the contents, as follows:

```
cd tensorflow-for-poets-2
ls
```

This directory should contain three other subdirectories:

- The android/tflite directory contains all the files necessary to build a simple Android app using TFLite to classify images as it reads them from the camera. You will replace the model files with your customized versions.
- The scripts/ directory contains the python scripts you'll be using throughout the tutorial. These include scripts to prepare, test and evaluate the model.
- The tf_files/ directory contains the files you should have generated in the first part. At minimum you should have the following files containing the retrained tensorflow program:

```
ls tf_files/
retrained_graph.pb  retrained_labels.txt
```

Otherwise (if you don't have the files from the first Codelab)

Clone the Git repository

The following command will clone the Git repository containing the files required for this codelab:

```
git clone https://github.com/googlecodelabs/tensorflow-for-poets-2
```

Now cd into the directory of the clone you just created. That's where you will be working for the rest of this codelab:

```
cd tensorflow-for-poets-2
```

The repo contains three directories: android/, scripts/, and tf_files/

Checkout the files from the end_of_first_codelab branch

```
git checkout end_of_first_codelab

ls tf_files
```

Test the model
Next, verify that the model is producing reasonable results before starting to modifying it.
The scripts/ directory contains a simple command line script, label_image.py, to test the network. Now we'll test label_image.py on this picture of some daisies:

Fig 2.4 Plant leaf

Image CC-BY, by Fabrizio Sciami

Now test the model. If you are using a different architecture you will need to set the "--input_size" flag.

```
python -m scripts.label_image \
  --graph=tf_files/retrained_graph.pb  \
  --image=tf_files/flower_photos/daisy/3475870145_685a19116d.jpg
```

The script will print the probability the model has assigned to each flower type. Something like this:

```
Evaluation time (1-image): 0.140s

daisy 0.7361
dandelion 0.242222
tulips 0.0185161
roses 0.0031544
sunflowers 8.00981e-06
```

This should hopefully produce a sensible top label for your example. You'll be using this command to make sure you're still getting sensible results as you do further processing on the model file to prepare it for use in a mobile app.

Convert to model to TFLite format

TFLite uses a different serialization format from regular TensorFlow. TensorFlow uses Protocol Buffers, while TFLite uses FlatBuffers.
The primary benefit of FlatBuffers comes from the fact that they can be memory-mapped, and used directly from disk without being loaded and parsed. This gives much faster startup times, and gives the operating system the option of loading and unloading the required pages from the model file, instead of killing the app when it is low on memory.
We can create the TFLiteFlatBuffer with the following command:

```
IMAGE_SIZE=224
toco \
  --graph_def_file=tf_files/retrained_graph.pb \
  --output_file=tf_files/optimized_graph.lite \
  --input_format=TENSORFLOW_GRAPHDEF \
  --output_format=TFLITE \
  --input_shape=1,${IMAGE_SIZE},${IMAGE_SIZE},3 \
  --input_array=input \
  --output_array=final_result \
  --inference_type=FLOAT \
  --input_data_type=FLOAT
```

This should output a "optimized_graph.tflite" in your "tf_files" directory.

2.3 FEASIBILITY STUDY

Plant diseases have turned into a nightmare as it can cause significant reduction in both quality and quantity of agricultural products (Shen Weizheng, 2008influence the countries that primarily depend on agriculture in its economy (Prasad Babu, 2010). Consequently, detection of Plant diseases is an essential research topic as it may prove useful in monitoring large fields of crops, and thus automatically detect the symptoms of diseases as soon as they appear on plant leaves. Monitoring crops for to detecting diseases play a key role in successful cultivationBabu, 2010 and Weizheng, 2008). The nakedeye observation of experts is the main approach adopted in practiceWeizheng, 2008). However,this requires continuous monitoring of experts which might be prohibitively expensive in large farms. Further, in some developing countries, farmers may have to go long distances to contact experts, this makes consulting experts very expensive and time consuming(Prasad Babu, 2010). Therefore; looking for a fast, automatic,less expensive and accurate method to detect plant disease cases is of great realistic significance.

3. REQUIREMENT SPECIFICATION

3.1 FUNCTIONAL REQUIREMENTS

A functional specification (also, functional spec, specs, functional specification document (FSD)) in systems engineering and software development is the documentation that describes the requested behaviour of an engineering system. The documentation typically describes what is needed by the system user as well as requested properties of inputs and outputs. A functional specification is the more technical response onto a matching requirements document. Thus, it picks up the results of the

requirements analysis stage. On more complex systems multiple levels of functional specifications will typically nest each other.

3.1.1 PROCESS REQUIREMENTS

A computer running on Windows 7/8/10 is required for this applications to run. Python version 3.7 or more is required to run the raw code on desktop. An android mobile phone is required to operate the android application.

Additionally, The android mobile should have camera which should be able to capture a picture of leaf to be diagnosed.

3.1.2 DESIGN CONSTRAINTS

As information moves through the software, it is modified by a series of transformation. When developing a system with constraint, the constraints are developed based on user requirements.

3.1.3 EXTERNAL INTERFACE REQUIREMENTS

The android app is user friendly. For implementation of reusable raw code, the basic knowledge of python is required as for later, no interface is provided.

3.2 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements are requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that specify specific behavior or functions. Typical non-functional requirements are reliability, performance, scalability, and cost. Non-functional requirements are often called the ilities of a system. Other terms for non-functional requirements are "constraints", "quality attributes" and "quality of service requirements".

3.2.1 PERFORMANCE

User trying to load unsupported text formats. It should show appropriate message for such cases. The product should not crash under any circumstances such as user entering invalid values.

3.2.2 RELIABILITY

If any exceptions occur during the execution of the software it should be caught and thereby prevent the system from crashing.

3.2.3 SCALABILITY

The system should be developed in such a way that new modules and functionalities can be added, thereby facilitating system evolution.

3.2.4 COST

The cost should be low because a free availability of software package.

3.2.5 PORTABILITY

Our product will be portable in form .apk files. Additionally, to support research python source code will be set to open sourced.

4. SYSTEM ANALYSIS

4.1 PRESENT SCENARIO

Plant diseases cause periodic outbreak of diseases which leads to large scale death and famine. It is estimated that the outbreak of helminthosporium of rice in north eastern India in 1943 caused a heavy loss of food grains and death of a million people.Since the effects of plant diseases were devastating, some of the crop cultivation has been abandoned. It is estimated that 2007 plant disease losses in Georgia (USA) is approximately $653.06 million (Jean, 2009). In India no estimation has been made but it is more than the USA because the preventive steps taken to protect our crops are not even one-tenth of that in the USA.The naked eye observation of experts is the main approach adopted in practice for detection and identification of plant diseases. But, this requires continuous monitoring of experts which might be prohibitively expensive in large farms. Further, in some developing countries, farmers may have to go long distances to contact experts, this makes consulting experts too expensive and time consuming and moreover farmers are unaware of non-native diseases.

Fig 4.1 : Present Scenario

4.2 PROPOSED SYSTEM

In the proposed system, at first the images of leaf of the suspected diseased plants are provided to the app. This can be done by either taking a picture of the leaf from the plant through the application's take picture option or can be imported from the gallery. The image is then analysed for any disease by the app with the built in system and finally, output (whether the plant is infected or not) is displayed on the screen with accuracy equal to or above 0.94.

5. SYSTEM DESIGN

5.1 ARCHITECTURE DESIGN

In order to ease the implementation, verification and documentation, the system could be divided into three different components based on its functionalities. Fig- ure 3.1 below shows the architecture of the program. The first part is the android application as a client which provides a user interface to interact with the functions of the application. The user can choose to take a picture or select an image from the gallery of the mobile phone as the leaf image to be diagnosed. Once the user decides which leaf picture he/she wants to use, the android application will send the image to the tensorflow

lite CNN model. The second part is the trained model which is added as an asset in the android app. The third part is the main component of the application which is used to recognise the category of the disease and implemented in Python, and it is also the most difficult and central program.

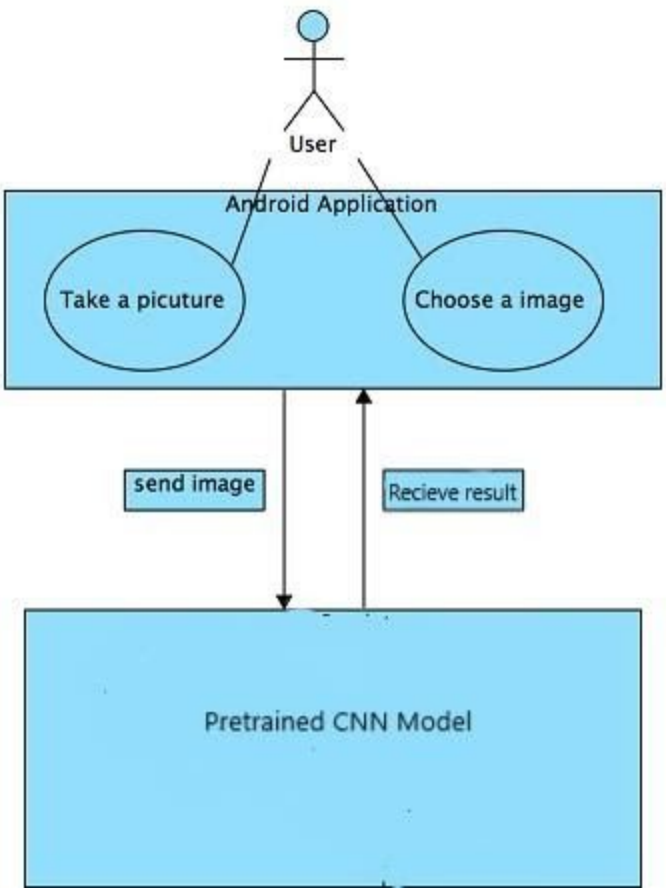

Fig 5.1 : System Architecture

5.2 USER INTERFACE DESIGN

Figure 5.2 shows the graphical user interface draft of the application in android mobile phone. As the difficulty of the project focused on the implementation of di- agnosis algorithm, the function buttons and provided operation are limited. There are two functional buttons at the bottom of the screen which allows the user to choose wether they want to use an existing leaf image in the gallery or take a new

photo of leaf. Above the buttons, user can see the diagnosis result in this area which contains category labels for every leaf. The design of the logo of the application is shown in Figure 5.2 below.

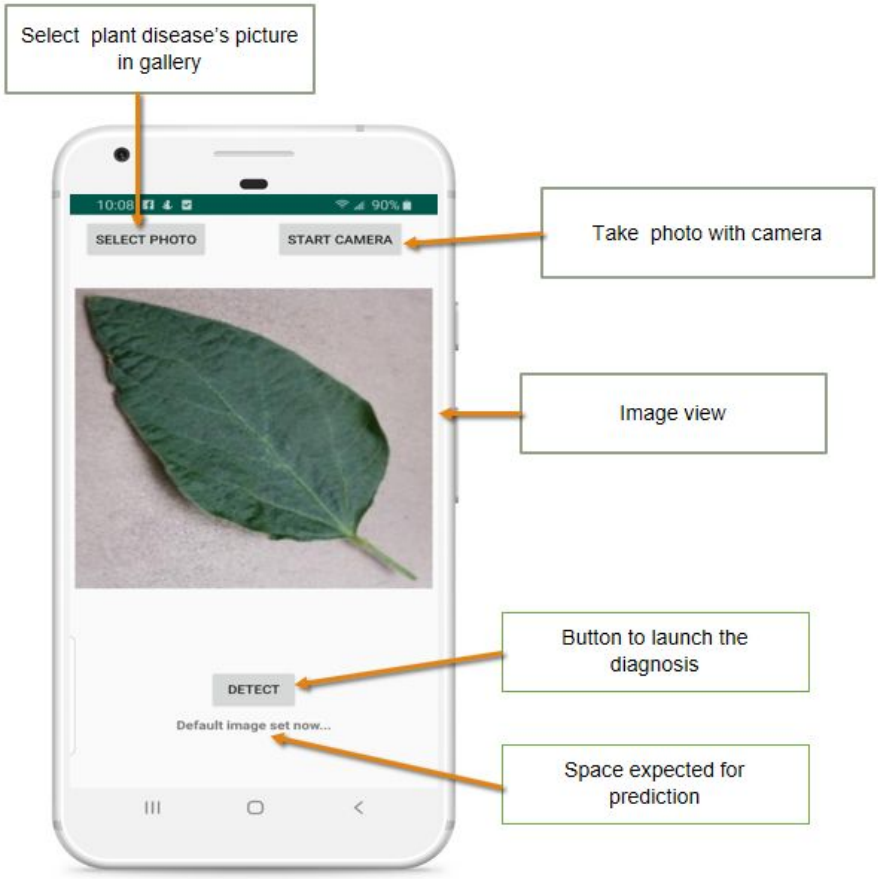

Fig 5.2 User Interface Design

6. SYSTEM IMPLEMENTATION

6.1 DISEASE IDENTIFICATION MODEL DEVELOPMENT

6.1.1 ANN

Artificial Neural Nets are computational model based on structure of biological neural nets designed to mimic the actual behaviour of a biological

neural network which are present in our brain. make the correct identification of disease and also guiding the right quantity of fertilizers. Only a single ANN can't get our job done. So, we make a bunch of them stacking one on other forming a layer which we can make multiple layers in between input layer(where weights and data are given) and output layer(result) those multiple layers called Hidden layers and it then makes a ***Deep Neural Network and the study of it called Deep Learning.***

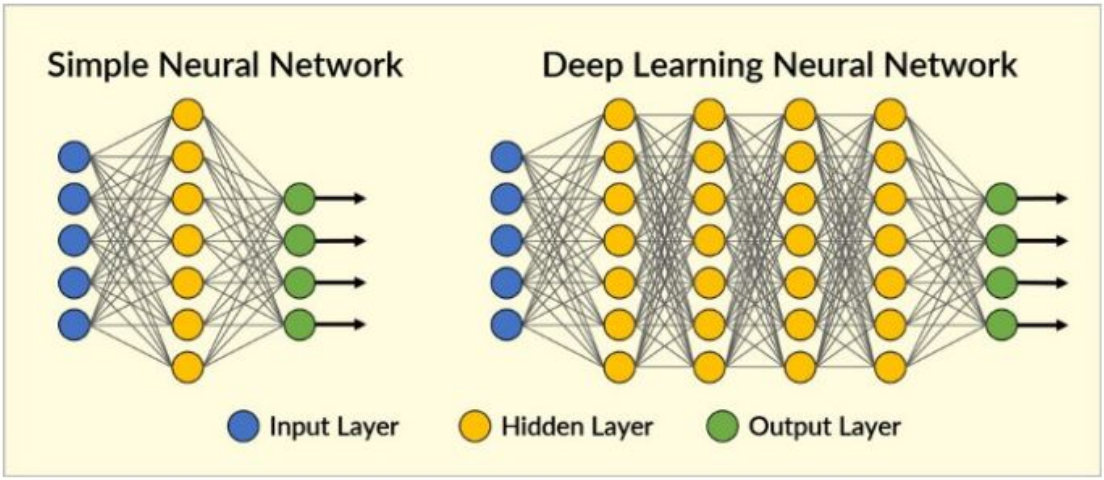

Fig 6.1 : Simple vs Deep Neural Nets

Simple Neural Nets are good at learning the weights with one hidden layer which is in between the input and output layer. But, it's not good at complex feature learning.

On another hand Deep Learning Neural Nets, the series of layers between input and output layer are called hidden layers that can perform identification of features and creating new series of features from data, just as our brain. The more layers we push into the more features it will learn and perform complex

operations. The output layer combines all features and makes predictions. Therefore, Simple Neural Nets are used for simple tasks and bulk data isn't required to train itself. whereas in Deep learning Neural Network can be expensive and require massive data sets to train itself on. I'm not gonna discuss this topic cause it goes beyond this article.

6.1.2 CONVOLUTION NEURAL NETWORK

It is well known for its widely used applications of image and video recognition and also in recommender systems and Natural Language Processing(NLP). However, **convolutional** is **more** efficient because it reduces the number of parameters which makes it different from other deep learning models.

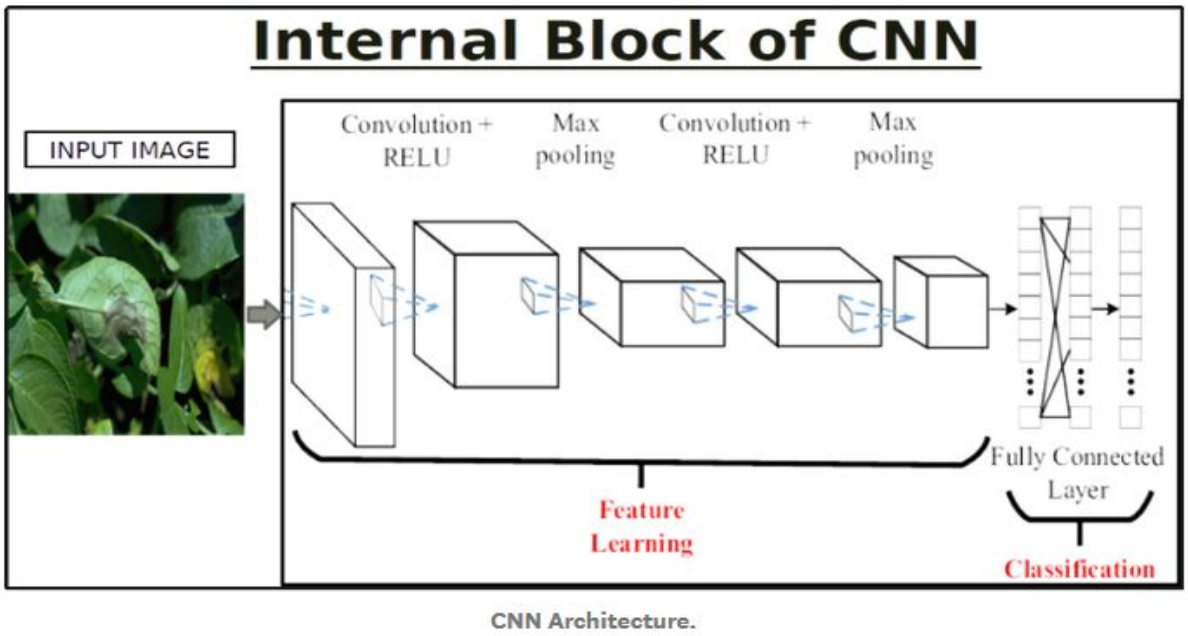

Fig 6.2 CNN Architecture

Main Steps to build a CNN (or) Conv net:

1. Convolution Operation
2. ReLU Layer (Rectified Linear Unit)
3. Pooling Layer (Max Pooling)
4. Flattening
5. Fully Connected Layer

1. Convolution is the first layer to extract features from the input image and it learns the relationship between features using **kernel or filters** with input images.

2. **ReLU Layer:** ReLU stands for the Rectified Linear Unit for a non-linear operation. The output is ***f(x) = max(0,x).*** we use this because to introduce the nonlinearity to CNN.

3. **Pooling Layer:** it is used to reduce the number of parameters by downsampling and retain only the valuable information to process further. There are types of Pooling:

- Max Pooling (Choose this).
- Average and Sum pooling.

4. **Flattening:** we flatten our entire matrix into a vector like a vertical one. so, that it will be passed to the input layer.

5. **Fully Connected Layer:** we pass our flatten vector into input Layer .We combined these features to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs.

6.1.3 IMAGE PREPROCESSING

The image acquired is preprocessed. The preprocessing starts by converting the RGB image to L*a*b* color space. The L*a*b* color space consists of Luminosity layer L*, chromaticity layer a* and b*. All of the color information is stored in the layers a* and b*. It requires to make color form so that the RGB colored image is converted to L*a*b* space. The function is makecform(), later the format is applied to the image that was acquired.

6.1.4 BUILDING CNN MODEL

First we preprocess and normalize the images. Rescaling image values between (0 –1) called normalization. Whatever pre-processing you do with the train it should be done to test parallelly. All these parameters are stored in the variable "**train_datagen and test_datagen".** Then we take the path to a directory, and generate batches of augmented data. Yields batches indefinitely, in an infinite loop. Batch Size refers to the number of training examples utilized in one iteration. Fig 6.3 shows the model summary.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 252, 252, 32)      2432
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 84, 84, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 82, 82, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 41, 41, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 39, 39, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 19, 19, 64)        0
_________________________________________________________________
flatten (Flatten)            (None, 23104)             0
_________________________________________________________________
dense (Dense)                (None, 512)               11829760
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_2 (Dense)              (None, 6)                 774
=================================================================
Total params: 11,926,374
Trainable params: 11,926,374
Non-trainable params: 0
```

Fig 6.3: Model Summary

6.1.5 TRAINING THE MODEL

For Training the model Adam optimizer is used with learning rate=0.001. Loss function categorical_crossentropy is used for our **Multi-class classification problem**. Metrics is **" accuracy".** Fit_generator is used to train the CNN model. Using the validation data parameter for **fine-tuning the model.**

6.1.6 VISUALIZATION

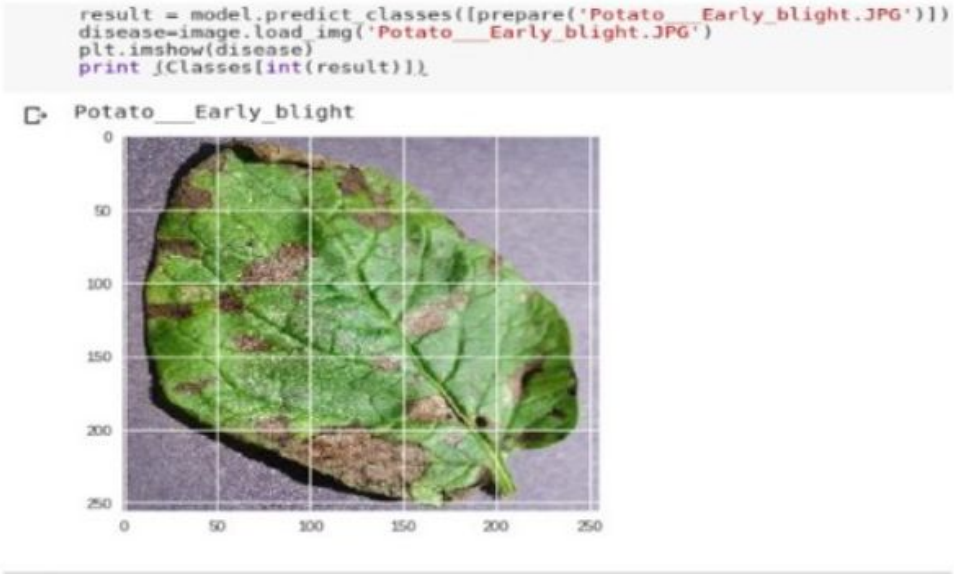

Fig 6.4 : Potato Early blight detection

```
[45] result = model.predict_classes([prepare('Tomato___Late_blight.JPG')])
     disease=image.load_img('Tomato___Late_blight.JPG')
     plt.imshow(disease)
     print (Classes[int(result)])
```

```
Tomato___Late_blight
```

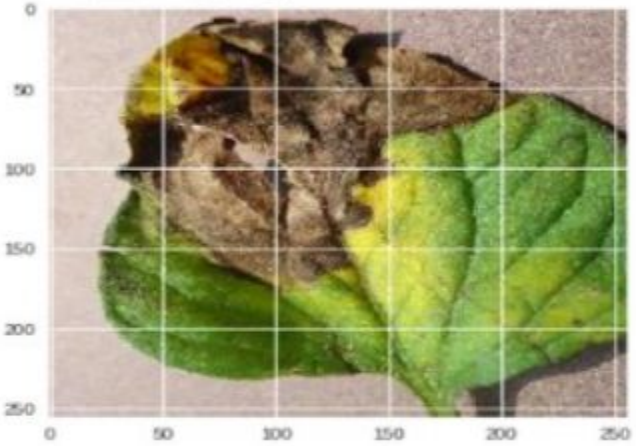

Fig 6.5 : Tomato late blight detection

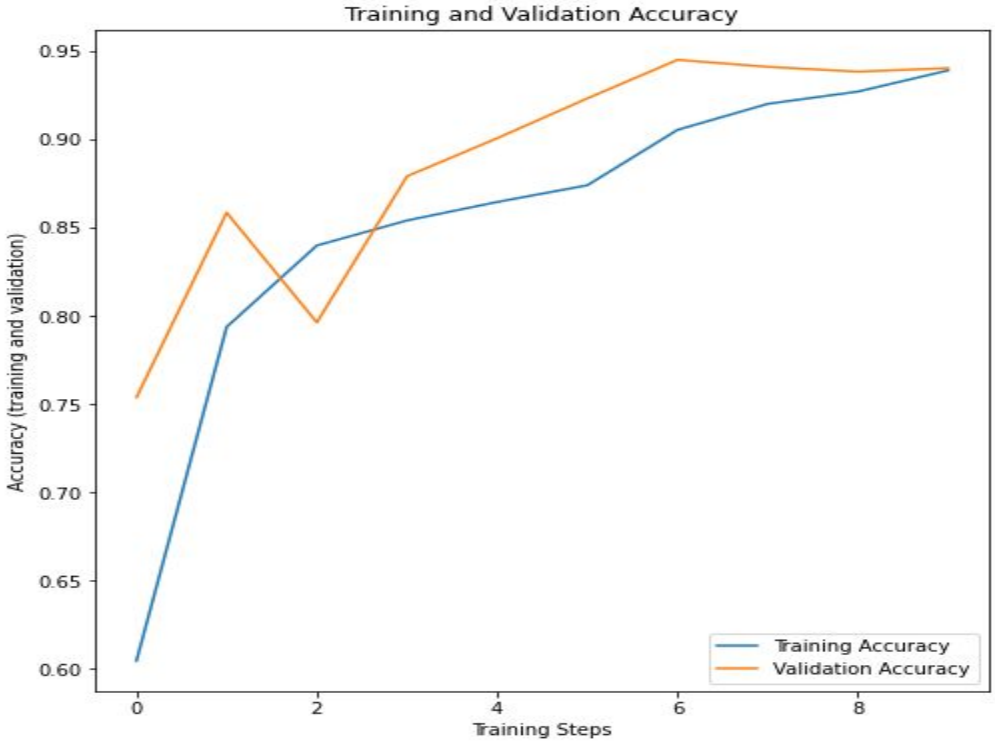

Fig 6.6 : Training and validation accuracy

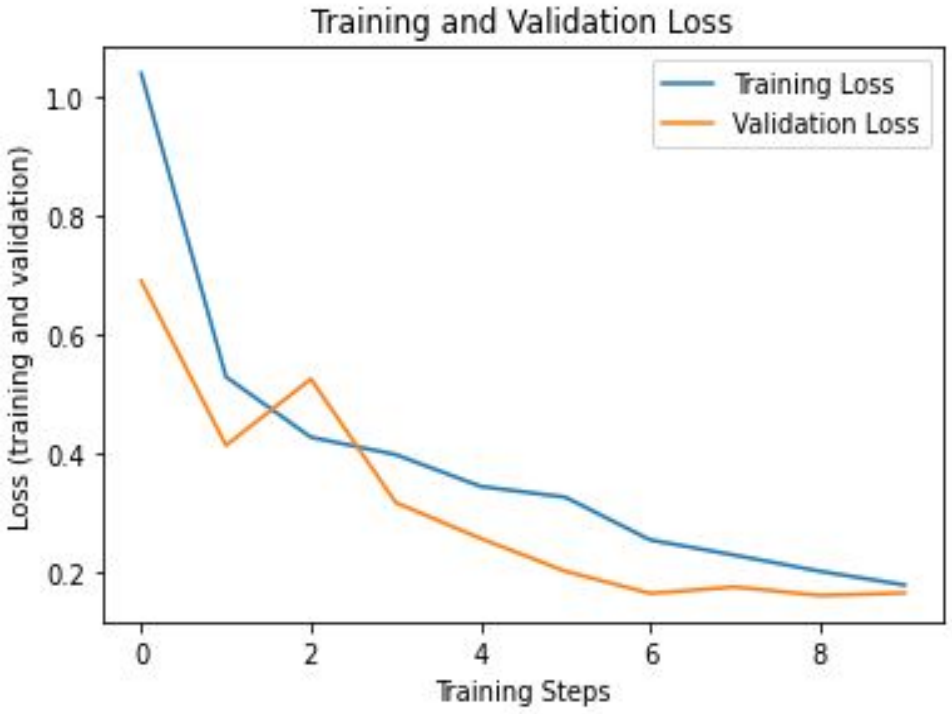

Fig 6.7 : Training and validation loss

6.2 DEPLOYMENT

We deployed the model using an Android Application developed with Android Studio Using Kotlin language. Application is completely offline. Users can click the picture of the leaf or can choose the picture from the gallery then this image will be passed to our pretrained model for the identification.

Fig 6.8 : Deployment GUI

7. TESTING

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test.Software testing can also provide an objective, independent view of the software allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs(errors or other

defects).Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test:meets the requirements that guided its design and development,responds correctly to all kinds of inputs,performs its functions within an acceptable time,is sufficiently usable,can be installed and run in its intended environments, andAchievesthe general result its stakeholder‟sdesire.As the number of possible tests for even simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As a result, software testing typically (but not exclusively) attempts to execute a program or application with the intent of finding software bugs(errors or other defects). The job of testing is an iterative process as when one bug is fixed;it can illuminate other, deeper bugs,or can even create new ones.Software testing can provide objective, independent information about the quality of software and risk of its failure to users and/or sponsors.Software testing can be conducted as soon as executable software (even if partially complete) exists. The Overall approach to software development often determines when and how testing is conducted. For example, in a phased process, most testing occurs after system requirements have been defined and then implemented in testable programs. In contrast, under anAgile approach, requirements, programming, and testing are often done concurrently.

7.1 THE MAIN AIM OF TESTING

The main aim of testing is to analyze the performance and to evaluate the errors that occur when the program is executed with different input sources

and running in different operating environments.In this project, we have developed a Android Application and a Image Processing code which helps in detection of disease. The main aim of testing this project is to check if the disease is getting detected accurately and check the working performance when different images are given as inputs.The testing steps are:

- Unit Testing
- Integration Testing
- Validation Testing
- User Acceptance Testing
- Output Testing

7.2 UNIT TESTING

Unit testing, also known as component testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. The following Unit Testing Table 7.1 shows the functions that were tested at the time of programming. The first column gives all the modules which were tested, and the second column gives the test results. Test results indicate if the functions, for given inputs are delivering valid outputs.

Function Name	Test Result
Uploading Image	Tested for uploading different types and sizes of images.
Detecting disease	Tested for different images of Tomato and Potato leaf of different classes
Get Message	Tested if message is displayed successfully

Table 7.1 Unit Testing Table

7.3 INTEGRATION TESTING

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

7.4 VALIDATION TESTING

At the culmination of integration testing, software is completed assembled as a package. Interfacing errors have been uncovered and corrected. Validation

testing can be defined in many ways; here the testing validates the software function in a manner that is reasonably expected by the customer.

In software project management, software testing, and software engineering, verification and validation(V&V) is the process of checking that a software system meets specifications and that it fulfills its intended purpose. It may also be referred to as software quality control. The following table indicates validation tests done for checking functionality of the project after its integration with the front-end.

.

Functionality to be tested	**Input**	**Tests Results**
Working of Choose File option	User convenience to access images stored	Different folders where images are stored can be uploaded
Working of View Message	User select the image	Details of disease are displayed

Table 7.2: Validation Testing Table

7.5 USER ACCEPTANCE TESTING

Performance of an acceptance test is actually the user's show. User motivation and knowledge are critical for the successful performance of the system. The above tests were conducted on the newly designed system performed to the expectations. All the above testing strategies were done using the following test case designs**.**

7.5.1 WHITE BOX TESTING

White Box Testing sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases. Using White Box Testing, we can derive test cases that: Guarantee that all independent paths within a module have been exercised at least once. Exercise all logical decisions on their true and false sides. Execute all loops at their boundaries and within their operational bounds.

7.5.2 BLACK BOX TESTING

Black Box Testing focuses on the functional requirements of the software. This enables us to derive sets of input conditions that will fully exercise all functional requirements for a program. Black Box testing attempts to find errors in the following categories:Incorrect or Missing Functions.Interface errors.Performance errors.Initialization and Termination errors.

8. CONCLUSION

This chapter summarises the features and evaluation of the software that is overall successful on the project specification. System limitation will be talked about to show what more functions could be added and which part of the application can be optimised. After that I will personally reflect myself on this project in the personal reflection part. Finally future work will be discussed on the possible extension of the project if further development needs to be done.

Summary

A plant disease recognition method that can support an extensible set of diseases is presented in this paper. This method can be implemented merely by the resources of a smartphone that does not have to be connected to a server. A few photographs of plant parts (e.g., less than 25% of the test set) that have been infected by a disease are used for the extraction of invariant features like the number and area of the spots, color histogram features, weather data, etc. The limits (wide and narrow) of these features are used to define disease signatures employed by a fuzzy clustering method that decides which is the most likely disease displayed in the analyzed photograph. The experimental results show that the disease recognition can be achieved with accuracy between 80% and 98% in most of the cases, competing with several popular classification methods. The achieved accuracy depends on the disease, the number and type of training samples used for the definition of the disease signatures and the employed spot-separation method. The most important advantages of the proposed approach are the extensibility of the supported disease signatures, the low complexity owed to the simple features used for the classification and the independence from the orientation, the resolution and the distance of the camera. Future work will focus on the support of different plants and diseases. Alternative classification methods like pure fuzzy clustering will also be tested as long as they provide extensibility of the supported set of diseases.

System Limitation

Although the system has many positive aspects and successfully achieved all the goals on the specification, it could still be enhanced to be a publishable application. In terms of the segmentation, the leaves cannot be segmented into

very good results if the background is too complex which is nearly an unavoidable problem. For feature extraction, more complicated features can be tried in experiments. It is a complicated part, many papers discuss that in different complex methods. However, implementing all the candidate algorithms is not practical given a limited amount of time. The system only trained three categories of disease of the same plant with limited amounts of images which can not lead generally used in reality. But the project is just a research in this research field, the recognition could be more generalised and better if more images are given and trained in the future.

Personal Reflection

On one hand, it is challenging to implement such a mobile application to identify the disease in such a complex background, particularly for me without image processing background before. Since no same application exists in software store and many researchers are studying this topic, it is impossible for an undergraduate student to build a perfect application for that. However, I worked hard and tried my best to learn and research some image processing and machine learning techniques so that the application was implemented which achieved the goals in the specification. On the other hand, with few requirements and constraints, the goals are clear and the system design is flexible.

The project brought valuable experience, so that I learnt lots of things which have a profound influence on me, both on research and programming skills. Before doing the project, I have no knowledge on computer vision and machine learning. The process makes me very interested in machine learning and probably choose to study this field in my postgraduate course. I also learnt how to use TensorFlow to process images and data, and know how to use the statistics and machine learning functions built in Python. It also taught me task planning and time management which would be much helpful in my future research career. My reading and self-learning skills are improved by reading academic papers and writing skills are enhanced by writing the dissertation. Through the project, self-motivation and perseverance are also mixed with my character.

9. FUTURE SCOPE

Many extensional functions can be added to the application which could make it a real application in a Google Store or App Store. Other types of software such as website could also be implemented to make the program more conveniently to attain. The application should allow the user to cut the leaf region out if the background in too big or too complex, which would be very help to segment the leaf out successfully. More other advanced features can be implemented and tested to optimise the algorithm to get a better result. Other categories of leaves with different diseases should be trained via experiments to make the application applicable to as many categories of plants as possible. The application software can be built in some additional functions such as viewing histories, saving locations and so on.

REFERENCES

- https://www.tensorflow.org/
- https://towardsdatascience.com
- https://medium.com
- https://google.com
- https://kaggle.com