AMITY UNIVERSITY

UTTAR PRADESH

MAJOR PROJECT

on

PLANT AND FRUIT DISEASE DIAGNOSIS AND TREATMENT THROUGH DEEP LEARNING

Submitted to

Amity University Uttar Pradesh

in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

In

Computer Science & Technology By

SACHIN KUMAR(A2305219503)

MAYANK KHANDELWAL(A2305219504)

TUSHAR GOYAL(A2305219540)

ISHU KHANDELWAL (A2305219549)

Under the guidance of **Dr.**

Ram Paul

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY UTTAR PRADESH

DECLARATION

We, **Sachin Kumar, Mayank Khandelwal, Tushar Goyal and Ishu Khandelwal** students of B.Tech (Computer Science Engineering) hereby declare that the project "**PLANT AND FRUIT DISEASE DIAGNOSIS AND TREATMENT THROUGH DEEP LEARNING**" which is submitted by us to Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, in partial fulfilment of requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering, has not been previously formed the basis for the reward of any degree, diploma or other similar title or recognition. We hereby declare that we have gone through project guidelines including policy on health and safety, policy on plagiarism, etc.

SACHIN KUMAR

MAYANK KHANDELWAL

TUSHAR GOYAL

ISHU KHANDELWAL

Date: 12-06-2023

CERTIFICATE

On the basis of declaration submitted by **Sachin Kumar, Mayank Khandelwal, Tushar Goyal and Ishu Khandelwal** student of B.Tech. CSE, I hereby certify that the major project titled " **PLANT AND FRUIT DISEASE DIAGNOSIS AND TREATMENT THROUGH DEEP LEARNING**" which is submitted to Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in CSE, is an original contribution with existing knowledge and faithful record of work carried out by her under my guidance and supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Date: 12-06-2023

Dr. Ram Paul

Assistant Professor –II

Department of Computer Science and Engineering

Amity School of Engineering and Technology Amity University Uttar Pradesh, Noida

ACKNOWLEDGEMENT

The satisfaction that accompanies that the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible, whose constant guidance and encouragement crown all efforts with success. I would like to thank Prof (Dr) Sanjeev Thakur, Head of Department-CSE, and Amity University for giving me the opportunity to undertake this project. I would like to thank my faculty guide Dr. Ram Paul who is the biggest driving force behind my successful completion of the project. He has been always there to solve any query of mine and also guide me in the right direction regarding the project. Without his help and inspiration, I would not have been able to complete the project. Also, I would like to thank my team guides who guided me, helped me and gave ideas and motivation at each step.

ABSTRACT

Technology has percolated into each and every sphere of our life. What seemed impossible yesterday is implemented today and evolved tomorrow. Agricultural sector is bound to change a lot than it was in these past decades. A lot has changed and use of technology has increased a lot in helping the farmers increase their production. Farmer these days have access to smartphone devices in which they can search the type of pesticides and fertilizers to use in order to improve production and prevent crop damage. But they cannot use any chemical without any expert's advice. The endpoint of these advancements in agricultural sector is to make the expert's opinion available to the farmers present anywhere event in remote locations. This can be done by developing a suitable application where they can check their crops health and see measures to prevent or cure diseases. This application would make use of machine learning and deep learning algorithms in order to detect the disease infecting the plant and with the help of the data already fed in the database it would be able to tell the farmer about how to cure a particular disease which is affecting his crops.

Identification of Diseases using Computer Vision field is of utmost interest. Agriculture is the most important sector of Indian Economy. Indian agriculture sector accounts for 18 percent of India's GDP and provides employment to 50% of the population. In this era of technology, we are trying to propose a method which can help farmer to detect diseases without going to any Agriculture center. Our purpose is to devise an application which can capture the photo of the plant leaves and it'll automatically detect the disease and its corresponding treatment using Artificial Intelligence. So, we have created Neural Network in Keras and trained the model over 17217 images validated over 4303 images. Then, Field Testing was done on 235 images and result is quite satisfactory with the average accuracy of 90.60%. Further, a cross-platform application using flask Server and Flutter Framework was created which can help the farmer.

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER 1: DETAILED PROBLEM STATEMENT

1.1 INTRODUCTION

The importance of agriculture cannot be over emphasized in today's world. The quality and the quantity of the food we eat depends on the agricultural sector. The better the health of the crops the better would be the health of the people and most importantly healthy farming would increase the farmers output and hence the profit. But unfortunately, farmers in today do not have access to expert's advice on which chemical or biological measures are to be taken to keep the crop healthy.

There should be a technological means through which a farmer, without going to the local vendor or a specialist physically, could somehow have access to the remedies recommended by the experts by sitting in their farm while looking over their crops so that immediate preventive action or cure could be done if his/her crops become diseased or damaged.

Our project predicts the plant's disease using the image uploaded by the user and gives the remedies for that particular disease by showing the chemical cure, biological cure and the preventive measures so that the disease does not occur in future. It takes the input in the form of an image which uses the camera of the mobile and by using the model running at the server at the backend it predicts the disease and gives the cure. So, the farmer, instead of going to the local vendor or any specialist, gets the expert's recommendation in his mobile device itself.

1.2 MOTIVATION

The main motivation behind "Plant Disease Diagnosis and Treatment Through Deep Learning" is to provide remedies recommended by experts and scientists to the farmers at their home or fields where they can cure their crops immediately and using correct medicines. There have been attempts to solve these problems, but none of the solutions have been particularly effective. However, application of deep learning in agricultural sector to detect and cure crop diseases has potential to change all these problems. If diseases can be caught in their early stages the we could prevent the worldwide loss of economy related to agricultural sector.

Diseases occurring in plants does not only affect the farmer but even affect the economy of the entire country. Sometimes the diseases are so worse that they can infect the entire farm within one night and all the crops get damaged till the sunrise and the entire yield have to be discarded.

This severity of the matter shows that how important technology has become because at that time the farmer cannot heed the advice of any expert or scientist which a mobile application would be able to give.

The below is the table showing the economic losses due to agricultural sector in various countries:

Country (Year)	Estimated Loss (mt)	Loss as Percentage of Expected Output (%)	Value of Production Loss (US$ Million, 1994)
Thailand (1994)	130	58	650
Philippines (1989)	57	96	285
Ecuador (1992)	34	27	170
Indonesia (1991)	50	34	250
China (1992)	180	84	900
Taiwan (1987)	100	72	500
Mexico (1994)	1	8	5
USA (1993)	12	NA[1]	60
India (1994)	25	36	125
Vietnam (1994)	10	20	50
Bangladesh (1994)	5	14	25
Total	**541**	**74**	**3,019**

[1]NA = not available.

Figure 1. 1 Loses due to diseased Crops

The above table clearly illustrates the sorry state of agricultural sector and the loses occurring due to diseases and spoiled crops. Although this data is quite old but the situation has even now not been improved.

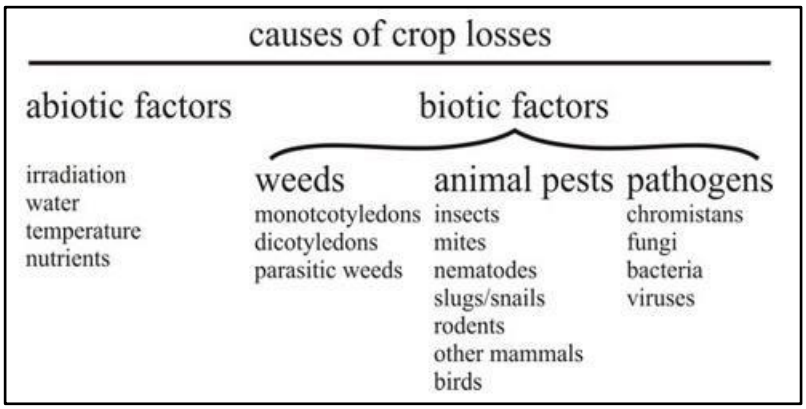


Figure 1. 2 Cause of crop loses

The use of deep learning and AI techniques can improve the sorry condition of our agricultural sector not only in India but all over the world. This would not only help the farmers to improve

their production and get more profits but also increase the nutrients of the food that we intake and would lead to a better health of the people consuming the crops.

These are the few reasons which inspired various organizations and researchers to focus on developing several techniques in order to detect and not only cure but also prevent diseases occurring in plant by using machine learning and deep learning techniques. This inspires us as well to develop a model for predicting the plant diseases and provide the cure and preventive measures.

1.3 PROBLEM STATEMENT

Develop a mobile application that predicts plant's disease and recommend the remedies to cure that disease and also tell the preventive measures so that the disease doesn't occur in future. The app would interact with a backend server which has a model running to predict the plant disease and the server transmits back the result to the application to show to the user.

1.3.1 Input to the System

The input to the system is the data received from the camera of the smartphone device. The input would be in the form of image that would be captured by the camera.

1.3.2 Output from the System

The output from the system will be the name of the disease that is infecting the plant in the uploaded image along with the chemical and biological cure along with the preventive measures.

1.4 SIMILAR SYSTEMS STUDY

1.4.1 Yellow Vein Mosaic Virus Disease Detection System

As explained by (D. Mondal, 2015) Okra is a common crop in India advantages include ease of planting and versatility to changing temperature and humidity conditions. The most widely known okra disease is the YVMV.

The system is designed for the recognition and prevention of YVMV disease in okra leaf by leaf vane detachment. The designed application detects whether an Okra plant is healthy or not

by analysing images of Okra leaves. Figure 1 shows the process of the proposed system which contains two junctures.

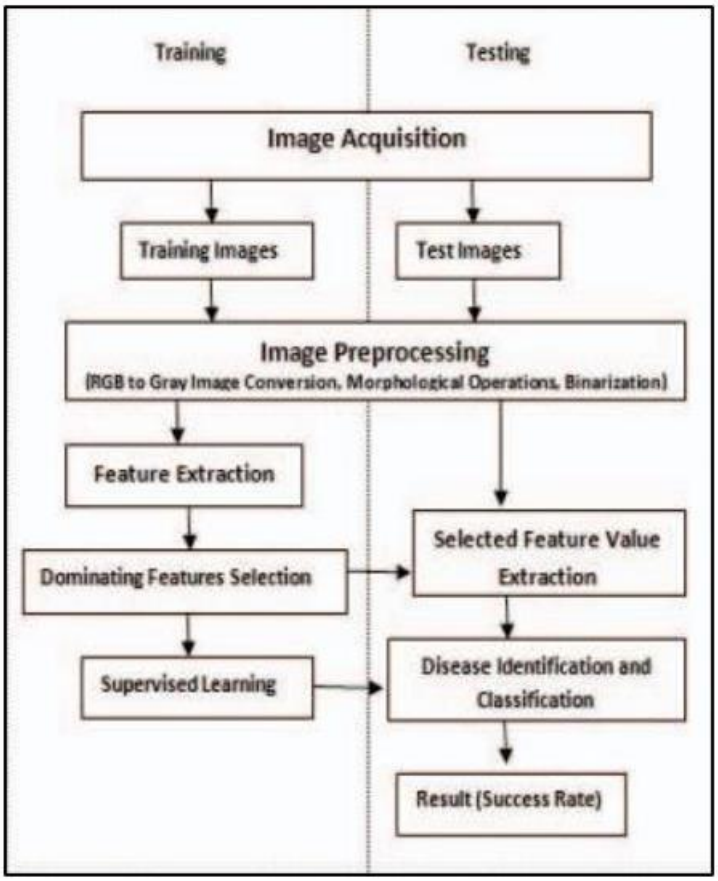


Figure 1. 3 Flow chart of proposed method

K-Means algorithm has been used as the primary requirement of Naive Bayes classifier in order to classify and recognize YVMV diagnosis. The proposed method achieved 87% success rate.

1.4.2 Plant Disease Identification Mobile Application

As (Naveen Chandra Gowda, 2019) has established, the system consists of an android mobile application that provides a feature in which farmers can scan the suspicious plant for disease from the application camera and identify the plant disease.

The front-end module consists of the 'detect a disease' option and the method to detect various plant types. The user can enter all the details of the infected plant and can get remedies accordingly. Following figure an example of a plant image scanning process of the application.
Input: Image of the diseased plant

Output: Diagnosis result

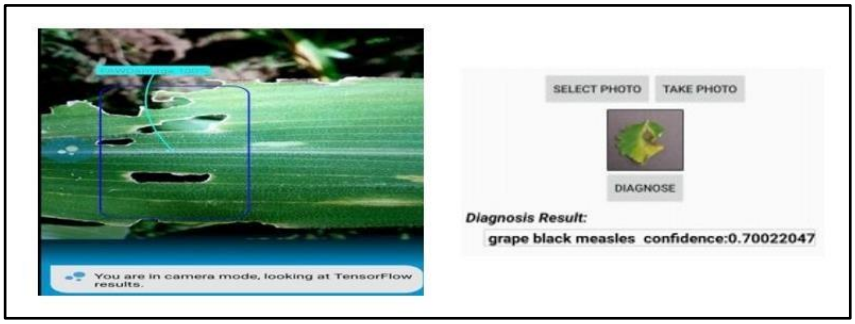


Figure 1. 4 Working in disease identification

A TensorFlow model specially developed on the dataset of infected plants to perform plant recognition process. The approach used to identify crop disease is the use of the Convolutional Neural Network (CNN). Data set has been separated in to 10:1 ratio as training data and validation data. The clustering algorithm of machine learning is used which are prewritten in the TensorFlow framework. The model is optimized by using the optimization techniques in the TensorFlow such as model optimization, pruning, stemming. It is found that the deep learning model in the system has performed a 97.61% accuracy.

1.4.3 Jute Plant Disease Detection System

In the system as (Z. N. Reza, 2016) has pointed out the images that will be taken from device camera will be sent to server-side application through the client-side mobile application. Images will be analysed according to image texture features by using classification techniques, finally the detected crop disease name will be displayed on user interface.

The client-side application is a simple android application which will be sent to the server and the server is responsible for the plant disease identification and providing classified result through an API. Sever side operation can be described as below.

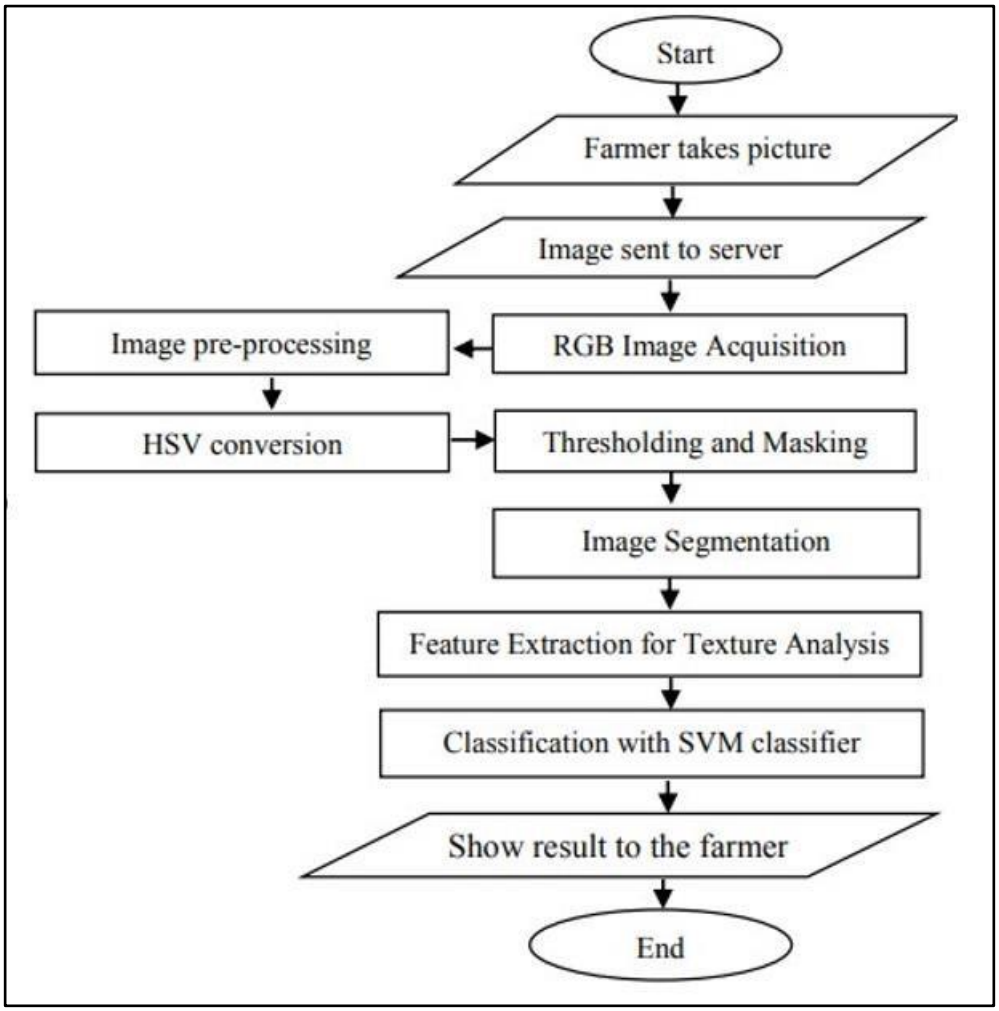


Figure 1. 5 Flowchart of overall process

The plant image will be merging through a pre-processing procedure in order to acquiring a better result. The method of pre-processing the image is carried out by following those steps: image resizing, edge enhancement and image filtering.

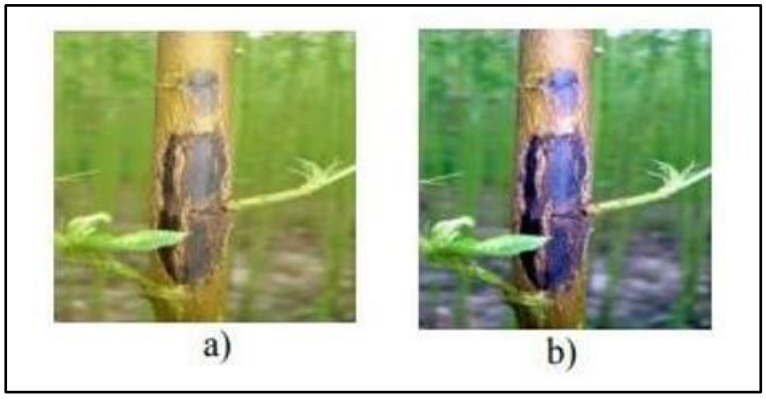


Figure 1. 6 A jute stem infected by anthracnose a) Original Image b) Enhanced Image

Using eight-way local neighbourhood measurement the diseased plant area will be segregated and in order to further examine the segmented section, it is important to return to its original color by converting it to RGB format.

Figure 1. 7 Segmented RGB Image

The color co-occurrence methodology developed through the GLCM was used in feature extraction. The SVM algorithm is used to describe the disease. Most notably, the accuracy of the hand-on-use standard (81%) is almost the same as the accuracy of the test photos (87%).

1.4.4 Crop disease recognition System Using Machine Learning

Contrary to (al., 2018) Pathogens and crop diseases can lead to reduce in food production leading to food malnutrition. In order to address above issue a desktop application which is capable of detecting plant diseases was developed using machine learning.

The application provides the feature to the farmer to upload a picture of the diseased crop leaf and the system returns the condition of the plant. Some steps have been taken to determine if the leaf is diseased or healthy. Ex: pre-processing, Extraction of Features, Classifier Learning and Identification.

The system has been used random forests classifier. Dataset is divided in to two parts such as training and validation data. Using HOG feature extraction technique feature vector is produced and trained using Random Forest algorithm. Feature vector generating using HOG feature descriptor for testing data and testing data can be represented as the given image below.

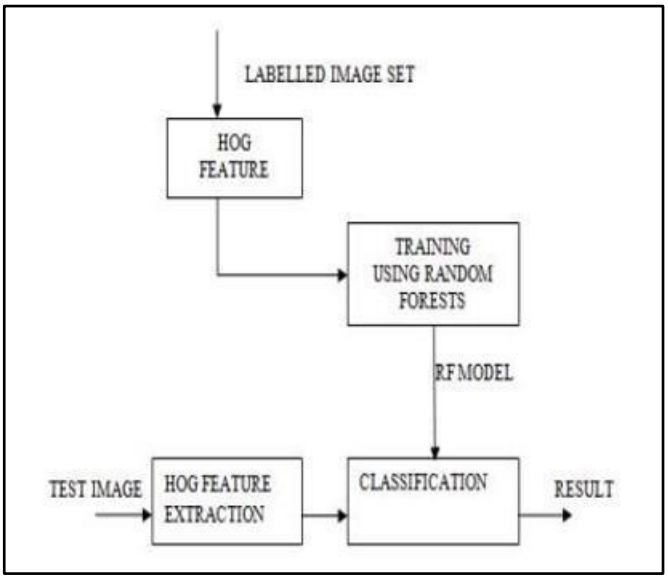


Figure 1. 8 Model architecture flow

Application could classify diseased plants with approximate 70% percent accuracy.

CHAPTER 2: LITERATURE SURVEY

2.1 MACHINE LEARNING

Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can change when exposed to new data.

The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. However, instead of extracting data for human comprehension; as is the case in data mining applications, machine learning uses that data to detect patterns in data and adjust program actions accordingly. Machine learning algorithms are often categorized as being supervised or unsupervised.

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output. Supervised learning problems are categorized into "regression" and "classification" problems. In a regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. In a classification problem, we are instead trying to predict results in a discrete output. In other words, we are trying to map input variables into discrete categories.

Unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don't necessarily know the effect of the variables.

2.2 NEURAL NETWORK

Artificial neural networks (ANNs) are a computational model used in machine learning, computer science and other research disciplines, which is based on a large collection of connected simple units called artificial neurons, loosely analogous to axons in a biological brain. Connections between neurons carry an activation signal of varying strength. If the combined incoming signals are strong enough, the neuron becomes activated and the signal travels to other neurons connected to it. Such systems can be trained from examples, rather than explicitly programmed, and excel in areas where the solution or feature detection is difficult to express in a traditional computer program. Like other machine learning methods, neural networks have been used to solve a wide variety of tasks, like computer vision and speech recognition, that are difficult to solve using ordinary rule-based programming.

Typically, neurons are connected in layers, and signals travel from the first (input), to the last (output) layer. Modern neural network projects typically have a few thousand to a few million neural units and millions of connections; their computing power is similar to a worm brain, several orders of magnitude simpler than a human brain. The signals and state of artificial neurons are real numbers, typically between 0 and 1. There may be a threshold function or limiting function on each connection and on the unit itself, such that the signal must surpass the limit before propagating. Back propagation is the use of forward stimulation to modify connection weights, and is sometimes done to train the network using known correct outputs

Neural Networks are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value.

Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values. The following illustration shows a simple Neural Network:

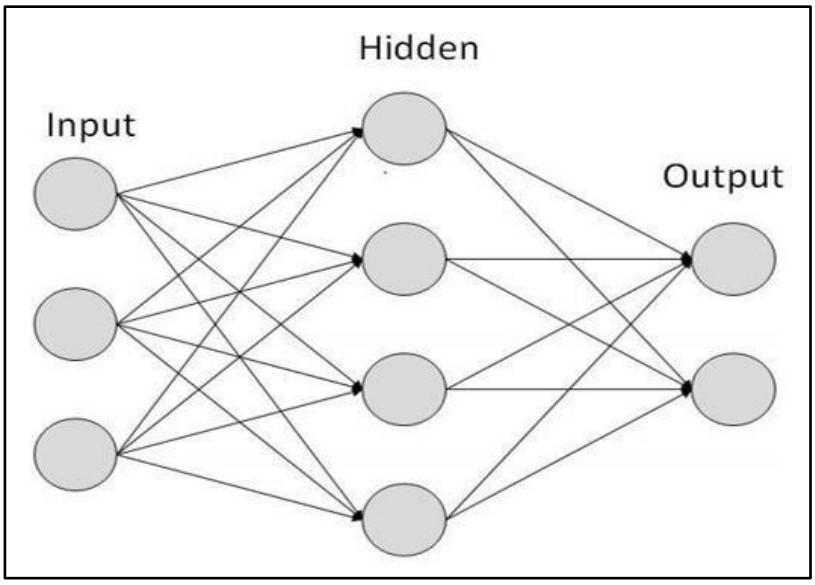


Figure 2. 1 An example of an Artificial Neural Network (ANN)

2.3 CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks (CNNs) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. CNN's have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self-driving cars.

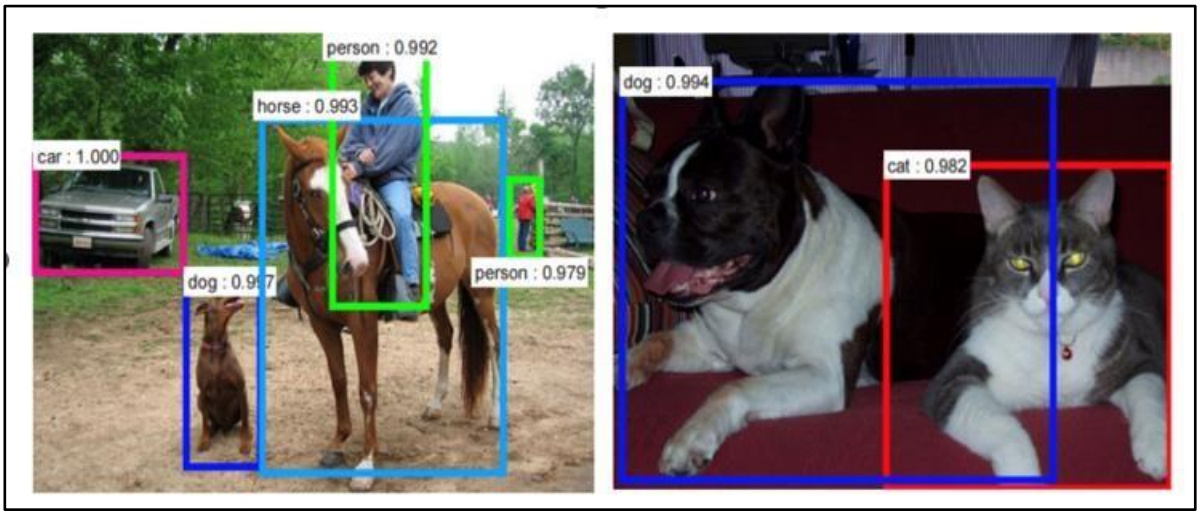


Figure 2. 2 An example of ConvNets being used for recognizing everyday objects, humans and animals

2.3.1 Convolutional Neural Network & Deep Learning Model

According to researches and observations which were performed by Naveen Chandra Gowda (2019) following architecture is used on this project in order to detect plant diseases. Using the model that project was able to get average accuracy of 97.6% which makes the chosen algorithms and technologies were good fit for the project.

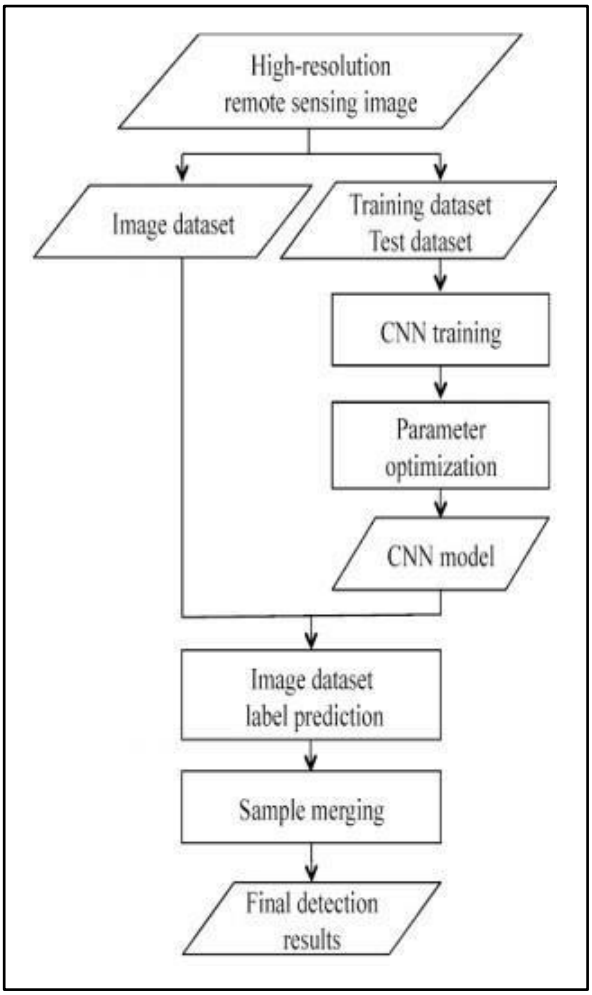


Figure 2. 3 Deep learning model flow

After the pre-processing process of the image, the training data set will be fed to neural network's initial layer where the model starts training according to the images provided. The plant village dataset (which was mentioned earlier) is divided in to different classes which will be necessary to perform a valid label prediction. Following model parameters will be used in this project as followings;

Convolutional Neural Network

- Number of layers: 4
- Output feature maps: 256

The following table demonstrates the architecture that will be used to construct a convolutional neural network model.

	Layer 1	Layer 2	Layer 3	Layer 4
Features in	1	64	128	256
Features out	64	128	256	512
Filter size	3	3	3	3
Max Pool Filter Size	2,2	2,2	2,2	2,2

2.3.1.1 Convolutional Layer

Convolutional layers are the main building blocks of a convolutional neural network. Application of a filter to an input (in this project it is an image) that results in the activation of the filter can be simply defined as convolution. Repeat of applying a certain filtration to the gradual return in an activation map called a feature map, signifying the stance and intensity of the feature intercepted in the source image.

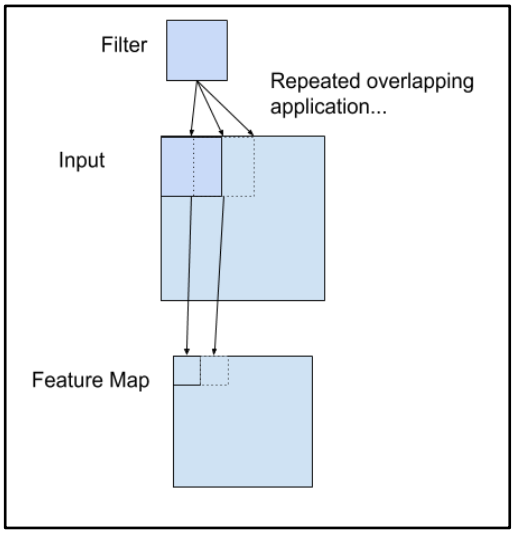


Figure 2. 4 CNN- filter mapping

Having ability to automatically learn a large number of filters in tandem specified to a training dataset under the constrictions of a stipulated predictive modelling problem is the specialty of neural networks. Ex: classifying images. Intensely stipulated features that can be identified wherever on input images can be taken as the result.

There are 4 convolutional layers are being used with various sizes of filter sizes. The filter sizes are spread from 32 to 512 in order to get more accurate predictions in the project. (3, 3) is used as kernel size in order to reduce memory usage and compute more accurately and faster.

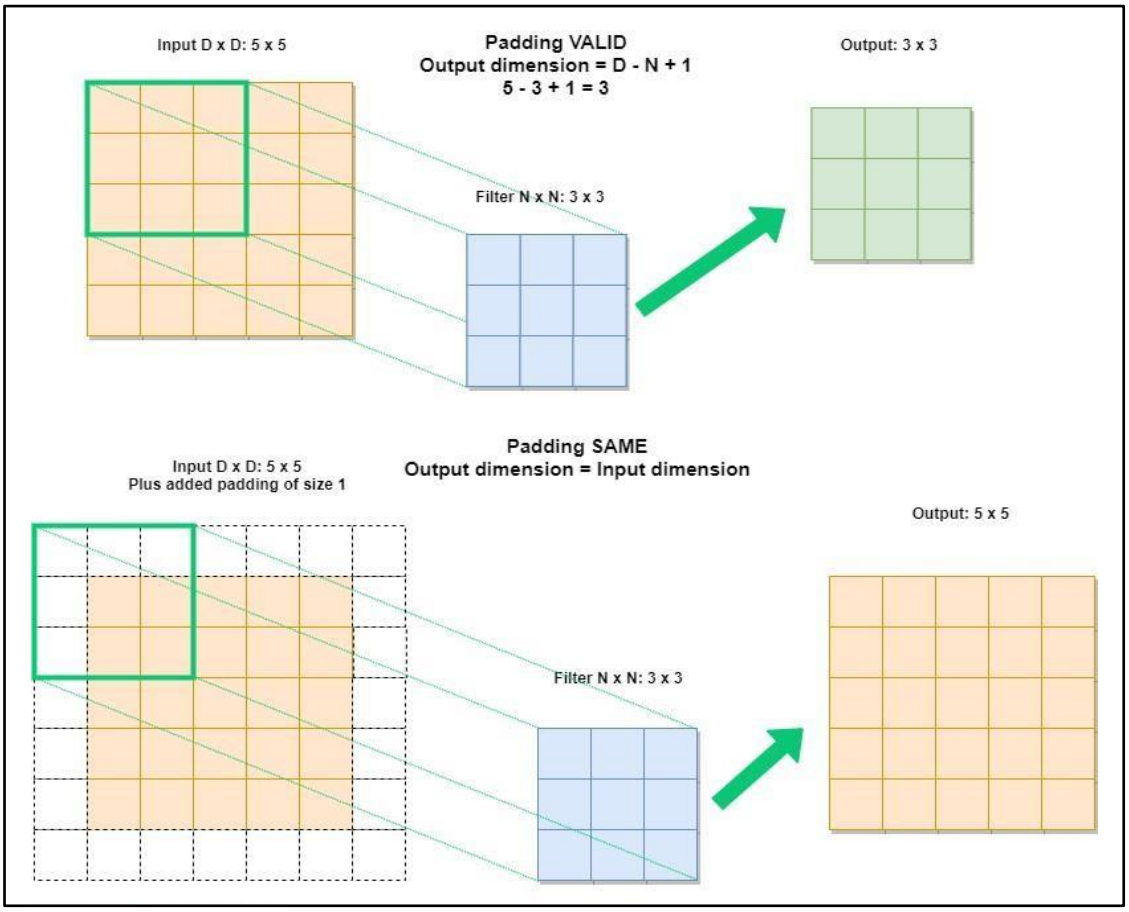


Figure 2. 5 Convolutional layer-filter adding

Above image emphasizes how the filters are being applied to a given image in a convolutional layer.

2.3.1.2 Pooling Layer

The input image goes through the convolution layer, then the activation process, and finally into the pooling layer in CNN. In order to calculate the highest merit represented by the kernel and to generate a downsize-matrix max pooling is used, which results in a shift in the height and width of the matrix while the width remnants constant. This often significantly reduces the hardware power required for processing of images.

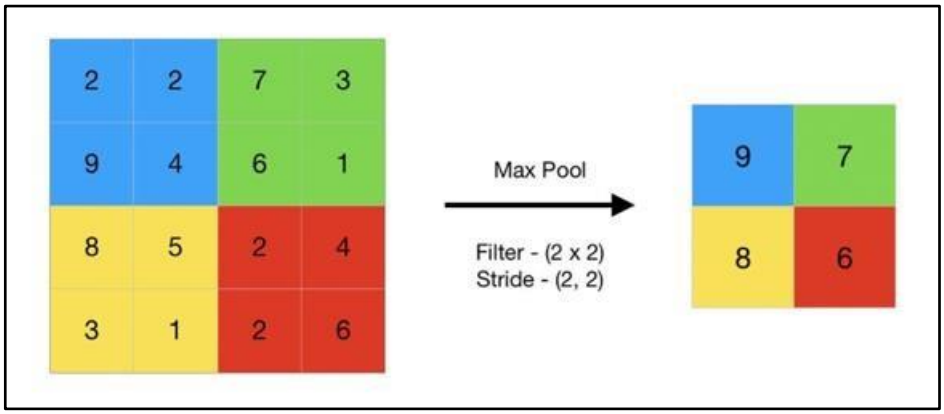


Figure 2. 6 Max pooling

Above image is a graphical representation of a max pooling process of a Convolutional Neural Network (CNN). In this project (2, 2) is used as the pooling size of the layer of max-pooling because of the importance of reducing number of parameters when dealing with color images. Even though it is necessary to minimize parameters since this is a plant disease detection system plant leaf should be analysed critically, keeping general information of the plant leaf is important. Considering above mentioned fact (2, 2) sized max pooling layer will be a perfect fit to the CNN model.

2.3.1.3 Activation function

In deep learning neural network activation can be taken as a key component. The success of prediction model can be evaluated through activation methods. Its exactness and consistency, as well as the computational effectiveness of preparing a show that can make or break a large-scale neural network. In other words, activation function can be defined as the mathematical conditions which decide the yield of a neural network. Actuation functions habitually tend to legitimize the yield of each neuron to a run between 1 and or between-1 and 1.

And the activation function may be a transformation that produces an input signal to the output signals necessary for the neural network to continue.

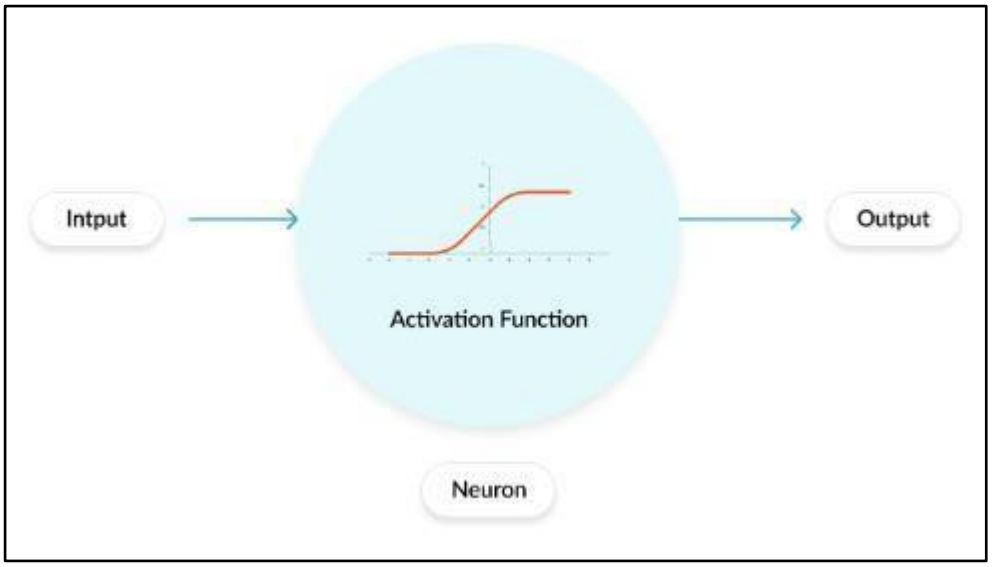


Figure 2. 7 Activation function process

In this project as the activation function of convolutional layers ReLU has been utilized considering its computational efficiency. In the dense layer SoftMax is used as the activation layer hence the project uses a multiclass model as well as SoftMax activation function mostly widely presented in the last layer of the model which needs to classify numerous categories.

2.3.1.4 Dropout Layer

Dropout layers are normally used in Convolutional Neural Network models in order to prevent a model getting over fitted. In the project there is a one dropout layer used sized 0.6 in order to make sure the model is not getting over-fitted (which was discussed earlier) which acts as a regularize in the neural network.

2.4 SOFTWARE DESIGN

2.4.1 Activity diagrams

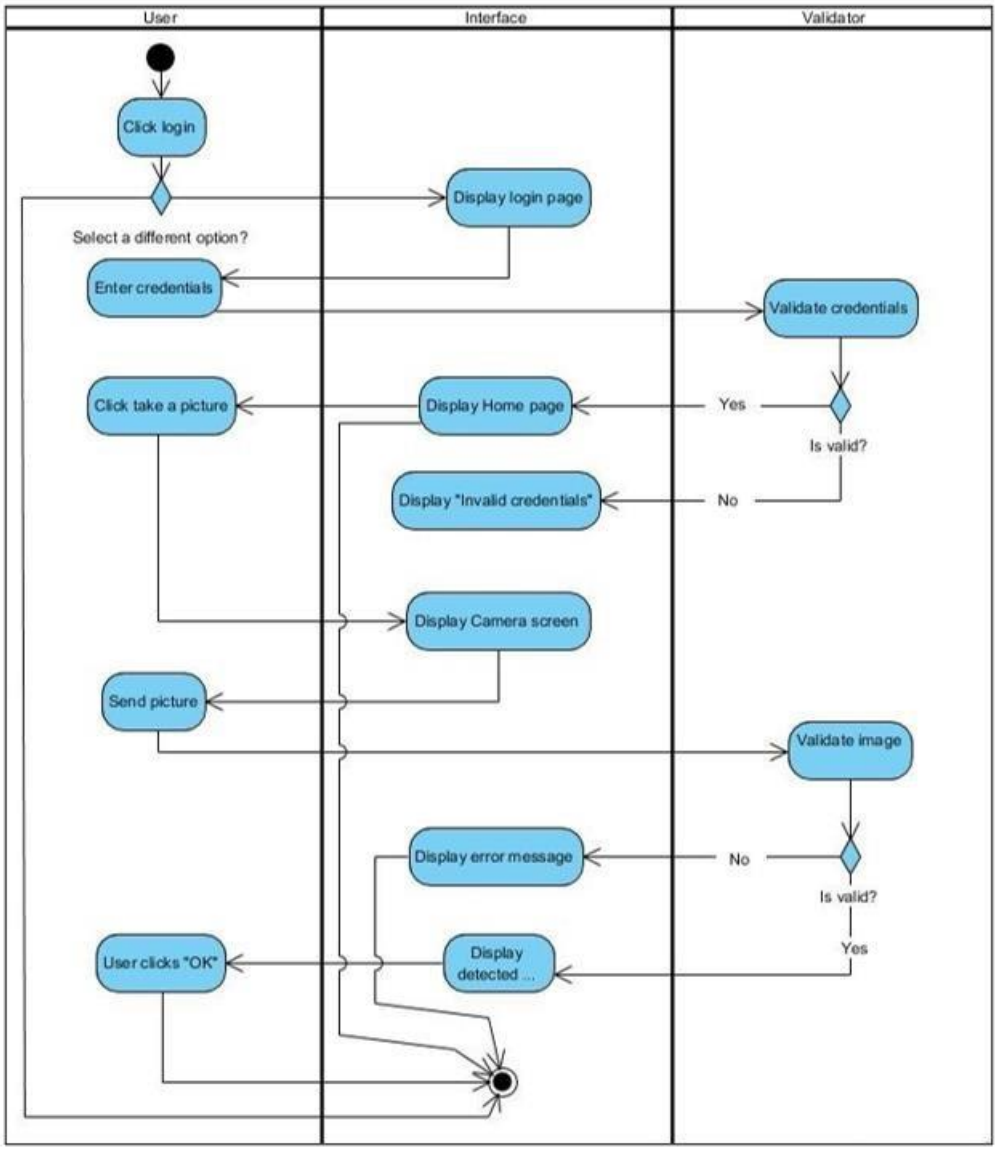


Figure 2. 8 Activity diagram-plant disease detection

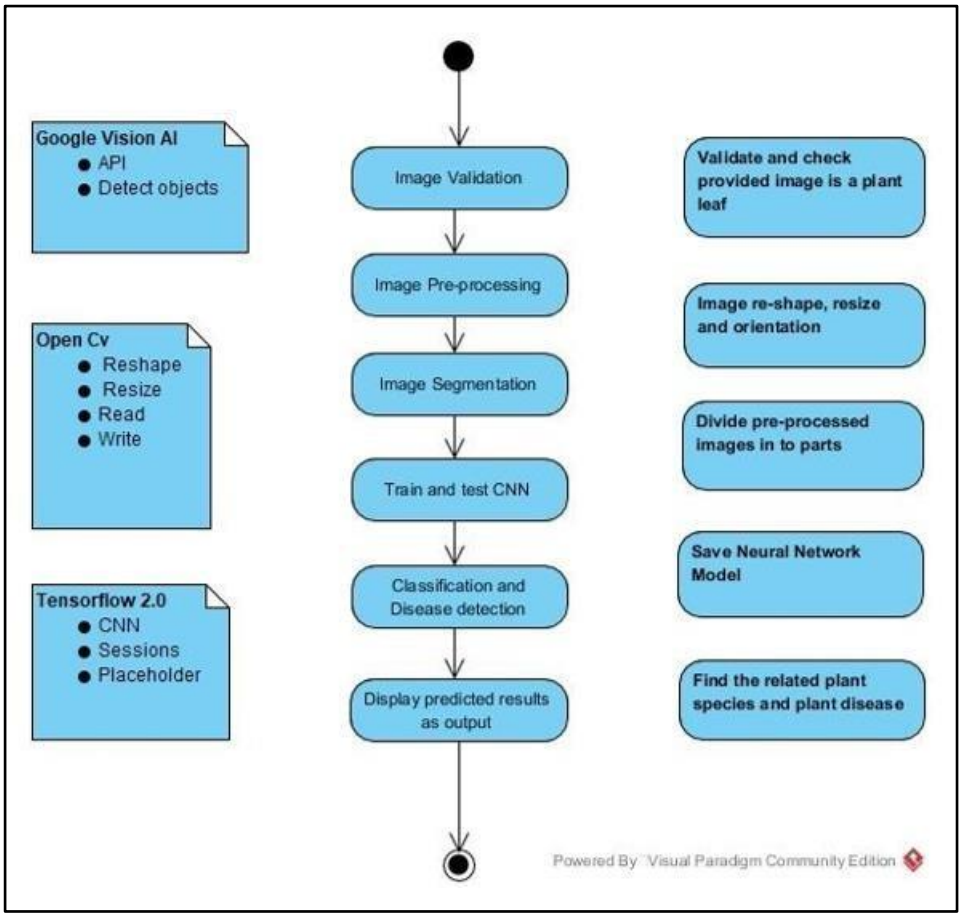


Figure 2. 9 Activity diagram-system procedure

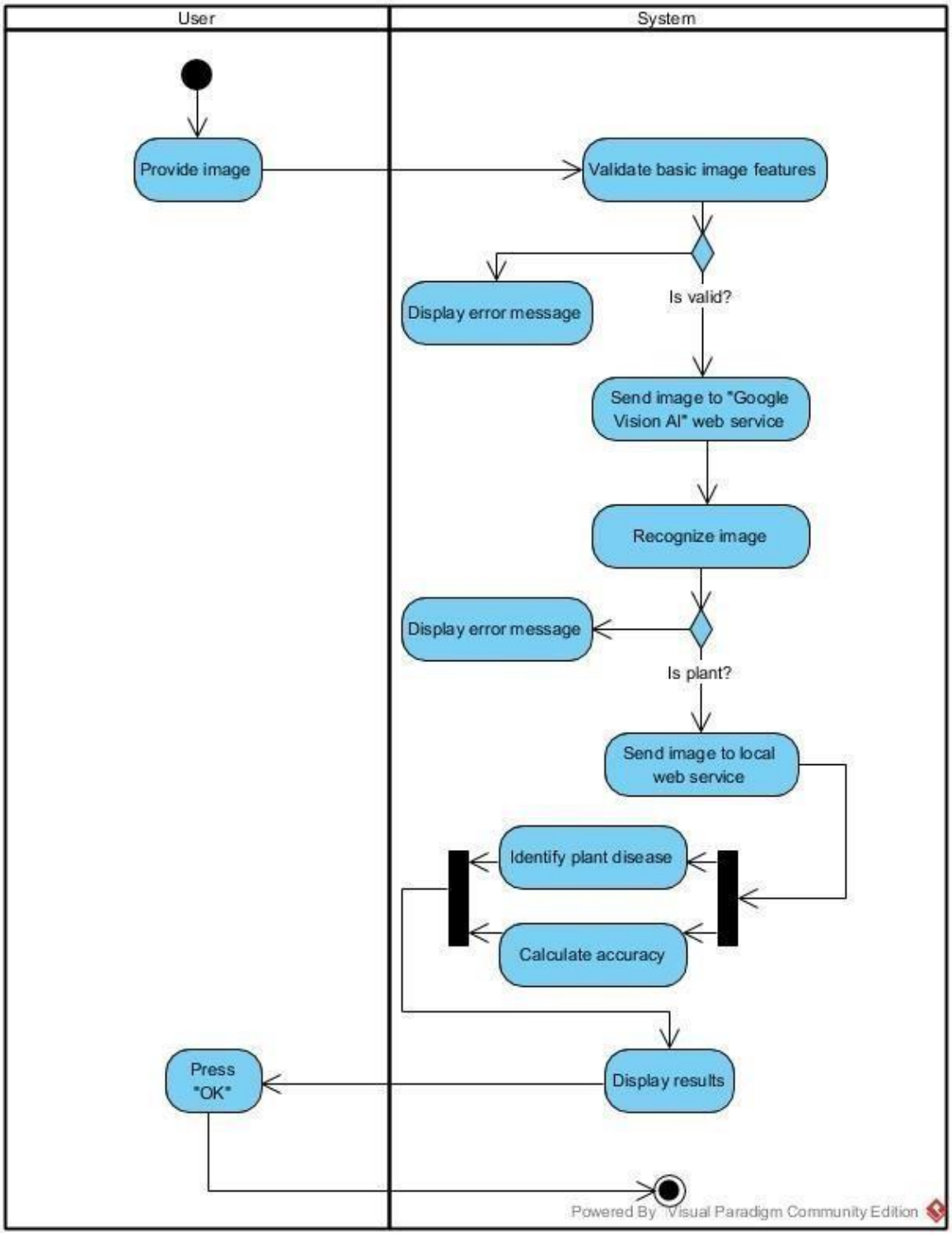


Figure 2. 10 Activity diagram-image validation

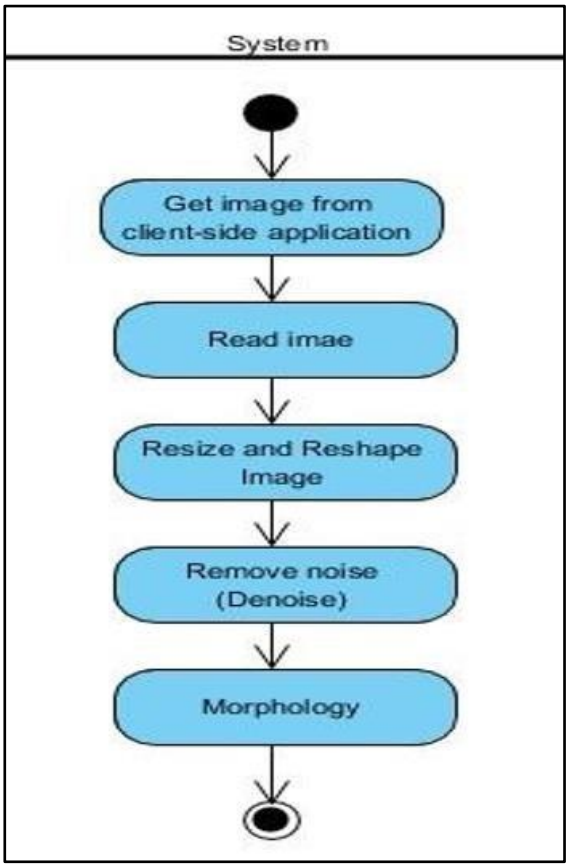


Figure 2. 11 Activity diagram-image pre-processing

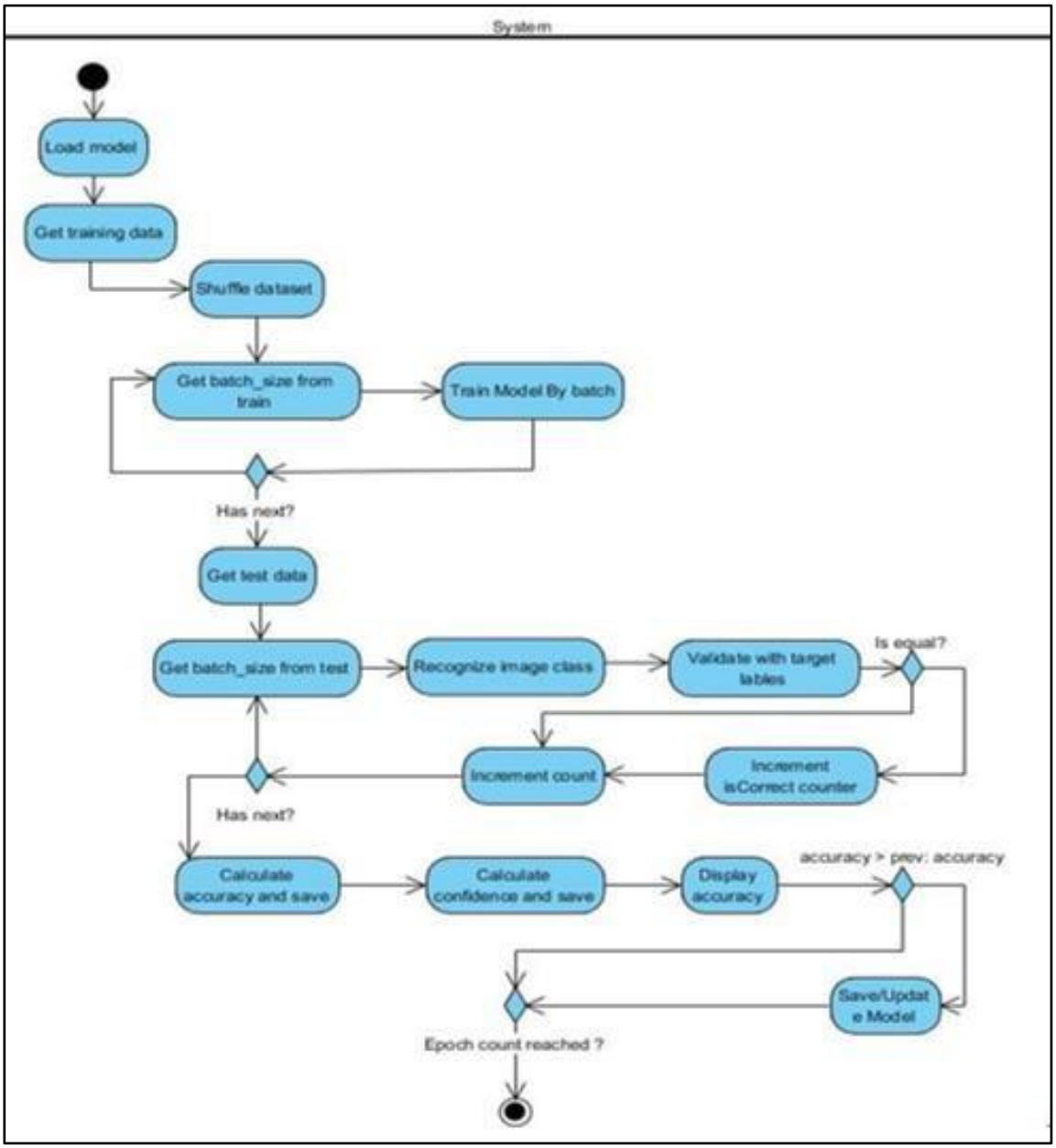


Figure 2. 12 Activity diagram create CNN model

2.4.2 Class diagrams

According to the analysis the application will be developed using react native and redux will be used as the state management library of the client-side application. Following diagram represents the state management process of the client-side mobile application.

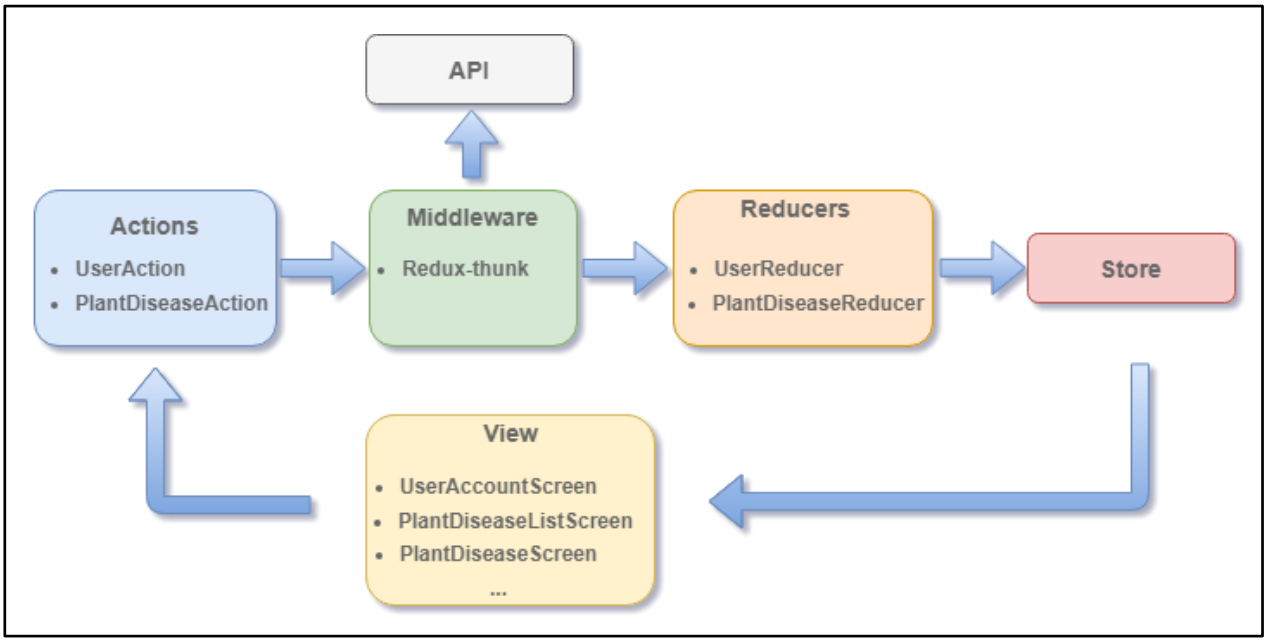


Figure 2. 13 Client Side-State management flow (Redux)

Following diagram represents the class diagram of client-side mobile application of the system which is responsible for client-side operations.

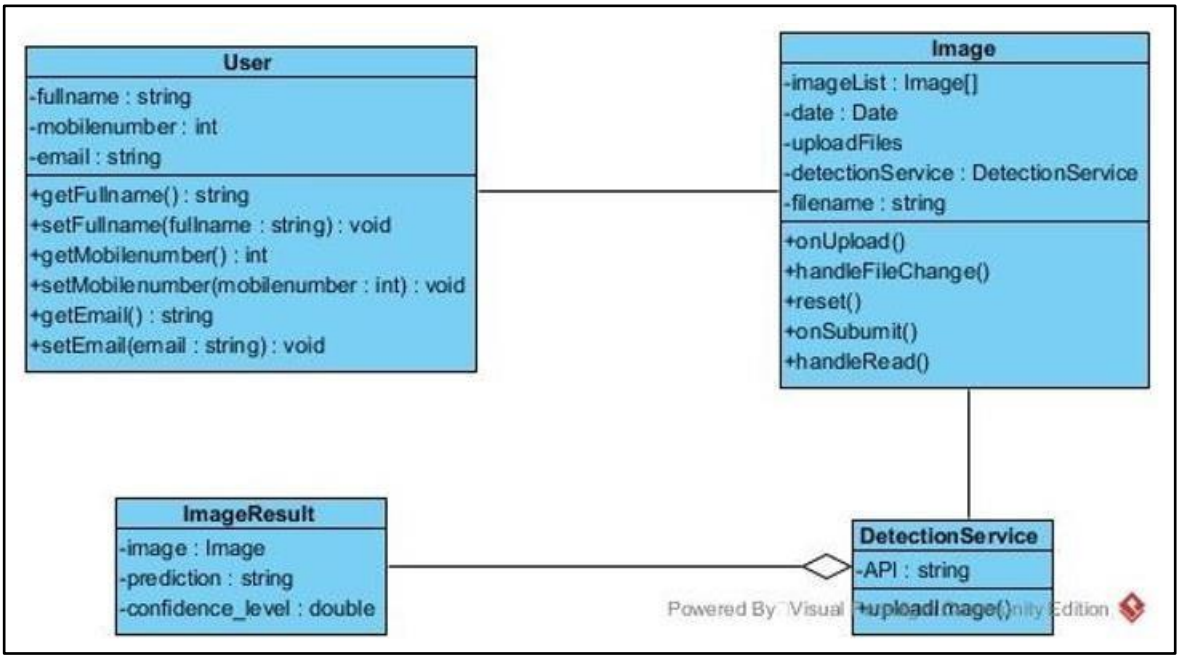


Figure 2. 14 Client Side-class diagram

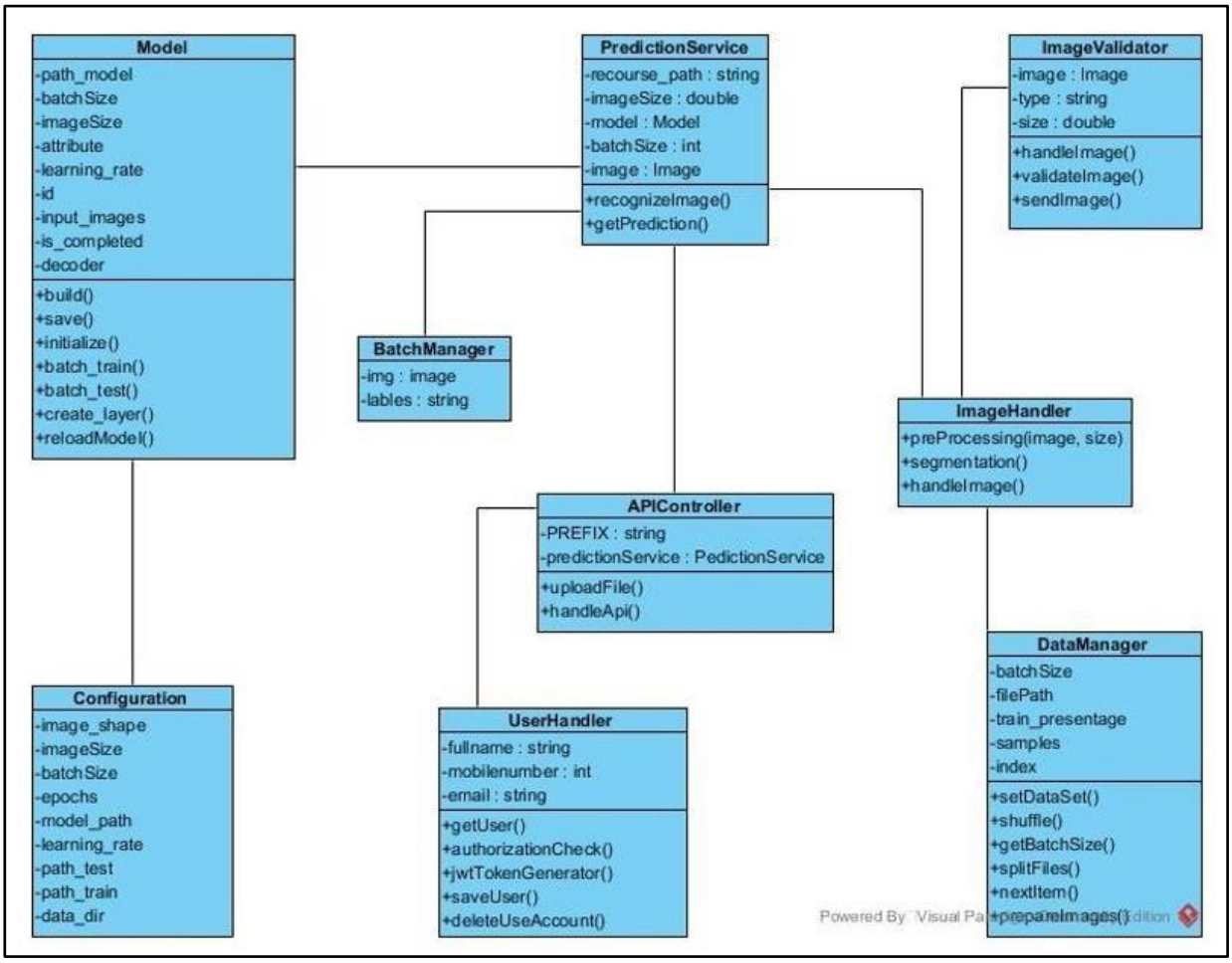


Figure 2. 15 Server side-class diagram

2.4.3 Sequence diagrams

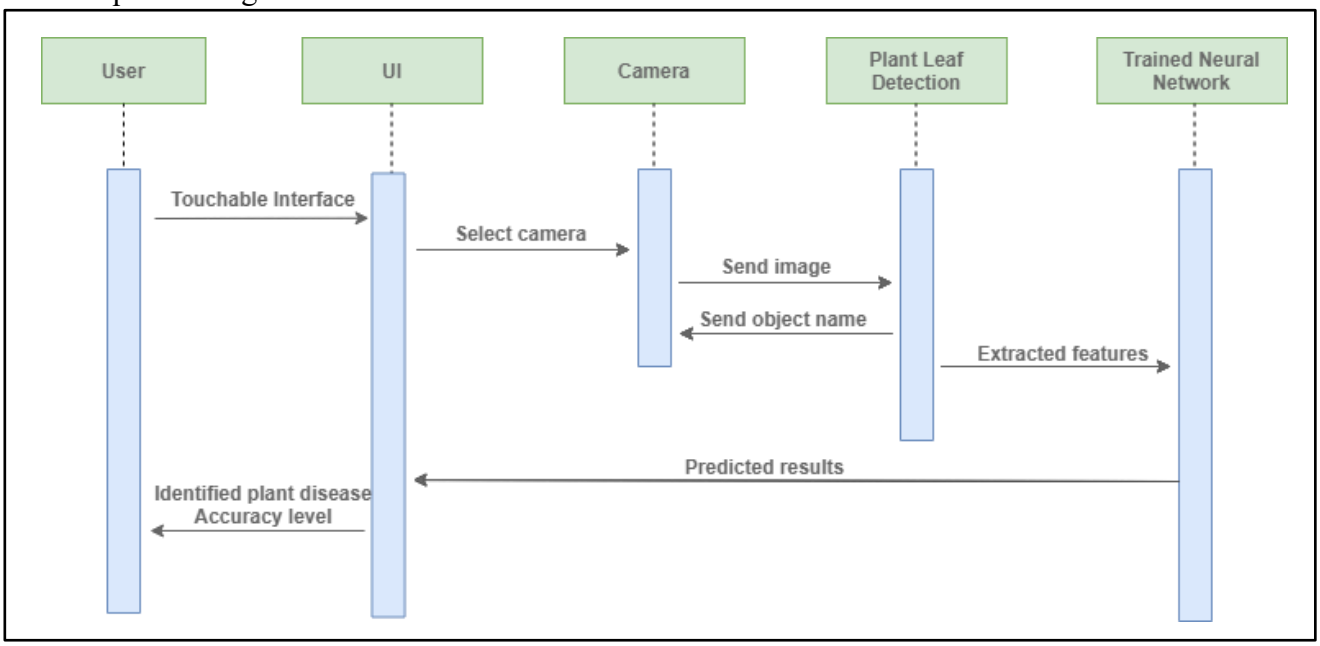


Figure 2. 16 Sequence diagram-main flow

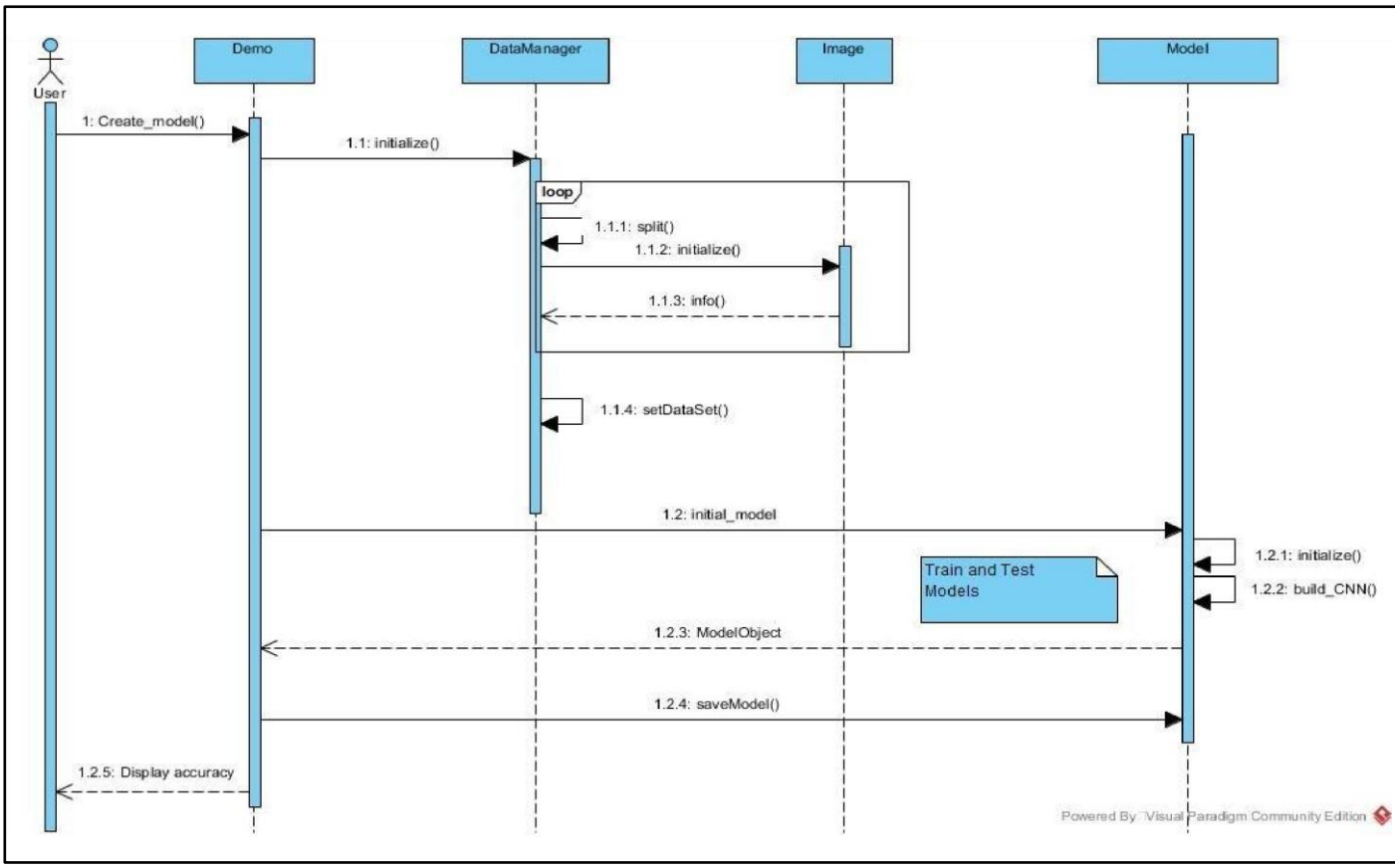


Figure 2. 17 Sequence diagram-main flow

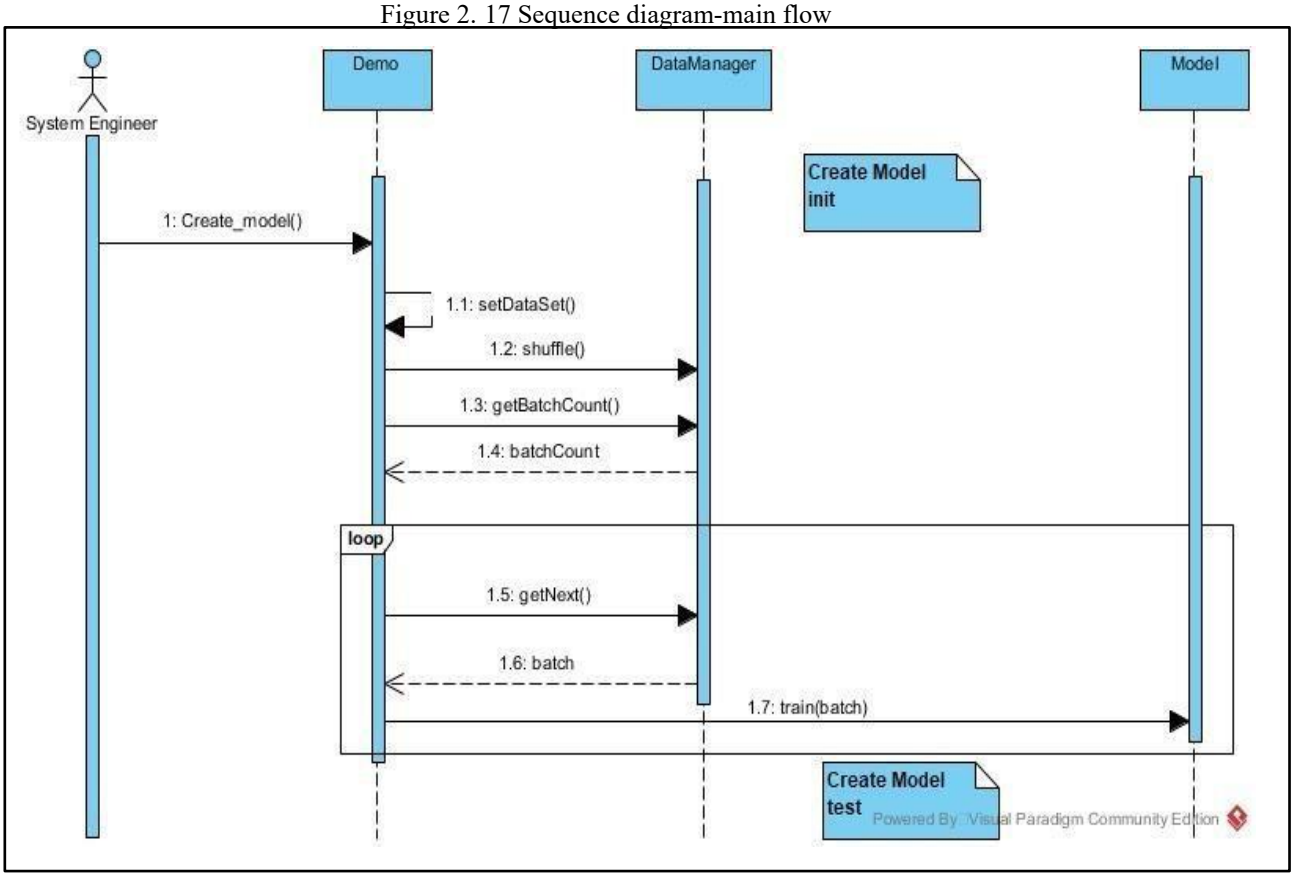


Figure 2. 18 Sequence diagram-model flow

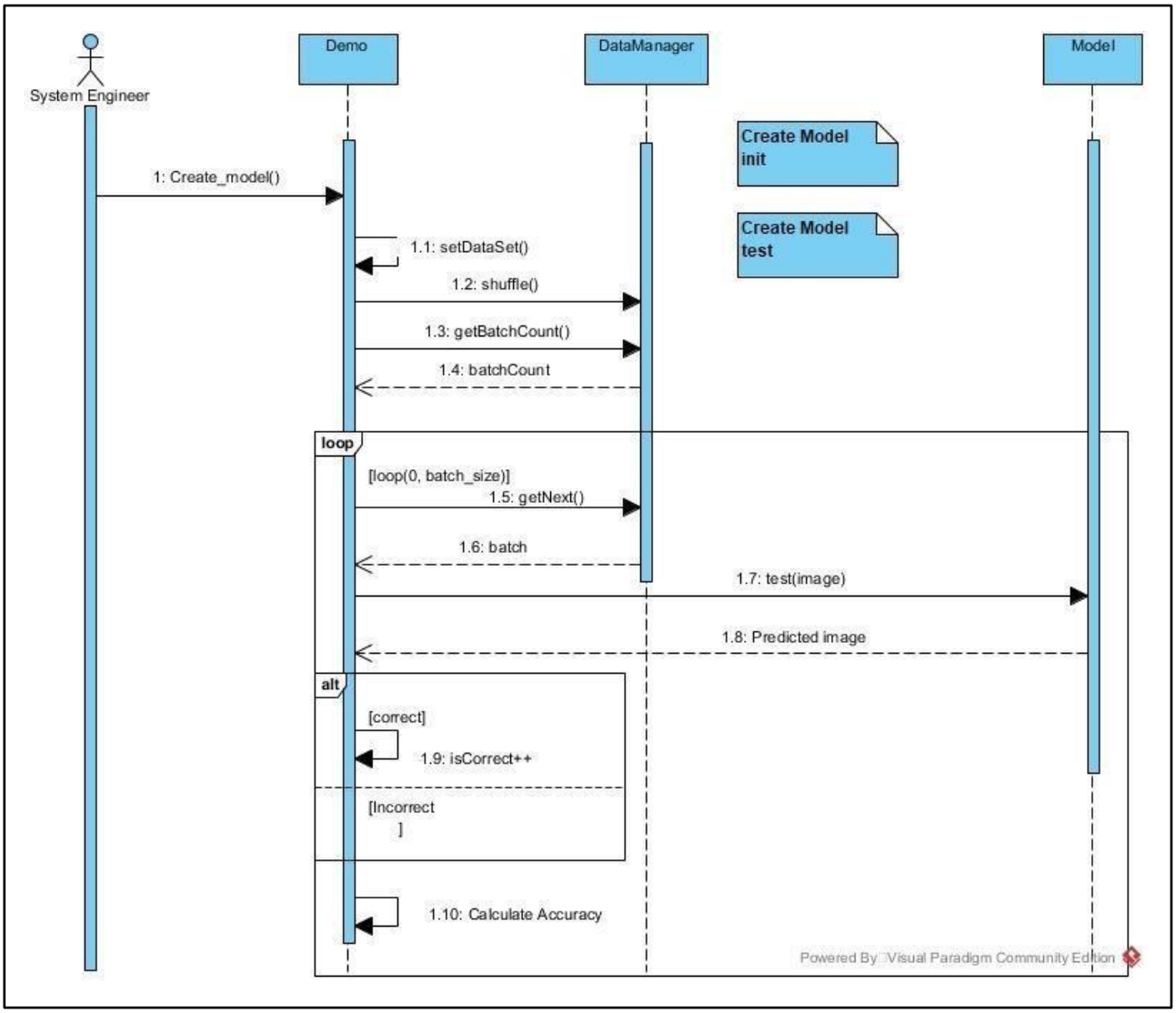


Figure 2. 19 Sequence diagram-train model

2.4.4 The LeNet Architecture (1990s)

LeNet was one of the very first convolutional neural networks which helped propel the field of Deep Learning. LeNet architecture was used mainly for character recognition tasks such as reading zip codes, digits, etc.

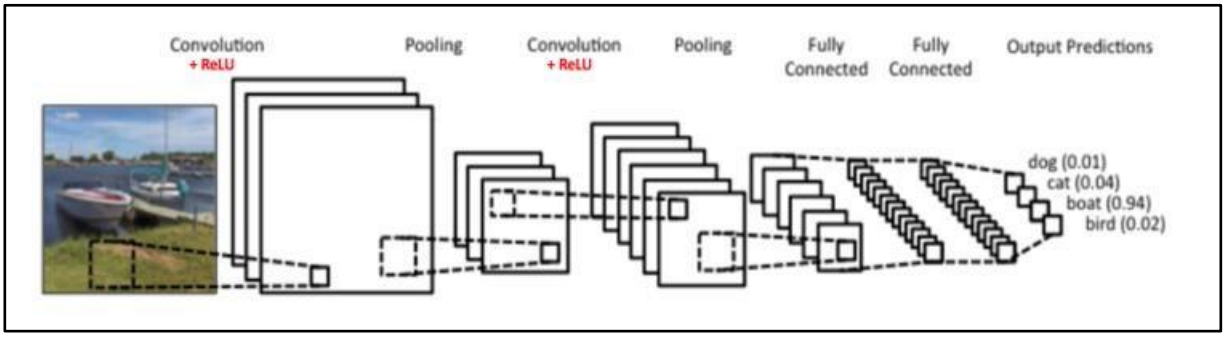


Figure 2. 20 Basic setup of a Convolutional Neural Network

The Convolutional Neural Network in Figure 5 above is similar in architecture to the original LeNet and classifies an input image into four categories: dog, cat, boat or bird (the original LeNet was used mainly for character recognition tasks). As evident from the figure above, on receiving a boat image as input, the network correctly assigns the highest probability for boat (0.94) among all four categories. The sum of all probabilities in the output layer should be one There are four main operations in the ConvNet shown in figure above:

1. Convolution
2. Non-Linearity (ReLU)
3. Pooling or Sub Sampling
4. Classification (Fully Connected Layer)

2.5 RELU FUNCTION

ReLU stands for Rectified Linear Unit and is a non-linear operation. Its output is given by:

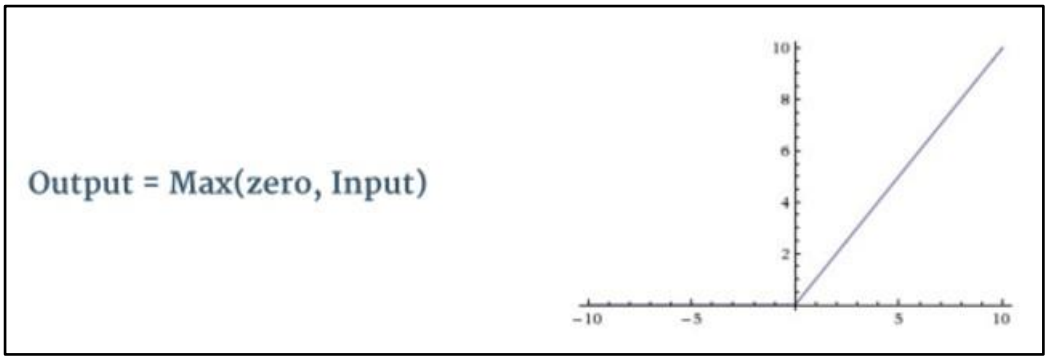


Figure 2. 21 RELU function

ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear (Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

2.5 TENSORFLOW

TensorFlow is an open-source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

CHAPTER 3: SOFTWARE REQUIREMENTS SPECIFICATION

3.1 INTRODUCTION

This document is a Software Requirement Specification (SRS) for the application that predicts citrus, corn and tomato plant diseases and provides the remedies for the predicted disease. SRS will be used for the extensions. The document is prepared as per the standard IEEE standard for Software Requirement Specification.

3.1.1 Purpose

The purpose of this document is to present a detailed description of the Plant disease detection and diagnoses using deep learning. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. Through this document, the workload needed for development, validation and verification will ease. To be specific, this document is going to describe functionality, external interfaces, performance, attributes and the design constraints of the system which is going to be developed.

3.1.2 Technologies to be used

3.1.2.1 Programming languages

1. Python

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. 2. Flask

Flask is a micro web framework written in Python. It is classified as a micro framework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing thirdparty libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. There are many extensions for object-relational mappers, form validation, upload handling, various open authentication technologies and several common frameworks related tools.

3.1.2.2 Libraries

1. TensorFlow

   TensorFlow is an open-source software library for machine learning across a range of tasks, and developed by Google to meet their needs for systems capable of building and training neural networks to detect and decipher patterns and correlations, analogous to the learning and reasoning which humans use.

2. Keras

   Keras is an open-source neural network library written in Python. It is capable of running on top TensorFlow or Theano. Designed to enable fast experimentation with deep neural networks, Keras focuses on being minimal, modular and extensible.

3. OpenCV

   OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.

   The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

   OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. The library is used extensively in companies, research groups and by governmental bodies.

3.1.2.3 Libraries used in backend app

1. Flask SQL alchemy

Using raw SQL in Flask web applications to perform CRUD operations on database can be tedious. Instead, Sqlalchemy, a Python toolkit is a powerful OR Mapper that gives application developers the full power and flexibility of SQL. Flask-Sqlalchemy is the Flask extension that adds support for Sqlalchemy to your Flask application.

2. Flask Restful

   REST is acronym for REpresentational State Transfer. It is an architectural style, and an approach to communications that is often used in the development of Web services. REST web services are a way of providing interoperability between computer systems on the Internet. REST-compliant Web services allow requesting systems to access and manipulate textual representations of Web resources using a uniform and predefined set of stateless operations.

3. Marshmallow

   Flask-Marshmallow is a thin integration layer for Flask (a Python web framework) and marshmallow (an object serialization/deserialization library) that adds additional features to marshmallow, including URL and Hyperlinks fields for HATEOAS-ready APIs.

4. Pillow (PIL)

   Python Imaging Library (abbreviated as PIL) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

5. Flask JWT Extended

   JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.

   Basically, rather than adding user information in every request we make to our server, we can send an encrypted token which is compact & self-contained.

   - Compact: Because of their smaller size, JWTs can be sent through a URL, POST parameter, or inside an HTTP header. Additionally, the smaller size means transmission is fast.

- Self-contained: The payload contains all the required information about the user, avoiding the need to query the database more than once.

3.2 OVERALL DESCRIPTION

This section gives background information about specific requirements of the product to be developed in brief. Although we will not describe every requirement in detail, this section will describe the factors that affect the final product.

The product has four main components: a mobile client which is the mobile application, a backend server which is made using flask in our case and a trained model which would predict the disease in the given image and a database which would store the treatment to various diseases and preventive measures which could be taken in order to prevent those diseases to happen in future.

The first component is the mobile client which is basically a mobile application where the end user needs to login and after login he can click image of the leaf of the diseased crop and the app will tell him the disease and the cure for that particular disease.

In order to predict the disease, the mobile application communicates with the backend flask server. The app sends the clicked image to the server and the trained model which has been deployed on the server runs on the received image and predicts the disease. After predicting the disease, the server responds back to the mobile application but before the it checks into the database.

The database consists of the treatments for various diseases. The server searches the treatment for the predicted disease in the database and responds back to the app giving it the predicted disease and its chemical and biological cure. Along with that it also gives the preventive measures that can be taken so that the disease does not infects the crops in future.

The overall functioning of the entire system is show in below figure:

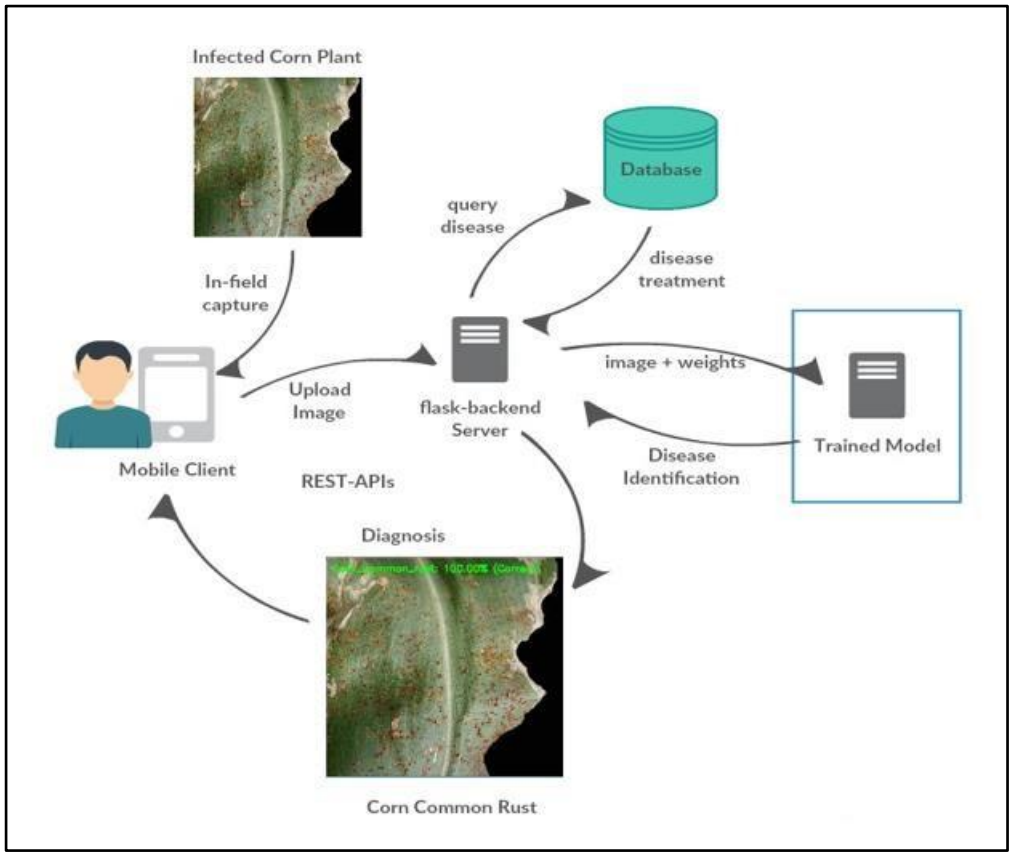


Figure 3. 1 Overall functioning of the entire system

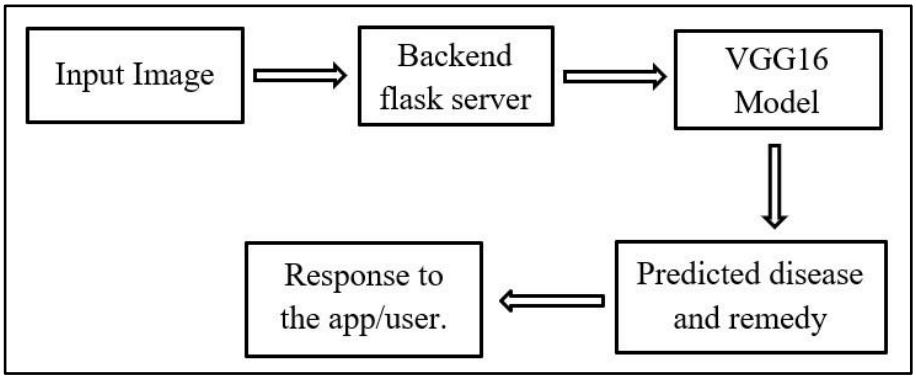


Figure 3. 2 Data Flow

3.2.1 Product Perspective

This software product is eventually intended to automate the process of plant disease detection and diagnosis. The product will be used and deployed in the smartphone devices as a mobile app. The product would enable the farmer to detect the plant disease and cure it without any human expert intervention which would prevent loses that occur due to failure in identification of diseases or wrong medication given to the crops or disease not detected at the correct time.

3.2.2 Product Functions

This system allows users to use functionalities which have been explained above in the introduction. Required functionalities of the product can be summarized in four categories; mobile client management, server management, trained model management and updation, database management and updation. Overall description of the requirements can be found below:

First the data from the camera is read as inputs and stored for processing.

Then the received image/frame is resized and sent to the backend flask server to predict the disease.

In the server which has the deployed model the data is passed through a convolutional net which extracts the required features.

Then the extracted features are given as inputs to the fully connected neural network.

The network then predicts the disease using the features extracted.

The server responds back to the app giving it the information of the disease and the required remedy and preventive measures.

3.2.3 User Characteristics

- Users of this mobile application can be done by any farmer or any normal person having a smartphone device. The app could be installed in the mobile therefore anyone can use it.
- Target users may have some knowledge about the disease to no knowledge at all.
- End-user is educated with basic smartphone usage.

3.2.4 Assumptions

- Users have android installed in their device.
- User have access to internet so that app can communicate with the server.
- User have the knowledge the where the occurrence of the disease is more prominent.
- User should provide a good image of the leaf that he thinks is infected by some disease.

3.3 SPECIFIC REQUIREMENTS

With this section and later, we will describe the requirements of the product in detail. Basically, we will categorize requirements in three which are namely external interface requirements, functional requirements and non-functional requirements. Except non-functional requirements, requirements of the product will be detailed under this section with brief information.

3.3.1 External Interface Requirements

- User Interface Requirements: User friendly interface is required so that user can easily login or register in the application and click images of the diseased leaf easily and without any problems and know the disease (if any) infecting his crops.
- Hardware Interface Requirements: There are no special hardware interface requirements. The user just needs to have a smartphone device having the installed app.
  Communication Interface Requirements: Communication is involved between the app and the backend server. This communication is done with the help of REST API's wherein the server and app communicate through http status codes like 404 is for not found etc. The term REST stands for representational state transfer. They use GET, PUT, POST, DELETE methods to retrieve, modify or delete data.

3.3.2 Functional Requirements

Input: The input to the system would be the image captured by the smartphone device of the user.

Output: The output of the system would be the detected plant disease and its chemical and biological cure along with some information about the disease and the preventive measures against that particular disease.

3.3.3 Performance Requirements

- Application should take the image as an input where the image can be a real-time one taken in the field or an image stored in the gallery and the opening time for application and camera should be as less as possible.
- Application should be able to predict plant disease from images captured in real time on the field without any significant delay.

- Generated application should be responsive in design and should not involve much delay in performing its functions.

3.3.4 Non-Functional Requirements

1. Security – Generated application should ensure safety of user login credentials and upload data of user.
2. Usability - The system shall be easy to use and understand. User will only need to open the mobile app and upload the image of the diseased leaf. Rest all the prediction work would be done on the backend side.
3. Maintainability – Component driven development and modular structure shall ensure maintainable code.
4. Portability – The mobile application should be able to run on any android platform without any bugs or difficulties. Also, the app should be less in size so that storage should not be an issue for the users.

CHAPTER 4: METHODOLOGY

4.1 RESEARCH METHODS

The main two research methods that are subsequently verified to deliver the most suitable and feasible methodology in order to perform research and analysis of this scope. Primary research contains of engaging and insightful ways to gather data and important information, such as discussions, surveys, assessments, questionnaires and etc. And secondary-research focuses solely on existing literature, it has been examined and studied for feasibility studies through reviewing different kinds of journal articles, journals, books and other reliable sources. In addition, tools such as tutorials and existing community websites such as Stack Overflow and GitHub have also been used.

4.1.1 Primary research

Performing primary research is important in order to get along with the system specifications and a good design of a system because system requirement specifications and client needs can be establishing through proper primary research. This form of research was necessary to determine client needs, problems that they currently face and how much impact would hold if the project succeed.

Through surveys that performed with farmers and locals highly important and critical information was discovered in this project. This information can be emphasized as given below.

Surveys were performed using paper surveys since some of the old farmers in the area didn't know how to use google docs. The surveys consist with a set of questions which were divided in to two sections addressing the problem and how their attitude towards current systems and how the users of the proposed system would think that the application would be advantageous.

Physical copies of following questionnaires were shared with 21 people in the area including mainly targeted participants in order to obtain necessary information.

All 21 participants have responded to the survey questions and answers were successfully recorded. Following snap shots will represent one of the responded survey questionnaires by the participants.

Obtained information through the survey can be divided in to sections and visualized as below.

Are you farming?

As following pie chart emphasized 100% of the participants do cultivation according to the survey reports. Hence it is proved that the appropriate participants has been selected regarding the completion of the survey.

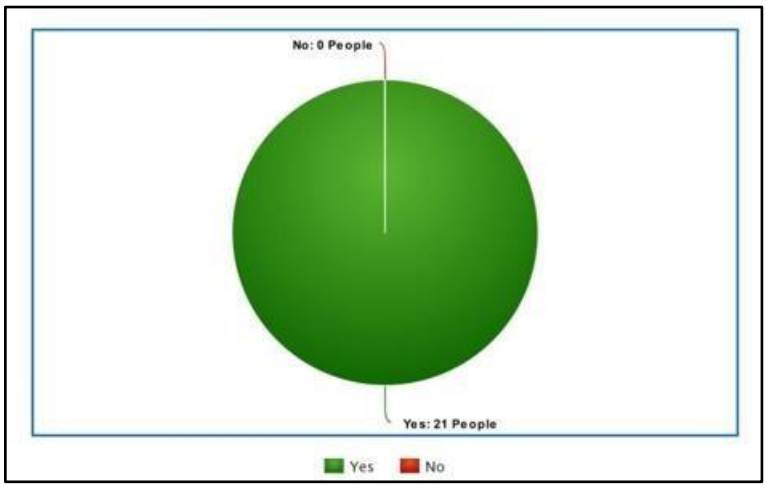


Figure 4. 1 Pie chart 1

4.1.1.1 Internet facility

Above 97% had a stable internet facility in their homes which is really necessary when deciding system requirement specifications. Even though others didn't have a stable internet connection all of them had access to internet through their smartphones.

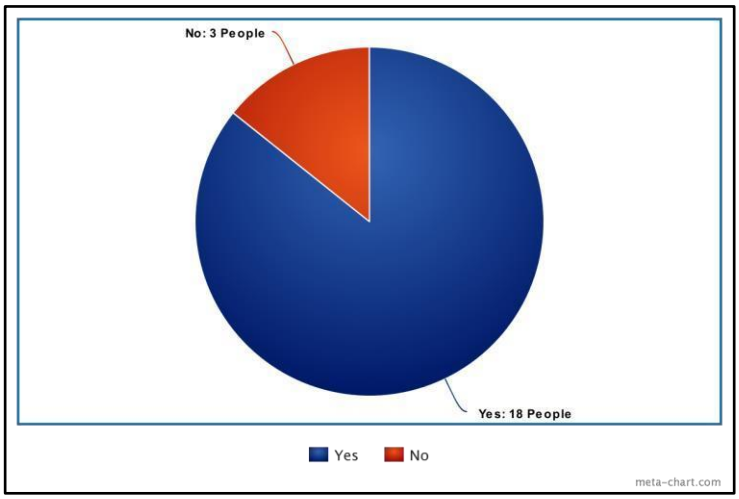


Figure 4. 2 Pie chart 2

4.1.1.2. Mostly used smart device

According to the survey 94% used a smartphone more often which is necessary when deciding the platform for the application.

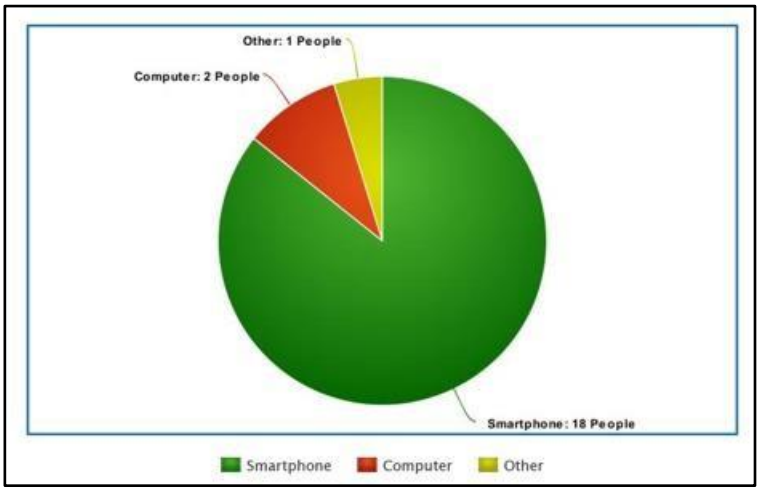


Figure 4. 3 Pie chart 3

4.1.1.3 Crop losses because of plant diseases

Over 87% of the participants have faced to crop losses because of various kinds of plant diseases.

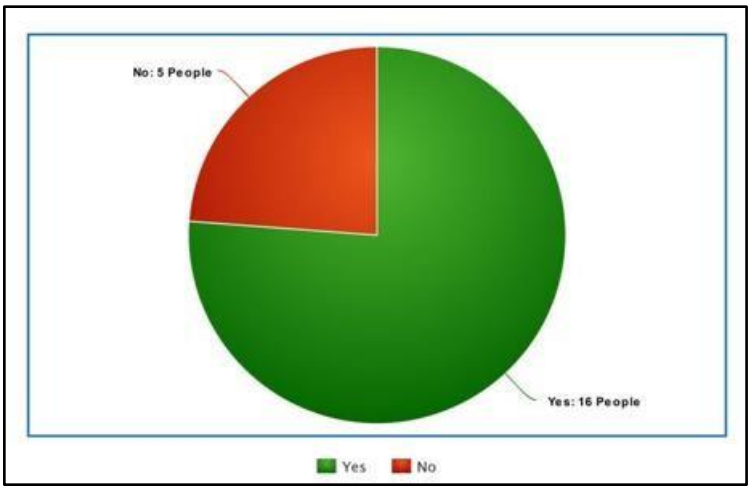


Figure 4. 4 Pie chart 4

4.1.1.4 Reasons to take help

As following pie chart emphasized most of the participants take expert help in recognizing plant diseases because of their limited knowledge on plant diseases and their symptoms.

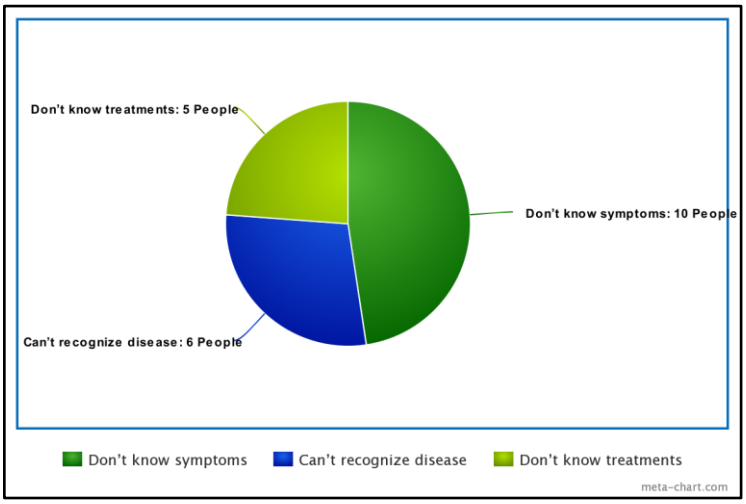


Figure 4. 5 Pie chart 5

4.1.1.5 Plant disease identification methods

According to reports of the surveys following pie chart represents the method that participants chose in order to identify plant diseases.

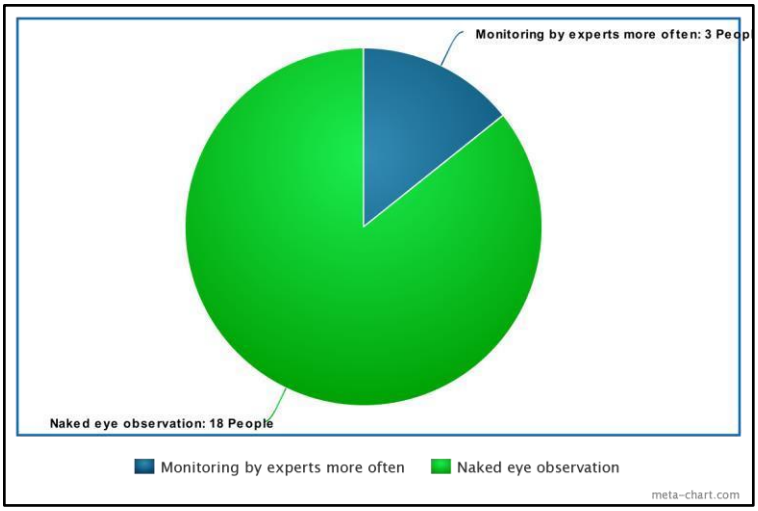


Figure 4. 6 Pie chart 6

4.1.2 Secondary Research

Analysis and review of current existing literature can be taken as the second-research of the project. For the dissertation aspect of this project, journal articles and books were mainly used to gather information. The journals and research papers have been selected as clearly and firmly viable sources providing precise knowledge with lawful authorization.

- Detection and Classification Technique of Yellow Vein Mosaic Virus Disease in Okra Leaf Images using Leaf Vein Extraction and Naive Bayesian Classifier by Dhiman Mondal
- SVM Classifier Based Grape Leaf Disease Detection by Pranjali B. Padol
- Detecting Jute Plant Disease Using Image Processing and Machine Learning By Zarreen Naowal Reza1
- Detection of unhealthy plant leaves using image processing and genetic algorithm with Arduino by Arya M S
- Cucumber Disease Detection Using Artificial Neural Network by Ms. Pooja Pawar
- Detection And Measurement of Paddy Leaf Disease Symptoms using Image Processing by R.P.Narmadha
- Recent Machine Learning Based Approaches for Disease Detection and Classification of Agricultural Products by Mukesh Kumar Tripathi
- Detection of Leaf Diseases and Classification using Digital Image Processing by R.Meena Prakash
- Plant Disease Detection Using Machine Learning By Shima Ramesh
- Android Application on Plant Disease Identification using Tensorflow by Naveen Chandra Gowda
- Mobile Platform Implementation of Lightweight Neural Network Model for Plant Disease Detection and Recognition by Anton Louise P. de Ocampo
- Importance of Plant Disease, Scope and Objective of Plant Pathology by Shraddha Karcho
- Neural Networks vs. Random Forests – Does it always have to be Deep Learning? by Prof. Dr. Peter Robbach

4.2 DEVELOPMENT METHODOLOGY

Considering the limited timeframe of the project and limited resources, it is compulsory to follow a carefully guided plan. Following components should be consider when deciding a software development methodology is being decided.

- The size of the venture
- How explicit necessities are

- How much the client will need to change requirements?
- How large the development team is?
- Due date

The project should be designed and developed with in a very limited time framework and the design, development, testing and deployment all done by a single person, it is important to follow a well-chosen software development approach in order to reduce the possibility of failing to fulfil the planned project deadlines. As stated above the project will be developed by one person with in a very tight time framework it is anticipated that there will be regular meetings with the supervisor when designing the program to ensure that the system is implemented in compliance with standards and specifications. Following table presents a comparison between traditional SDLC methodologies and agile methods.

	Traditional Methodologies(Ex: waterfall)	**Agile methods**
Adaptability to change	Change sustainability	Change adaptability
Development approach	Predictive	Adaptive
Development Orientation	Process oriented	People oriented
Project Size	Large	Small/Medium
Planning Scale	Long term	Short term
Management Style	Command and control	Leadership and collaboration
Learning	Continuous learning while development	Learning is secondary to development
Documentation	High	Low

Figure 4. 7 Traditional Vs Agile methodologies

Waterfall methodology is not a fit to the project since the time frame is very limited, waiting each step to finish is not practical and ideal. RAD methodology can be excluded because of the limited resources and hence only one person is developing the system. Considering all the most widely used SDLC methodologies Agile's SCRUM methodology would be a perfect fit for this project.

Below given is the detailed description of the working of the entire system.

The first step is for the user to login into the smartphone application or register himself first if he has not already registered. The screenshots of the login and register screen are shown below:

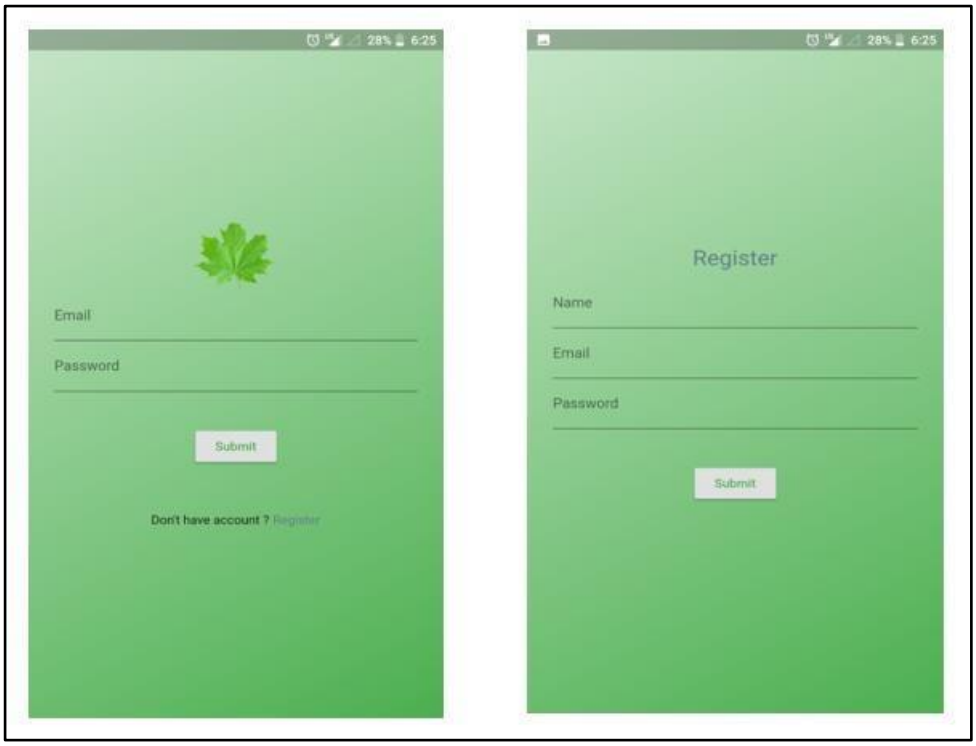


Figure 4. 8 Login and Register screen of the application

After logging into the app, the user enters into the home screen whose screen shot is shown below. The user can open the home page where he can see his last photo taken, upload a new image of the leaf of the plant he suspects to be diseased and he can open the library which consists of all the diseases which can be identified by the application.

The upload screen is shown in the below figure. It consists of two options one to click the image from the camera itself and the other option is to upload an image which is already present in the gallery. But first the user needs to select the type of plant for which he is uploading the image i.e., citrus, corn or tomato. The screenshot of the uploading screen is shown below:

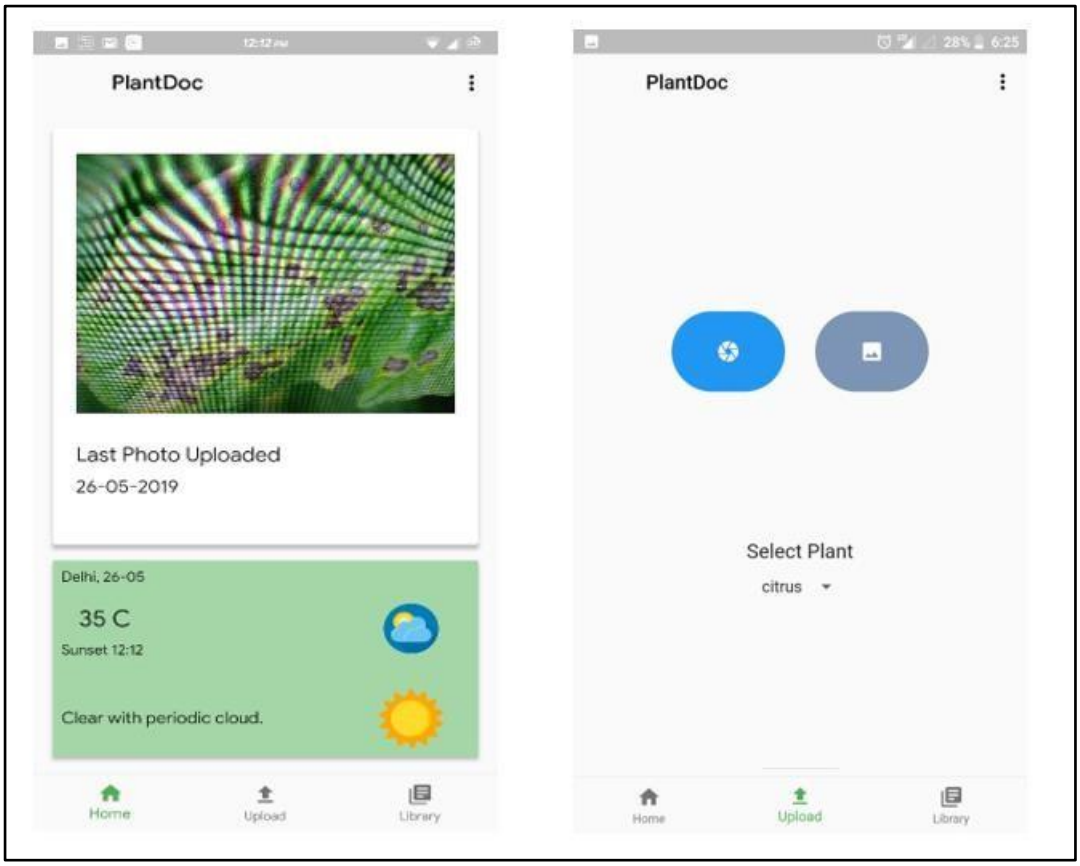


Figure 4. 9 Screenshot of Home and Upload screen

After selecting or clicking the image of the plant that, the user suspects, is diseased, the app uploads that particular image onto the server. The communication between the server and the mobile application is done using the REST api's.

The directory structure of the server on which the image is being uploaded is shown below. The backend app is deployed online on Linode server. Along with that a screen shot of the server responding to the requests made by the mobile application in which the user who is making the request can be seen and what the request is being made can be seen. The model which is used to predict the disease i.e., the VGG16 model is deployed on the backend server. The backend server is also responsible for communicating with the backend database which is stored using PostgreSQL so that it can retrieve the cure for the disease that is predicted by the model. The cure consists of the chemical control, biological control along with extra information about the disease i.e., the symptoms and the preventive measures to be taken so that the disease does not occur in the future.

CHAPTER 5: MODEL PROPOSED

After going through all the papers, websites and tutorials, we have followed a process for the implementation of the project. First step will be to get the image dataset from the fields. So, we have collected the image dataset and we have also considered the point that image should be centered so that it will be easy for the better model training. Image is needed to go through the process of the pre-processing. Feature extraction and detection process is followed and then last stage will be image classification.

5.1 IMAGE ACQUISITION

The most important part for the given task is to acquire the real time field images because a person would like to use the application in the real fields. So, we need to have those background so that model is well trained. The example of field images is shown below figure:

Figure 5. 1 Field Images

5.2 IMAGE PRE-PROCESSING

Next task was to pre-process the image to remove the unwanted noise from the images and we need to augment the images to provide different image rotation, zoom-in, zoom-out etc. To make multiple images with different rotation, size, ratio etc., we have used the pillow library in OpenCV. We have mainly focused on the effected part of the leaf. It helps the model to learn certain features which is required. Then few images are also introduced in the training dataset which is not so much focused on the diseases. The sample of these images is shown in fig:

Figure 5. 2 Pre-Processed Images

5.3 DISEASE SEGMENTATION AND FEATURE EXTRACTION

Now, we have our training images and we have pre-processed the image. So, our next task will be to train the model. So, our strategy will be as follows: we will only instantiate the convolutional part of the model, everything up to the fully-connected layers. We will then run this model on our training and validation data once, recording the output (the "bottleneck features" from the VGG16 model: the last activation maps before the fully-connected layers) in two numpy arrays. Then we will train a small fully-connected model on top of the stored features. We have used the convolutional layer and each layer contains different kernel of size 3 X 3 and when the whole image is convoluted, it gives the feature map of the image and in the next layer, we have max polling layer of size 2 X 2 which extract the feature from feature map and then gives the extracted feature map and now new convolution layer is used on this feature map with different kernel to obtain the final feature map. The reason why we are storing the features online rather than adding our fully-connected model directly on top of a frozen convolutional base and running the whole thing, is computational efficiency. Running VGG16 is expensive, especially if you're working on CPU, and we want too only do it once. Note that this prevents us from using data augmentation. So, we have augmented the data using OpenCV

and python before training of the model. The overall VGG16 model is shown below in the figure 11.

5.4 CLASSIFICATION

Next task is to flatten the feature map to obtain the neuron like structure of neural network, now hidden layer of 256 neurons is inserted with the ReLU activation function and next layer is dropout layer with 50% neurons have disconnected randomly. The final output layer with the six neurons is used to classify the images. The final layer will contain only six neurons which shows the output with SoftMax activation function.

First neuron talks about the class canker, second neuron talks about greening, third neuron talks about the gummosis, fourth neuron talks about healthy, fifth neuron talks about leaf Miner and last one tells about the Lemon Buttery in the citrus plant diseases. The corresponding output is predicated based upon the learning.

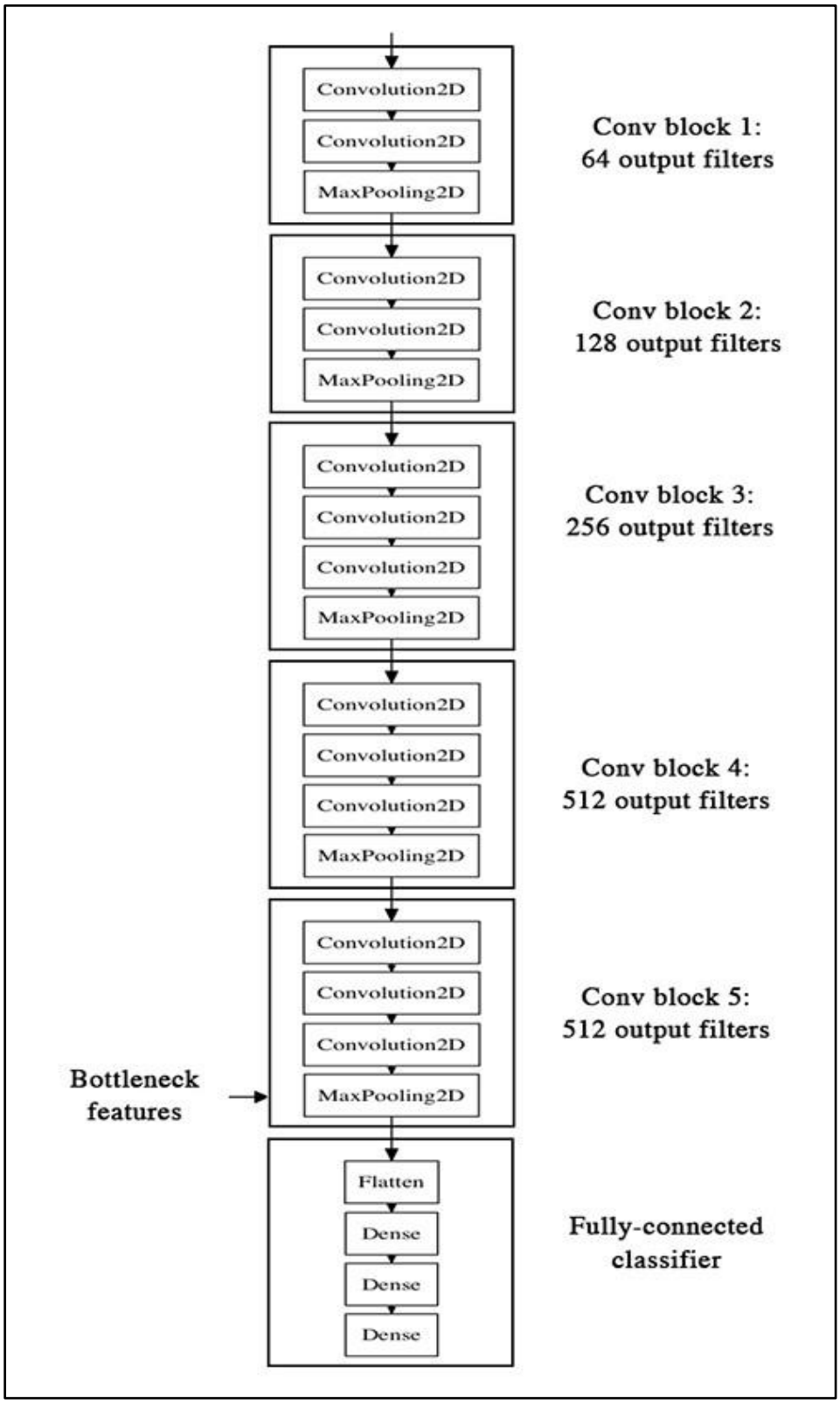


Figure 5. 3 Model Architecture

CHAPTER 6: EXPERIMENTS AND RESULTS

6.1 EXPERIMENTS

In this section, we'll determine performance evaluation of the convolutional neural network (CNN) architectures, we'll create the general structure of the model, we'll identify hyperparameters and then further tune those hyperparameters such as dropout etc. and then we'll train the model on google colab.

Performance metric: We use accuracy as our performance metric. We'll save the model weights after the epoch. In our dataset, number of samples is distributed among the six classes. The accuracy metric plot and loss metric plot are shown in fig.

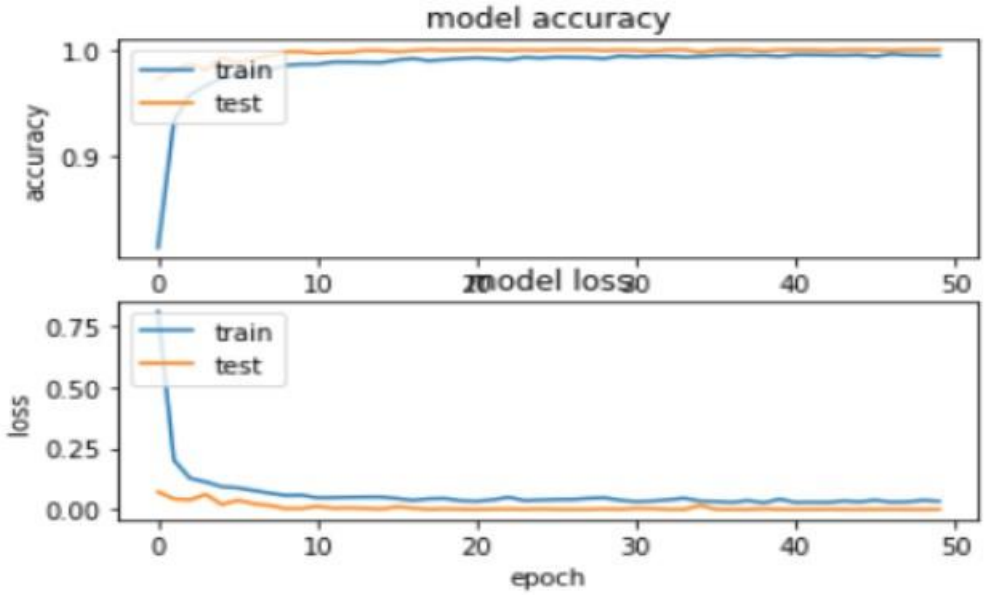


Figure 6. 1 Accuracy and loss metric plot

6.1.1 Loss function

We use Categorical cross entropy as our loss function. We have total six classes. If we have two classes, then we have used binary cross entropy. Since there are multiple classes, so this loss function is actually a log loss version of this case.

6.1.2 Basic Architecture

We have used the transfer learning. if we have trained the VGG model from scratch, they it may perhaps take weeks so we have used the transfer learning method and used CNN architectures of VGG16. We remove the top three layers of the original convolutional neural network architectures. We flatten the output of the last layer among the remaining layers of the

architectures. The remaining convolutional layers carry the pre-trained weights from ImageNet classification task. As a result, these architectures are already well-trained with basic image features. We add three densely connected layers with ReLU activation function and one dense layer with SoftMax activation function on top. This topmost layer with SoftMax activation function contains six nodes for the six classes.

6.1.3 Hyperparameter Tuning

Hyperparameters are not trainable. But these are the most important parameters. These parameters are learning rate, batch size, epoch etc. We fix their values at the start of training. If we have given the perfect values to these parameters, then model can work unexceptionally well. We have used epochs value as 50 and batch size of 32. So, we mainly tune two hyper parameters. Dropout rate is the first hyper parameter that we tune. A dropout rate of 0.25 means that our model will ignore 25% of the neurons of the previous layer at random in each epoch. It helps to reduce overfitting. We add dropout layer after each dense layer except for the last layer. We test our models with dropout rate of 0.50. The second hyper parameter that we tune is Learning rate. Learning rate determines how fast our model weights are adjusted in order to get to the local or global minima of our loss function. A learning rate of 0.0001 is a good choice many times.

6.1.4 Optimizer

We use Adaptive Moment Estimation (Adam) for training our models. It is a method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients, Adam also keeps an exponentially decaying average of past gradients. It combines the advantages of two other extensions of stochastic gradient descent - AdaGrad and RMSProp.

6.1.5 Experimental Environment

We have used the google colab gpu. Since, it is free of cost and it have GPU named 1xTesla K80 which is having 2496 CUDA cores that can compute 3.7GHz and also have RAM size of 12GB (11.439GB Usable) GDDR5, VRAM.IT contains CPU with these specifications 1xsingle core hyper threaded i.e. (1 core, 2 threads), Xeon Processors @2.3Ghz (No Turbo Boost), 45MB Cache. RAM is 12.6GB available and disk size of 320 GB available. For every 12hrs or

so Disk, RAM, VRAM, CPU cache etc., data that is on our allotted virtual machine will get erased. This colab can be used on the laptop with intel i5 processor and with a good internet speed. It hardly took 1 hour for training process. Some Training log has been shown below in fig:

6.2 FLASK BACKEND IMPLEMENTATION- PYCHARM

The backend of the application is developed using flask framework which was discussed in the previous chapters. HTTP Request handling and machine learning model loading process happens through backend of the application.

```
@app.route('/api/predict', methods=['POST'])
def get_prediction():
    target = os.path.join(APP_ROOT, 'leaves/')

    if not os.path.isdir(target):
        os.mkdir(target)

    file = request.files.get('file')

    filename = file.filename
    destination = '/'.join([target, filename])

    file.save(destination)

    result = output_prediction(filename)

    return jsonify(result)
```

Above method has used to handle the POST request that comes from the client side when the image is being uploaded. /api/predict is the rout the system uses to upload the image form the client side. The image is getting saved in given path and pass the image to the output prediction method which is responsible for returning the prediction result of the supplied image. Finally, the method returns the server response as a json object with the status of the request.

6.2.1 Configure image file

```
def config_image_file(_image):
    img = cv2.imread(_image)
    img = cv2.resize(img, (224, 224))
    img = img / 255
    return img
```

config_image_file method is used to handle the pre-processing step whenever an image is uploaded to get the prediction, Using OpenCV library which was discussed earlier the given image will be reshaped, resized and rescaled before feed in to the model to get the prediction. The model is expecting 224 x 224 shaped RBG image to process. Lastly the method returns the image array.

6.2.2 Predict the image

```
def predict_image(image):
    probabilities = model.predict(np.asarray([image]))[0]
    class_idx = np.argmax(probabilities)

    return {img_classes[class_idx]: probabilities[class_idx
],
            'index': class_idx}
```

In the above the method takes a parameter called image which is an array, then it passes the image to the model and get the probabilities. Img_classes are a list of currently supported plant diseases in the system it checks the output of the above function and maps the output with the index of the image classes array. And finally, the method returns a dictionary which contains disease name, confidence and the index of the disease (in order to make the frontend manipulations easily).

6.2.3 Return the prediction result

Output prediction method is implemented in order to return the final outcome of the prediction and all-important details regarding the prediction.

```
def output_prediction(filename):
    _image = f"./leaves/{filename}"

    print(_image)
    img_file = config_image_file(_image)
    prediction = predict_image(img_file)

    disease = list(prediction.keys())[0]
    confidence = list(prediction.values())[0
]   index = list(prediction.values())[1]

    result = {
        "id": str(index),
        "disease": disease,
        "confidence": str(confidence)
    }
    return result
```

The method itself invokes the config_image_file method and it passes the pre-processed image to the model by invoking predict_image method. As stated above predict_image method returns a dictionary which includes the name of the crop diseases (predicted class) and the confidence. The method creates another dictionary including id for the identifying purposes in the client application side.

6.3 TENSORFLOW TRAIN THE MODEL- GOOGLE COLAB

6.3.1 Importing necessary libraries

In order to train the deep learning model, it should use various kinds of libraries which makes the implementation a lot easier. All of the libraries were selected after considering alternative libraries and comparing them each other.

```
import tensorflow as tf
import tensorflow_hub as hub
import os
from tensorflow.keras.layers import Dense, Flatten, Conv2D
from tensorflow.keras import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.image import imread
```

Mostly used libraries in the project are represented in the above image such as TensorFlow that is used as the main library that the deep learning model has been trained on, Matplotlib which has used in graphing, NumPy that has used to handle complex array objects in the application and etc.

6.3.2 Loading dataset

First the dataset is uploaded categorizing according to the 10 classes that is supported in this classification as a zip file to the google drive. Google Colabs can integrate the google drive separately using below block of code.

```
from google.colab import drive
drive.mount('/content/drive')
```

After the google drive is successfully mounted the zip file was unzipped by using following code.

```
# Unzip data set
!unzip "drive/My Drive/_leaves.zip"
```

Unzipped data will be temporary stored in the Colabs run time environment and will be deleted within 12 hours (in the free version) for this development it was enough.

6.3.3 Splitting dataset

In order to split the unzipped data, set in to 3 categories namely training, validation and testing data that are used to implement and demonstrate the accuracy of the deep learning model a python library called “split-folders” has been used. Since the library is not pre-installed in to the Colabs

environment it should installed in to the Colabs environment then following code snippet should be used in order to split dataset appropriately.

```
import splitfolders

splitfolders.ratio('/content/_leaves/', output="/content/data", seed=1337, ratio=(.6, .3, .1))
```

It will automatically create 3 folder structures in the given directory and split the dataset to a given ratio. In this project the dataset is divided into 6:3:1 ratio after many attempts have been done.

```
import time
import os
from os.path import exists

def count(dir, counter=0):
    "returns number of files in dir and subdirs"
    for pack in os.walk(dir):
        for f in pack[2]:
            counter += 1
    return dir + " : " + str(counter) + "files"

print('total images for training :', count(train_dir))
print('total images for validation :', count(val_dir))
print('total images for test :', count(test_dir))
```

```
total images for training : /content/data/train : 6000files
total images for validation : /content/data/val : 3000files
total images for test : /content/data/test : 1000files
```

Above image shows the number of images which has included in each data split directory. The function count is defined in order to output the number of images in each directory.

6.3.4 Loading labels

Following code snippet is used to load the prediction class labels form a pre-written json file which is stored in the google drive.

```
import json

with open('/content/drive/My Drive/diseases1.json', 'r') as f:
    cat_to_name = json.load(f)
    classes = list(cat_to_name.values())

print (classes)
print('Number of classes: ', len(classes))
```

There are 10 different plant disease names that are included the given json file.

6.3.5 Image pre-processing

Since the dataset has been created manually, all images in the dataset are not in same dimensions and in order to keep the original image data the images aren't resized manually. As the image pre-processing step of the project all the images should be resized in to a one common size.

```
#Declaring image size and shape
IMAGE_SHAPE = (224, 224, 3)
```

All images of the dataset are reshaped to the above declared shape. 224 x 224 height and width along with the color channels. From the first two numbers of the image shape represents the width and the height of the image while the last number represents the 3 RBG color channels.

```
validation_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1/255)
validation_generator = validation_datagen.flow_from_directory(
    val_dir,
    shuffle=False,
    seed=42,
    color_mode="rgb",
    class_mode="categorical",
    target_size=IMAGE_SHAPE[:2],
    batch_size=BATCH_SIZE)
```

From the pre-processing method which is an inbuilt method of Keras library which comes inbuilt from TensorFlow 2.0 the dataset has been pre-processed. Both training and validation datasets are pre-processed rescale property is used to transform every pixel value from range (0 to 255) to (0 to 1) because some of the images in the dataset are high pixel range and some of the images in the dataset are low pixel range, rescaling each image to a same rang 0-1 will make images contributes more evenly to the total loss.

6.3.6 Model Training

TensorFlow library is used to define the different types of model layers and to set various parameters in training the model.

```
model = Sequential()


model.add(Conv2D(filters=32, kernel_size=(3,3),input_shape= IMAGE_SHAPE,padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=64, kernel_size=(3,3), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=128, kernel_size=(3,3),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(filters=256, kernel_size=(3,3),padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

By trying many times of attempts above layers and parameters were selected considering the model accuracy and the total loss of the model. The model consists with 4 convolutional layers with 32, 64, 128 and 512 filter sizes while using “ReLU” as the activation function. Maxpooling layers (2, 2) are used to reduce the number of total weights of the model. 2 fully connected layers are used in the above model.

```
model.add(Flatten())

model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.6))

model.add(Dense(train_generator.num_classes))
model.add(Activation('softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

As described in the above image it is used a 0.6 of dropout layer in order to prevent the model is being overfitting. The value 0.6 is selected after building the model using different values for the dropout layers. “Adam” optimizer has used as the optimizer for this classification. Below image provides a summary of a defined model.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 224, 224, 32)      896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 112, 112, 64)      18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 56, 56, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 56, 56, 128)       73856
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 28, 28, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 28, 28, 512)       590336
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 14, 14, 512)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 100352)            0
_________________________________________________________________
dense_2 (Dense)              (None, 512)               51380736
_________________________________________________________________
activation_2 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130
_________________________________________________________________
activation_3 (Activation)    (None, 10)                0
=================================================================
Total params: 52,069,450
Trainable params: 52,069,450
Non-trainable params: 0
```

The model is trained under total number of 52,069,450 parameters.

```
results = model.fit(
                      train_generator,
                      epochs=10,
                      steps_per_epoch=train_generator.samples//train_generator.batch_size,
                      validation_data=validation_generator,
                      validation_steps=validation_generator.samples//validation_generator.batch_size,
                      callbacks=[tensorboard]
                     )
```

Model was trained for 10 epochs and for the callbacks it is used TensorBoard in order to represent the model status in a proper manner.

6.3.7 Saving the model

Once the model was trained completely. The model has been saved to the google drive.

```
model.save('/content/drive/My Drive/model_89.h5')
```

6.4 REACT NATIVE MOBILE APPLICATION IMPLEMENTATION– VISUAL STUDIO CODE

The client-side mobile application is developed using React Native which is a modern cross platform mobile application developing framework that was discussed in the previous chapters. The mobile application provides two options for a user to upload images; Select from gallery or take picture from the device camera and upload it to the server.

6.4.1 Select from Gallery Option

The method is implemented to handle the image selecting from the gallery process. Whenever the method executes the component pops up and open the device gallery where the user will be able to select an image and upload it.

```
pickFromGallery = async () => {
    this.setState({
      isVisible: false
    });
    const { granted } = await Permissions.askAsync(Permissions.
CAMERA_ROLL);
    if (granted) {
      let data = await ImagePicker.launchImageLibraryAsync({
        mediaTypes: ImagePicker.MediaTypeOptions.Images,
        allowsEditing: true,
        aspect: [4, 3],
        quality: 1 //1 means high quality
      });
      if (!data.cancelled) {
        let newFile = {
          uri: data.uri,
          type: `test/${data.uri.split('.')[1]}`,
          name: `test.${data.uri.split('.')[1]}`
        };
        this.onUpload(newFile);
      }
    } else {
      Alert.alert('You need to give permissions');
    }
  };
```

The method pickFromGallery is an Asynchronous method first it checks the user whether the application has the correct permission rights in order to use the device camera and the device gallery, If the permission rights are given the method continuous by popping up the device gallery. ImagePicker has various properties which defines the behaviour of the image picker. media Types is used to view only the image formatted files from the device gallery,

allowEditting allows the user to perform basic image editing operations before confirming the image that is chosen such as rotate, flip and crop image and aspect property defines the dimensions of the selecting image, finally quality property is represent the quality of the image that is being uploaded after the confirmation here it has chosen 1(High) quality in order to keep the original image information as much as possible.

6.4.2 Pick from Camera Option

The method is used to handle the expo camera that provides the component to renders a preview of the device's front or back camera.

```
pickFromCamera = async () => {
    this.setState({
      isVisible: false
    });
    const { granted } = await Permissions.askAsync(Permissions.
CAMERA);
    if (granted) {
      let data = await ImagePicker.launchCameraAsync({
        mediaTypes: ImagePicker.MediaTypeOptions.Images,
        allowsEditing: true,
        aspect: [4, 3],
        quality: 1 //1 means high quality
      });
      if (!data.cancelled) {
        let newFile = {
          uri: data.uri,
          type: `test/${data.uri.split('.')[1]}`,
          name: `test.${data.uri.split('.')[1]}`
        };
        this.onUpload(newFile);
      }
    } else {
      Alert.alert('You need to give permissions');
    }
  };
```

First the method checks the device's user permissions to open the device camera and notify the user if the permissions are blocked. The method itself is an Asynchronous method which handles inside functionalities asynchronously. ImagePicker's launchCameraAsync method is responsible for open the device camera and capture the image. Then the captured image will be sent to the

server by a POST request and onUpload method is implemented to handle the image uploading process of the application.

6.4.3 Image Uploading Process

```
onUpload = async (image) => {
    const data = new FormData();
    data.append('file', image);

    try {
      this.setState({
        isLoading: true
      });
      const response = await axios.post(
        `${BASE_URL}api/validate`,
        data
      );
      if (response) {
        this.setState({
          isLoading: false
        });
        if (response.data.status > 0) {
          this.getDiseasePredection(image);
        } else {
          alert(
            'No Plant Leaf Detected, Please provide an image of a plant leaf'
          );
          console.log('No Plant Detected');
        }
      }
    } catch (err) {
      console.log(err.message);
    }
  };
```

The selected image will be converted and assigned to formData typed object in order to send to the server-side application. IsLoading variable is initialized and bound to the component level state in order to handle the spinner process status whenever an image is being uploaded and once the image uploading process is done. Http requests are handled through a library called axios and first the method calls api/validate endpoint to validate the image basically this will return a Boolean value along which determines whether the uploaded image is a plant leaf or not. If the uploaded image is a plant leaf then the method will invoke the getDiseasePrediction method to make the next api request to get the plant disease detection response from the machine learning model else it notifies the user with an alert saying the supported images is not a plant leaf.

6.4.4 Get Prediction

```
getDiseasePredection = async (image) => {
  const data = new FormData();
  data.append('file', image);
  try {
    this.setState({
      isLoading: true
    });

    const response = await axios.post(
      `${BASE_URL}/api/predict`,
      data
    );

    if (response.data) {
      this.setState({
        isLoading: false
      });

      const { disease, confidence, id } = response.data;

      let _disease = disease.split('__').join(' ');

      this.props.navigation.navigate('PredictionScreen', {
        _disease,
        confidence,
        id
      });
    }
  } catch (err) {
   alert(err.message)
  }
};
```

The method getDiseasePrediction takes the selected image as a parameter and send the image to the server-side application through api/predict api which returns 3 response data namely disease, confidence and the id that will be sent to the PredictionScreen to user to view. If a network error or a server error occurs the user will be notified about the error using an alert.

6.4.5 View Predictions

The render method in PredictionScreen.js is responsible for display the predicted results in an attractive manner that user can understand. RenderImage method handles the predicted disease image view, renderDescription handles the predicted

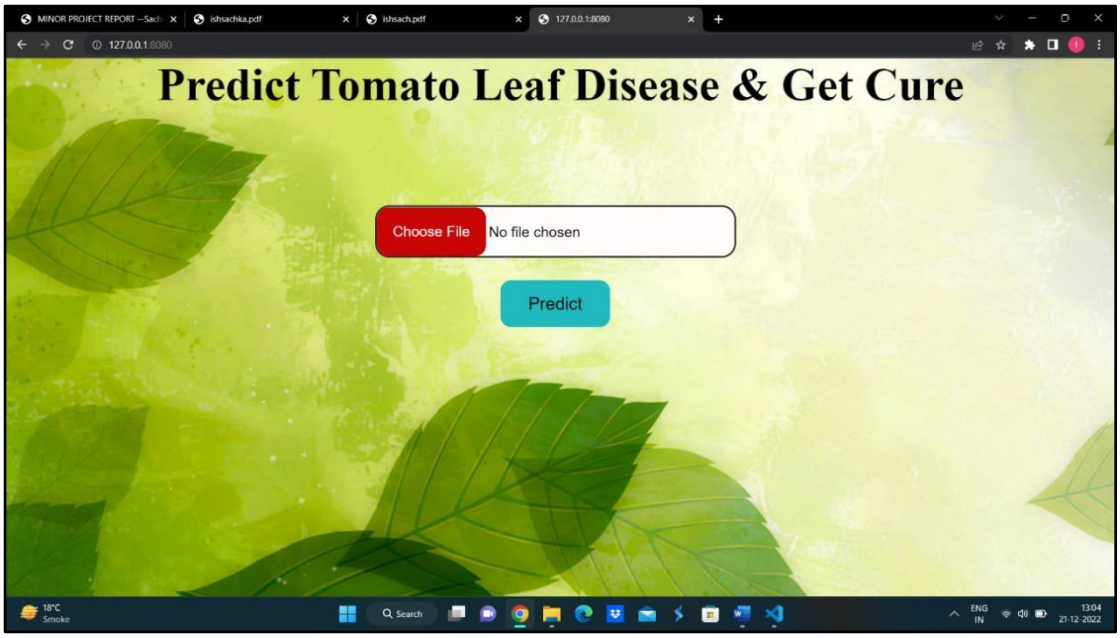
disease description and the render method renders all necessary fields populated with the retrieved data from the previous API response.

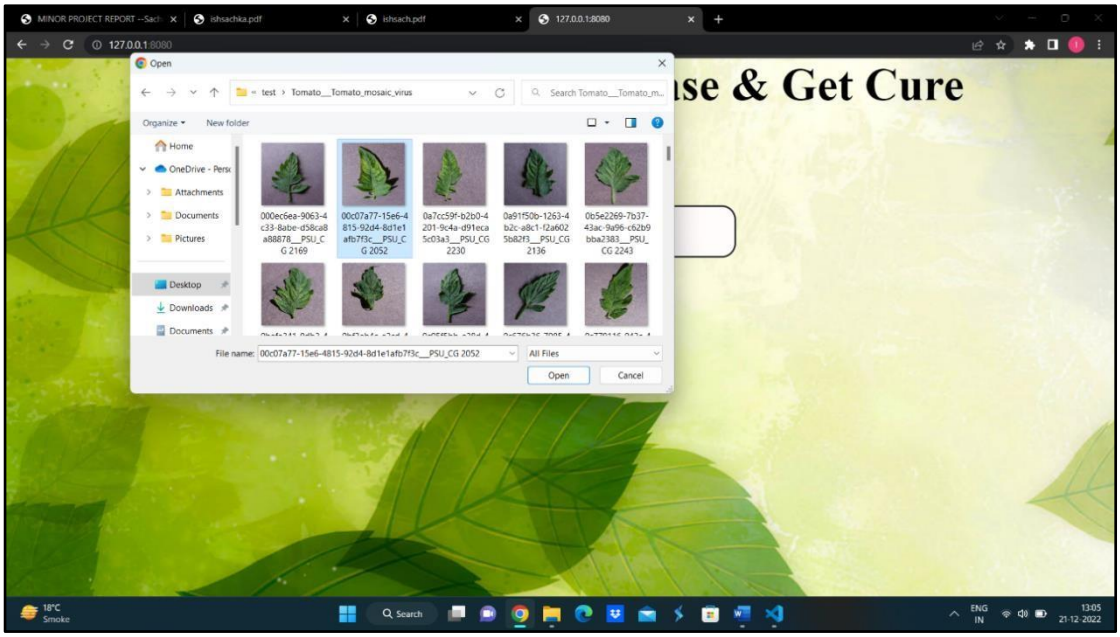
```
renderDescription = (_id) => {
   if (!data.length == 0) {
     let item = _.filter(data, { id: parseInt(
_id) });
     return <Text>{item[0].description}</Text>;
   } else {
     return <Text>No records found</Text>;
   }
 };
```

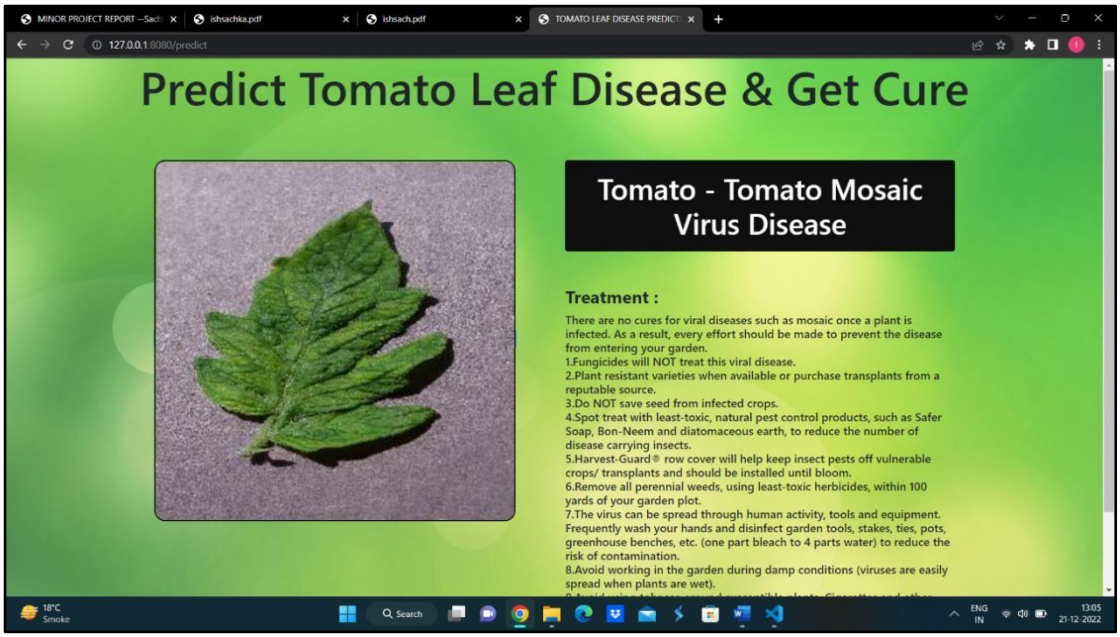
The method is responsible for rendering the description of the predicted disease.

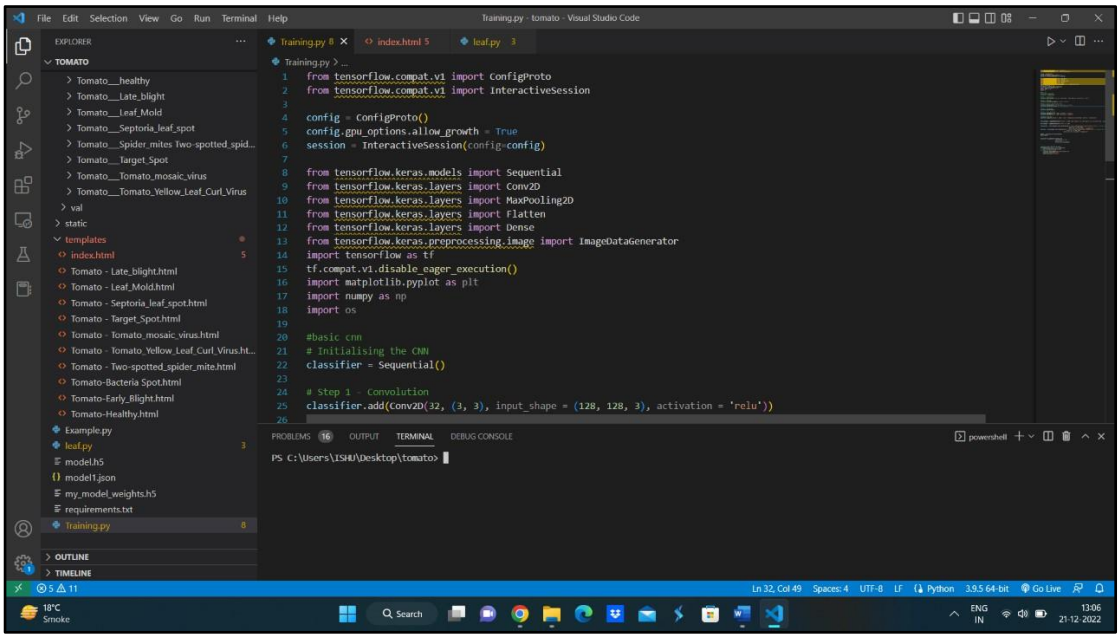
```
renderImage = (_id) => {
   if (!data.length == 0) {
     let item = _.filter(data, { id: parseInt(
_id) });

     return (
       <Image
         source={item[0].image}
         resizeMode='cover'
         style={styles.plantImage}
       ></Image>
     );
   } else {
     return <Text>No image found</Text>;
   }
 };
```

The method returns and image of the predicted crop disease by using returned id from the API response. If there's an issue displaying the image then the method will return a text saying No image found.


Predict Tomato Leaf Disease & Get Cure
Choose File
No file chosen
Predict


Open
Organize
New folder
Home
OneDrive - Persc
Attachments
Documents
Pictures
Desktop
Downloads
Documents
File name: 00c07a77-15e6-4815-92d4-8d1e1afb7f3c__PSU_CG 2052
All Files
Open
Cancel


MINOR PROJECT REPORT --Sach
ishsachika.pdf
ishsach.pdf
TOMATO LEAF DISEASE PREDICT
127.0.0.1:8080/predict
Predict Tomato Leaf Disease & Get Cure
Tomato - Tomato Mosaic Virus Disease
Treatment :
There are no cures for viral diseases such as mosaic once a plant is infected. As a result, every effort should be made to prevent the disease from entering your garden.
1.Fungicides will NOT treat this viral disease.
2.Plant resistant varieties when available or purchase transplants from a reputable source.
3.Do NOT save seed from infected crops.
4.Spot treat with least-toxic, natural pest control products, such as Safer Soap, Bon-Neem and diatomaceous earth, to reduce the number of disease carrying insects.
5.Harvest-Guard® row cover will help keep insect pests off vulnerable crops/ transplants and should be installed until bloom.
6.Remove all perennial weeds, using least-toxic herbicides, within 100 yards of your garden plot.
7.The virus can be spread through human activity, tools and equipment. Frequently wash your hands and disinfect garden tools, stakes, ties, pots, greenhouse benches, etc. (one part bleach to 4 parts water) to reduce the risk of contamination.
8.Avoid working in the garden during damp conditions (viruses are easily spread when plants are wet).
18°C
Smoke
Search
ENG
IN
13:05
21-12-2022


File Edit Selection View Go Run Terminal Help
Training.py - tomato - Visual Studio Code
EXPLORER
TOMATO
Tomato__healthy
Tomato__Late_blight
Tomato__Leaf_Mold
Tomato__Septoria_leaf_spot
Tomato__Spider_mites Two-spotted_spid...
Tomato__Target_Spot
Tomato__Tomato_mosaic_virus
Tomato__Tomato_Yellow_Leaf_Curl_Virus
val
static
templates
index.html
Tomato - Late_blight.html
Tomato - Leaf_Mold.html
Tomato - Septoria_leaf_spot.html
Tomato - Target_Spot.html
Tomato - Tomato_mosaic_virus.html
Tomato - Tomato_Yellow_Leaf_Curl_Virus.ht...
Tomato - Two-spotted_spider_mite.html
Tomato-Bacteria Spot.html
Tomato-Farly_Blight.html
Tomato-Healthy.html
Example.py
leaf.py
model.h5
model1.json
my_model_weights.h5
requirements.txt
Training.py
Training.py 8
index.html 5
leaf.py 3
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
config = ConfigProto()
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
tf.compat.v1.disable_eager_execution()
import matplotlib.pyplot as plt
import numpy as np
import os
#basic cnn
# Initialising the CNN
classifier = Sequential()
# Step 1 - Convolution
classifier.add(Conv2D(32, (3, 3), input_shape = (128, 128, 3), activation = 'relu'))
PROBLEMS 16
OUTPUT
TERMINAL
DEBUG CONSOLE
powershell
PS C:\Users\ISHU\Desktop\tomato>
OUTLINE
TIMELINE
Ln 32, Col 49
Spaces: 4
UTF-8
LF
Python
3.9.5 64-bit
Go Live
18°C
Smoke
Search
ENG
IN
13:06
21-12-2022

CHAPTER 7: TESTING AND EVALUATION

7.1 TEST PLAN

Every test method is designed to detect a particular type of product malfunction. However, all test types are designed to achieve a common objective of "early identification of all defects before the product is released to the user".

The following types of testing should be performed prior to publication of the application.

- Unit test – Since the application will be developed feature by feature testing the smallest piece of verifiable software is essential.
- API test – Test the API's will be built for the backend of the application.
- Integration testing – Individual application components will be combined together and will be tested as a one component.
- System test – Conducted on a complete, integrated platform to check the compliance of the system with its particular requirements.

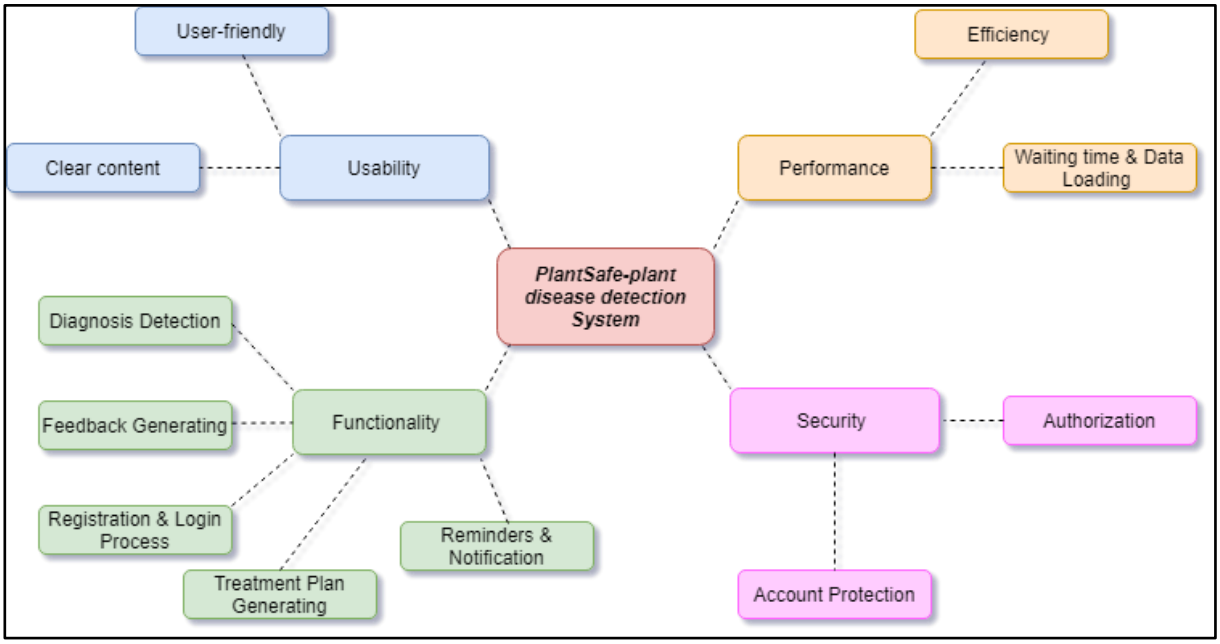


Figure 7. 1 Test and evaluation

7.2 TEST ENVIRONMENT

The application is designed to identify various kinds of plant diseases by analysing plant leaves. Since the app is designed to run on a mobile device the test environment will be a mobile device which has a camera and a proper internet connection.

7.3 TEST DATA

Images of diseased plant leaves. This can be previously collected diseased plant leaf images or the images that is capturing in the testing process.

7.4 TESTING TECHNIQUES

The application will be tested according to the following techniques;

- Validation testing
- Accuracy testing
- Functional testing
- Performance testing

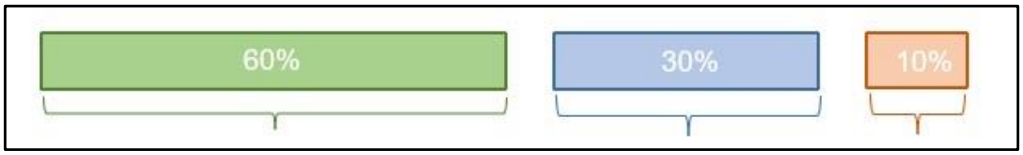
Functional tests are performed using application ui while validation testing, accuracy testing and performance testing are performed considering the machine learning aspect of the system. The validation testing of the system it checks how the validation dataset helps to train the model and the comparison between validation loss vs training loss and validation accuracy vs training accuracy. Accuracy testing was done evaluating the parameters and optimize the best training model for the expected system requirements. The system tests the accuracy of the testing data by observing confusion matrix. Performance was tested checking how the performance of the training model and how it predicts the correct output by using the model.

7.5 APPLICATION OF TESTS

7.5.1 Validation Testing

7.5.1.1 Split of the dataset

As it is mentioned in the above chapters the dataset is manually collected by visiting real time agricultural environments and taking images of the disease plant leaves. For training, validation and testing purposes the dataset was split in to 3 categories. The model is tested by using various kinds of data split ratios in the beginning and selected the support best for the model. Final selection of the data split ratio was 60% of the total dataset was used for the training purpose and 30% of the dataset was used as the validation data and remaining 10% was used as the test data to evaluate the model.



Training data (6000 images) Validation data (3000 images) Test data (1000 images)

7.5.1.2 Validation Accuracy and Validation Loss

For the validating purpose 3000 images out of 10,000 images were used, and all the images were organized and structured according to 10 different classes. After the model training process is completed it clearly displays the both validation accuracy and the validation loss of the model in each and every epoch.

Following image establishes the validation loss and the validation accuracy of the final model which was selected to use as the machine learning model in the project.

```
Epoch 1/10
93/93 [==============================] - 1103s 12s/step - loss: 1.7415 - accuracy: 0.3381 - val_loss: 1.2555 - val_accuracy: 0.5489
Epoch 2/10
93/93 [==============================] - 1094s 12s/step - loss: 1.1597 - accuracy: 0.5672 - val_loss: 0.8359 - val_accuracy: 0.6882
Epoch 3/10
93/93 [==============================] - 1090s 12s/step - loss: 0.7851 - accuracy: 0.7079 - val_loss: 0.6592 - val_accuracy: 0.7612
Epoch 4/10
93/93 [==============================] - 1087s 12s/step - loss: 0.6041 - accuracy: 0.7729 - val_loss: 0.5632 - val_accuracy: 0.7901
Epoch 5/10
93/93 [==============================] - 1087s 12s/step - loss: 0.4811 - accuracy: 0.8221 - val_loss: 0.4718 - val_accuracy: 0.8240
Epoch 6/10
93/93 [==============================] - 1088s 12s/step - loss: 0.3757 - accuracy: 0.8590 - val_loss: 0.4594 - val_accuracy: 0.8240
Epoch 7/10
93/93 [==============================] - 1088s 12s/step - loss: 0.2953 - accuracy: 0.8865 - val_loss: 0.4829 - val_accuracy: 0.8254
Epoch 8/10
93/93 [==============================] - 1090s 12s/step - loss: 0.2566 - accuracy: 0.9043 - val_loss: 0.4621 - val_accuracy: 0.8438
Epoch 9/10
93/93 [==============================] - 1091s 12s/step - loss: 0.2124 - accuracy: 0.9213 - val_loss: 0.4044 - val_accuracy: 0.8590
Epoch 10/10
93/93 [==============================] - 1088s 12s/step - loss: 0.1667 - accuracy: 0.9375 - val_loss: 0.3698 - val_accuracy: 0.8781
```

Figure 7. 2 Accuracy

Above image is the representation of validation loss and validation accuracy of the final model which has trained in by using 6:3:1 dataset split ratio and it is clearly visible that the validation accuracy has been incrementing from the 1st epoch to the 10th epoch, the validation accuracy of the 10th epoch is 87.81%. And observing validation loss of the model training, it has been decrementing from 1st epoch to 10th and that is a sign of a fine machine learning model because the validation loss becomes more and more little in each epoch and in the final epoch it only has 36.98%.

Following graph shows the validation accuracy and the validation loss of the training model over the number of epochs.

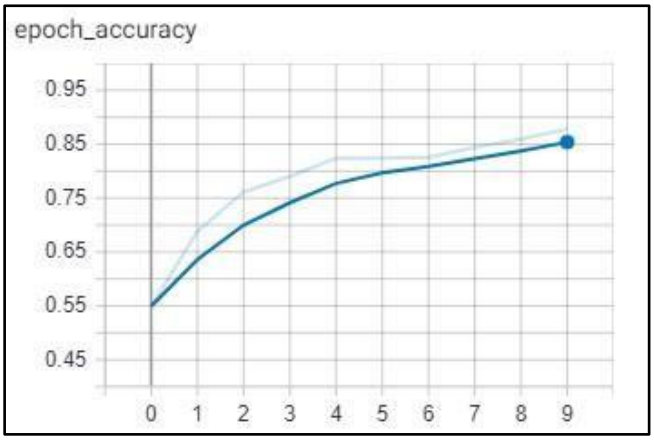


Figure 7. 3 Epoch accuracy

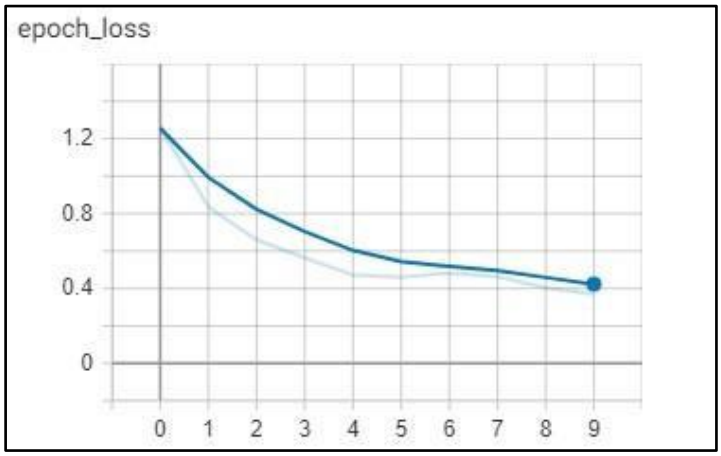


Figure 7. 4 Epoch loss

7.5.2 Accuracy Testing

7.5.2.1 Parameter Evaluation

In a well performing machine learning model the difference between actual predicted data and the training data should be minimum. In order to archive this, the machine learning model should be trained avoiding model overfitting and model underfitting. To avoid model overfitting and underfitting it should determine the suited parameters of the training model. To find that it should keep changing the parameter values and check the final output.

In this scenario the final model has been selected after 72 attempts. In an underfitting model the model can't perform well in both validation and training datasets while in an overfitting model it can perform very well on the training dataset but perform poorly on the validation dataset. In a good machine learning model the accuracy of the model should be increased in epoch and the loss should be decreased. The accuracy of the model can be depend on various parameters ex. Number of layers, data splitting ratio, filter values and etc. Following graphs represents the accuracy and the loss of the finally selected model over the number of epochs.

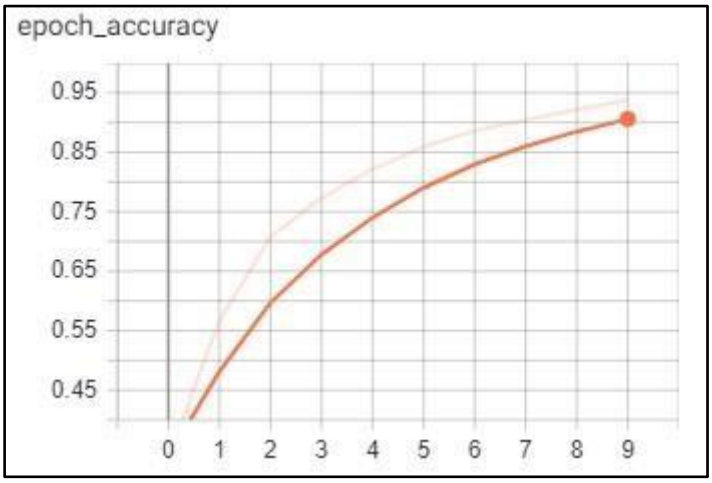


Figure 7. 5 Epoch Accuracy

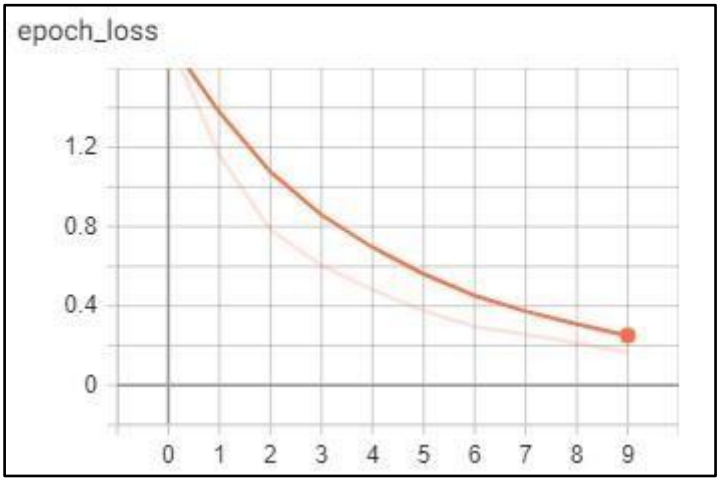


Figure 7. 6 Epoch loss

As the above graphs shows the accuracy of the training model has been increased from 1st epoch to 10th epoch and the accuracy of the model in the final epoch is 93.75% and the loss is getting decreased in each epoch, the loss of the 10th layer of the model is 16%. In order to grain these results, it has used 4 convolutional layers and 2 dense layers.

7.5.2.2 Confusion Matrix

By using a confusion matrix, a summary of predicted results can be visualized. It shows how many actual values and predicted values are existing in each class that the model will be predicted. Confusion matrix can only be used in a scenario which has two or more classes as outputs in a classification algorithm.

7.6 RESULTS

The result obtained is shown in table for various categories for the classes. The output of test image is shown below fig;

Figure 7. 7 Healthy image prediction

Classes	Canker	Greening	Healthy	Gummosis	Leaf Miner	Lemon Butterfly
Test Images	54	35	40	26	53	27
Correct	49	31	37	24	48	24
Accuracy	0.9074	0.8857	0.925	0.92307	0.90566	0.8888

Figure 7. 8 Results of Citrus crop

Classes	Grey Leaf Spot	Common Rust	Healthy	Norther Leaf Blight
Test Images	78	90	85	60
Correct	70	80	77	53
Accuracy	0.8974	0.9222	0.9058	0.8833

Figure 7. 9 Results of Corn crop

Classes	Bacterial Spot	Early Blight	Healthy	Late Blight	Leaf Mould
Test Images	45	73	70	65	43
Correct	39	68	64	58	39
Accuracy	0.8667	0.9315	0.9143	0.8923	0.9069

Figure 7. 10 Results of Tomato crop

Classes	Mosaic Virus	Septoria Leaf Spot	Two-Spotted Spider Mites	Target Spot	Yellow Leaf Curl
Test Images	63	52	61	45	75
Correct	57	47	56	41	69
Accuracy	0.9047	0.9038	0.9180	0.9111	0.9200

Figure 7. 11 Results of Grapes crop

CHAPTER 8: CRITICAL EVALUATION

8.1 EVALUATE OF THE DEGREE OF SUCCESS IN CARRYING OUT PROJECT

The success of a project mainly relies on several key factors such as the risk management of the project, well planned objectives, avoiding scope creep, clear requirements, time management and etc. In this project the requirements were decided according to the primary and the secondary research that followed by the developer. In order to keep the project with in the estimated time line there was three main milestones with in the timeframe of the project namely Project Specification Report submission where the project manager of the project analyse the project proposal, Mid-Point Report Submission where the half-developed porotype and relevant research details were demonstrated to the project manager and assessor. Lastly the final submission where the fully developed system along with the completed document was submitted.

The development of the deep learning based cross platform mobile application that is capable of identifying crop diseases by analysing plant leaves was begun with specified and finalized clear requirements. Having clear requirements effects the success of the project a lot because if the project requirements get changed additional work should be done in order to revise the project deliverables. Capture image of a crop plant leaf, uploading process of a captured plant leaf, develop the deep learning model that recognize plant disease, view predicted results for the user and display images and a short description about the crop disease can be taken as the mainly considered functional requirements for this project and all the functional requirements were satisfied successfully.

Using Convolutional Neural Networks, the developer has implemented the deep learning model that identifies the crop diseases and the developer has performed secondary research observing research papers, e-books, journal articles and etc. in order to finalize the technologies that was used in the project and all the justification were included in the literature review section of the report. The deep learning model has earned a higher accuracy of 93.75% by training the model with 10,000 manually collected diseased plant leaf images. By completing appropriate project milestones and meeting all the functional requirements were caused for the success of this project. Furthermore, the project “Plant Safe- Deep learning based cross platform mobile application that is capable of recognizing crop diseases” is a successful project.

8.2 NEW LEARNING OUTCOMES BY DOING THE PROJECT

Most likely in all the degrees in a Software Engineering degree also final year project plays a key role that dispenses a great occasion to apply the educated and studied technical knowledge and software engineering principles in a real time problem. This project is special because the even though the deep learning portion of system is a major part of the project, it was a brandnew learning area for the developer. In order to come up with a well-suited tech stack the developer had to research on various kinds of similar systems and the technologies that they have been using in those systems as well. Not only about the technologies that the system has been developed by completing the project in a given timeline improved the time management skills of the developer as well.

The system mainly consists with machine learning algorithms and image processing techniques. In order to use above mentioned technologies, the developer had to self-learn, which is an excellent teaching to become a great software engineer. The system is developed using most modern technologies the current world and by researching about the various kinds of alternative technologies that can be used in the project, the developer has broadened the knowledge on a lot of technologies such as backend frameworks, cross-platform mobile application development technologies, various kinds of traditional machine learning algorithms and etc.

The final year project is not only about the knowledge that the developer has gained, by fulfilling the given milestones and completing previously planned tasks helped to improve the time management skills and the punctuality of the developer. Having meetings with the supervisor and demonstrating prototype systems in weekly meetings and demo sessions really improved the communication and presentation skills of the developer. Project planning, task management skills, self-learning skills were improved by researching about the new technologies and algorithms and delivering final outcome of the project in a given time frame. Considering all above-mentioned facts, it is clear that by completing the final year project the developer has improved and broaden the knowledge on various kinds of technologies and key soft skills that are really important in software engineering.

8.3 DIFFERENT STRATEGIES DIFFERENTLY IF THE PROJECT WERE TO BE REPEATED

In order to finalize the best suited technologies for the project the developer himself done secondary research about machine learning algorithms and image processing techniques by observing research reports, journal articles, e-books and analysing existing similar systems. If the project would be to repeated the developer will be able to develop the existing system using a different machine learning algorithm without spending more time on researching about the machine learning and image processing techniques again.

When developing the system there were plenty of alternative traditional machine learning algorithms that could be used such as Random Forest, SVM classifier, k-NN and Naïve Bays which weren't selected because of various reasons and the justification is included in the literature review area. The developer can find solutions for the given reasons and try reimplementing the system using an above-mentioned machine learning algorithm. Not only the algorithm selection but also developer can make the changes on project scope by getting more advises and feedbacks from the project supervisor to complete this final year project successfully.

8.4 RECOMMEND EXTRA FEATURES IF THE PROJECT COULD BE EXTENDED

Acquiring data and images from real time agricultural environments can be tough and time consuming. In this project it has used 10,000 different diseased plant leaves according to selected 10 different crop diseases such as cinnamon gall forming mites, manioc mosaic virus disease, coffee powdery mildew and etc. The developer has selected most commonly spreading and localized plant diseases in order to narrow down the number of classifying classes for the application due to the limited time frame because of the hardware limitations and data acquiring problems.

If the project could be extended the developer recommend to increase the number of the crop diseases that the system supports from 10 to 100 as the second phase of the system. Moreover, the developer has an intention to classify the supported crop diseases with a higher accuracy

by designing the system to use a deeper level CNN architecture in order to increase the performance and the efficiency of the system.

CHAPTER 9: FUTURE SCOPE

- Plant Disease Diagnosis and Treatment using AI algorithms can prove to be a major breakthrough as far as the modern technology is concerned. With advances in Artificial Intelligence especially Deep Learning, the disease in the plant could be detected more efficiently and accurately without experts' advice.
- By removing expert's intervention, diseases in plants could be detected more quickly as the user just has to click an image of the diseased plant and the system will detect the disease and show the cure.
- Furthermore, complete human intervention can be avoided by deploying AI robots in the fields which would scan the fields for suspected disease and spray the correct medicine to cure the disease.
- Also, it can further be extended by installing cameras in the fields which would take images of the plant leaves regularly and check for their healthiness and if any disease is found they could sound an alarm so that the farmer could come and check what the alarm was for.

CHAPTER 10: REFERENCES

[1] Stephane Georges Mallat, Understanding Deep Convolutional Networks, Philosophical Transaction of The Royal Society: A mathematical, physical and Engineering Science 374(2065), 2016.

[2] Sharada Prasanna Mohanty, David Hughes, and Marcel Salath, Using Deep Learning for Image-Based Plant Disease Detection, Front. Plant Sci. 7:1419. doi: 10.3389/fpls.2016.01419, 2016.

[3] Amanda Ramcharan, Kelsee Baranowski, Peter McCloskey, Babuali Ahmed, James Legg, and David Hughes, Deep learning for Image-Based Cassava Disease Detection, Front. Plant Sci. 8:1852 doi: 10.3389/fpls.2017.01852, 2017.

[4] Philipp Lottes, Jens Behley, Nived Chebrolu, Andres Milioto, Cyrill Stachniss, Joint Stem Detection and Crop-Weed Classi_cation for Plant-specific Treatment in Precision Farming, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018.

[5] wikipedia.org, 'Citrus Canker', 2005. [Online]. Available: https://en.wikipedia.org/wiki/Citrus_canker. [Accessed: 18 – Apr – 2019].

[6] A.K. Das, National Research Centre for Citrus, J. Appl. Hort.,5(1):52-60, January-June, 2003

[7] wikipedia.org, 'Citrus Greening Disease', 2005. [Online]. Available: https://en.wikipedia.org/wiki/Citrus_greening_disease. [Accessed: 19 – Apr – 2019].

[8] wikipedia.org, 'Papilio demoleus', 2005. [Online]. Available: https://en.wikipedia.org/wiki/Papilio_demoleus. [Accessed: 20 – Apr – 2019].

[9] wikipedia.org, 'Phyllocnistis citrella', 2009. [Online]. Available: https://en.wikipedia.org/wiki/Phyllocnistis_citrella. [Accessed: 21 – Apr – 2019].

[10] wikipedia.org, 'Gummosis', 2005. [Online]. Available: https://en.wikipedia.org/wiki/Gummosis. [Accessed: 21 – Apr – 2019].

[11] britannica.com, 'Citrus', 1998. [Online]. Available: https://www.britan [12] nica.com/plant/Citrus. [Accessed: 10 – Jan – 2019].

AMITY UNIVERSITY

UTTAR PRADESH

MINOR PROJECT

on

PLANT AND FRUIT DIAGNOSIS AND TREATMENT THROUGH DEEP LEARNING

Submitted to

Amity University Uttar Pradesh

in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

In

Computer Science & Technology By

SACHIN KUMAR(A2305219503)

MAYANK KHANDELWAL(A2305219504)

TUSHAR GOYAL(A2305219540)

ISHU KHANDELWAL (A2305219549)

Under the guidance of **Dr.**

Ram Paul

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY AMITY UNIVERSITY UTTAR PRADESH

DECLARATION

We **Ishu Khandelwal, Mayank Khandelwal, Tushar Goyal** and **Sachin Kumar** student of B. Tech.(CSE) hereby declare that the "**PLANT AND FRUIT DIAGNOSIS AND TREATMENT THROUGH DEEP LEARNING**" which is submitted by us to the Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in CSE has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition.

Tushar Goyal

(A2305219540)

Sachin Kumar

(A2305219503)

Ishu Khandelwal

(A2305219549)

Mayank Khandelwal

(A2305219504)

CERTIFICATE

On the basis of the declaration submitted by Ishu Khandelwal, Tushar Goyal, Sachin Kumar, and, Mayank Khandelwal, student of B.Tech. CSE, I hereby certify that the minor project titled " **Plant and fruit diagnosis and treatment through deep learning**" which is submitted to the Department of Computer Science and Engineering, Amity School of Engineering and Technology, Amity University Uttar Pradesh, Noida, in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in CSE is an original contribution with existing knowledge and faithful record of work carried out by her under my guidance and

supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Date : 30.11.2022

Dr. Ram Paul Assistant Professor –II
Department of Computer Science and Engineering
Amity School of Engineering and Technology Amity University Uttar Pradesh, Noida

CONTENTS

CHAPTER 1: ABSTRACT

It is critical to have control over plant disease since it influences the overall quality and number of species of the plants, plus the nation's infrastructure. To avoid revenue damage and the endangerment of particular species, automated detection and sorting of leaf illness is critical. In past, numerous machine learning (ML) models have been suggested to observe and treat plant disease; nevertheless, they are not accessible because of the difficulty of procuring advanced equipment, the restricted scalability of models, and the complexity and inefficiencies of their application. Local expertise and previous experiences have historically been used to diagnose plant pathogens. A plant's health may be determined by a qualified specialist. If an unhealthy plant is discovered, signs appear on its leaves and fruits. Diagnosis of plant disease is hard because of the fact that leaves have distinct symptoms that need to be examined. Even

experienced plant pathologists and agronomists have trouble differentiating among various illnesses because of the quantity of adult plants, their extensive prior Phyto static problems, and their inherent ambiguity. This research paper will undergo ML/ deep learning in the field of plant health analysis.

CHAPTER 2: INTRODUCTION

To meet the difficulties confronting farmers, agrochemical firms provide a wide variety of products including those for increased yield, insect resistance, toughness, quality of water, and many other concerns. Marketers should use actual-world circumstances to calculate the effectiveness of their goods, in place of using controlled experiments alone. In single-crop farms, different fields are used and a particular procedure implemented in each. Hybrid seeds are planted in one location while a second location is fertilized, and the process is repeated for as many plots as are necessary. By observing the physical condition of crop in the plot in which the therapy was given, we are able to get a measure of the relative effectiveness of each therapy.

A widely accepted measure of plant growth is the Leaf Area Index (LAI), which has been inured to learn the effects of vegetation (grown as well as wild), climate, and surroundings in hundreds

of different research papers. Fertiliser and irrigation efficiencies are critical when it comes to most agronomical and horticultural research. LAI is an essential indication of how much rainfall and sunlight is intercepted; how much energy is converted; how much water is retained; and finally, how much vegetation is formed. Crop making varies depending on the kind of crop, as well as on the phase of the plant's life.

The difficult and time-taking procedure of gathering leaves using traditional LAI measurement takes days or weeks to complete. Each leaf must be carefully measured to ensure that the length is precise. Because the procedure is laborious and costly, measurements are rare and must be performed on limited controlled regions. Furthermore, in small numbers, the data are inadequate to construct models or train model.

Sophisticated data science methods like drones, computer vision, and sophisticated algorithms are already being used to solve the difficulties facing agrochemical businesses. But nevertheless, such difficulties remain. In crops that develop tall and thin, such as maize, features such as leaf direction, alignment, length, form, and twisting are hard to observe. Additional complications are brought on by non-constants including weather, topography, cloud refraction, and occlusion. As already mentioned, variables that influence plant health and treatment outcomes change over time, and frequent monitoring is thus needed.

Computer vision and deep learning techniques are maturing, and this is helping scientists in their assessment of LAI. With major agrochemical businesses, Tiger Analytics has created these services. We describe the many methods and difficulties in this post.

In the case of DL, the major difficulties in solving the problem is the scarcity of training data. It would take many months of human leaf measuring and a massive resource investment to train the network adequately. It must be produced (the training data, together with the model source) and the ‘real' data (which is rarely acquired and used) is kept for verification.

It is vital to have strong collaboration with botanists and agricultural experts in order to analyse the drone pictures and create leaf profiles. The breadth, twist, and color saturation of each leaf profiles are reproduced for each cross-section. Various random alternatives are implemented to leaf widths, distribution, the leaf area on plants, and plant age, as well as the orientation of development.

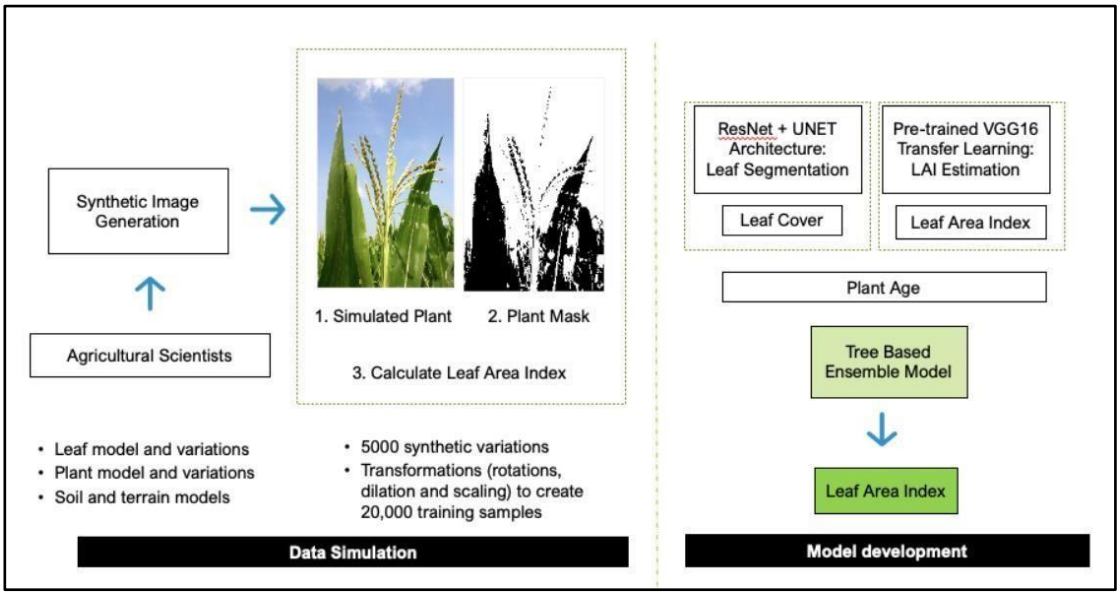


Figure 2.1

It is possible to benefit product quality and avoid economic losses by focusing on plant health evaluation and disease detection. Early diagnosis and species-specific disease categorization are critical to ensure crop production integrity. There have been many observations that have displayed the importance of early identification of plant illnesses since, over time, the species is affected by disease, and its signs emerge on the leaves. If a crop has an illness, and visible symptoms appear on the leaves, they provide a way to recognize and determine the illness. In order to contain the spread of illness, it is difficult to regulate and survey the disease's development. Species of particular fungi or bacteria are often present in spots and blight (lesions). Fungi provide “signs” of illness, such as molds forming and fruiting bodies appearing in the region of dead plant tissue as black spots. Haphazard dark-coloured and water-soaked patches with a distinct border that may have a light-coloured ring are occasionally seen when pathogenic bacteria occur during warm conditions on leaves or fruits. The decaying region appears identical to normal tissue, so it is difficult to identify illnesses at first.

An only main task of technology in agriculture is to assist in the assessment of the area. In an attempt to inspect the plant illness recognition and classification using machine learning and image listing methods, researchers are trying out different machine learning and image processing approaches. Detecting plant diseases directly is complicated, time-consuming, and inaccurate. As there is a significant amount of time and effort required to conduct a comprehensive health assessment on a single plant in a large-scale planting, we conduct such evaluations on an annual basis.

Plants may have more than one illness that have the similar outline of characteristics, making it hard to recognize the particular aliment. The main symptoms of the color change, bunch hardening, and irregular wood ripening that occurs with Grapevine Yellow (GY) frequently appear around the start of the summer months, such as leaf discoloration, bunch hardening, and unusual wood fruiting, enabling GY to be simply discriminated from other grapevine disorders that may share common modifications (e.g., leafroll or direct damage due to feeding of leafhopper). While there is not always a uniform symptomatology across GYs, the manifestation of symptoms among various GYs is very standardized, which means symptomatology is of little use in differentiating one GY from another. GY agent-induced symptoms and indexes on the hybrid Baco 22A, employed in the past, could not upgrade the indexing method for phytoplasmas, which are weakly transferred by grafting on woody plants.

The contrast among a deep learning method and a computer vision approach is that deep learning techniques have a lower mean absolute error when comparing to computer vision methods. With architecture, training data, processes, and forms, models vary often, but because there is little ground truth data, this makes deep learning initiatives unique. The most effective way to surmount this obstacle is for plant scientists and data scientists to join forces to build a simulated information set that has sufficient granularities to closely resemble the actual field environment but which includes generalization and randomness in order to prevent over fitting.

By establishing of a set of connections based on synthesized data and agreeing on reliable performance in the real world, it is essential to critically examine the weights in the neural network. To get almost right results, an alternative procedure of constant comparison of each intermediate result to that of plant scientists is used, after which the learning's are included into future training synthesis.

An assortment of various farm locations, crop mixtures, species particularities, environmental factors, and surveillance frequency will all have to be taken into consideration by agrochemical firms when investigating the effectiveness of their goods. Plant beds will not only include instruments to monitor factors such as humidity, dirt, and sunlight, but also temperatures and other factors.

Using artificial data, the artificial plant solution shows that working deep learning models with botanists can be done. Generalisability of synthesized data sets is assured by domain knowledge, context, and contextual understanding.

PICTURE RECOGNITION TECHNOLOGY ON DL

This is considered the best picture recognition technique, but it doesn't need extracting particular characteristics, since only via experimentation can it discover suitable features.

CHAPTER 3: DEEP LEARNING

Hinton et al. first proposed the idea of Deep Learning (DL) in their article in Science in 2006. Deep learning is the process of utilizing neural networks for data survey and feature learning, where low-level characteristics are retrieved by many hidden units, and then integrate those values to produce abstract high-level features. Existing methods, relying on artificiallydesigned characteristics, have had their own disadvantages, and deep learning solves them by attracting increasing interest from academics. Computer vision, pattern classification, voice recognition, natural language, and referring systems are all areas where the algorithm has recently been effectively used.

A manual feature extraction technique is only capable of extracting the feature data, and thus confines the potential of finding deeper and more complicated picture features. Additionally, DL may be used to address this issue. It is able to automatically obtain multi-level image feature material, such as limited features, intermediary characteristics, and high-level semantic, from

the real image without supervision. Most conventional plant disease and pest detection algorithms use manual-designed characteristics, which is time-consuming and subject to error, and thus can't be applied mechanically. This statement means that deep learning is not only capable of learning features from huge data sets, but it is able to do it without human modification. It is comprised of many layers, and this provides it with both a high level of learning strength to flexibly represent various characteristics. As a result, it can automatically identify picture content. Due to this, deep learning has the potential to make significant contributions to plant pest and diseases identification picture recognition. To this day, deep learning models such as deep belief network (DBN), deep Boltzmann machine (DBM), stack de-noising auto encoder (SDAE), and deep convolution neural network (DCNN) have produced several well-known models (CNN). By using deep neural network models to provide automated face detection in high-dimensional feature vector, these techniques are more suitable for the problem of picture identification than conventional human design methodologies. The accuracy of deep neural networks is increasing jointly with the quantity of training samples and the computing capacity. This surge in popularity for deep learning is being felt both within the industry and academia, with deep neural network models outperforming conventional models by a considerable margin. A (DCNN) is most often used in recent years in this field.

NEURAL NETWORK

Artificial neural networks (ANNs) are a computational model used in machine learning, computer science and other research disciplines, which is based on a large collection of connected simple units called artificial neurons, loosely analogous to axons in a biological brain. Connections between neurons carry an activation signal of varying strength. If the combined incoming signals are strong enough, the neuron becomes activated and the signal travels to other neurons connected to it. Such systems can be trained from examples, rather than explicitly programmed, and excel in areas where the solution or feature detection is difficult to express in a traditional computer program. Like other machine learning methods, neural networks have been used to solve a wide variety of tasks, like computer vision and speech recognition, that are difficult to solve using ordinary rule-based programming.

Typically, neurons are connected in layers, and signals travel from the first (input), to the last (output) layer. Modern neural network projects typically have a few thousand to a few million neural units and millions of connections; their computing power is similar to a worm brain, several orders of magnitude simpler than a human brain. The signals and state of artificial

neurons are real numbers, typically between 0 and 1. There may be a threshold function or limiting function on each connection and on the unit itself, such that the signal must surpass the limit before propagating. Back propagation is the use of forward stimulation to modify connection weights, and is sometimes done to train the network using known correct outputs

Neural Networks are composed of multiple nodes, which imitate biological neurons of human brain. The neurons are connected by links and they interact with each other. The nodes can take input data and perform simple operations on the data. The result of these operations is passed to other neurons. The output at each node is called its activation or node value.

Each link is associated with weight. ANNs are capable of learning, which takes place by altering weight values. The following illustration shows a simple Neural Network:

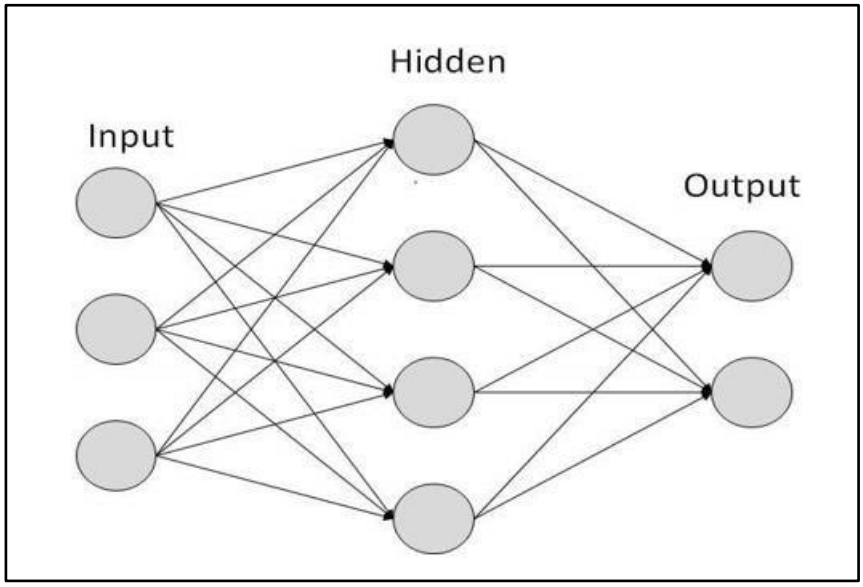


Figure 3.1: An example of an Artificial Neural Network (ANN)

CHAPTER 4: IMAGE RECOGNITION

Convolutional Neural Networks (**CNNs**) are a category of Neural Networks that have proven very effective in areas such as image recognition and classification. CNN's have been successful in identifying faces, objects and traffic signs apart from powering vision in robots and self-driving cars.

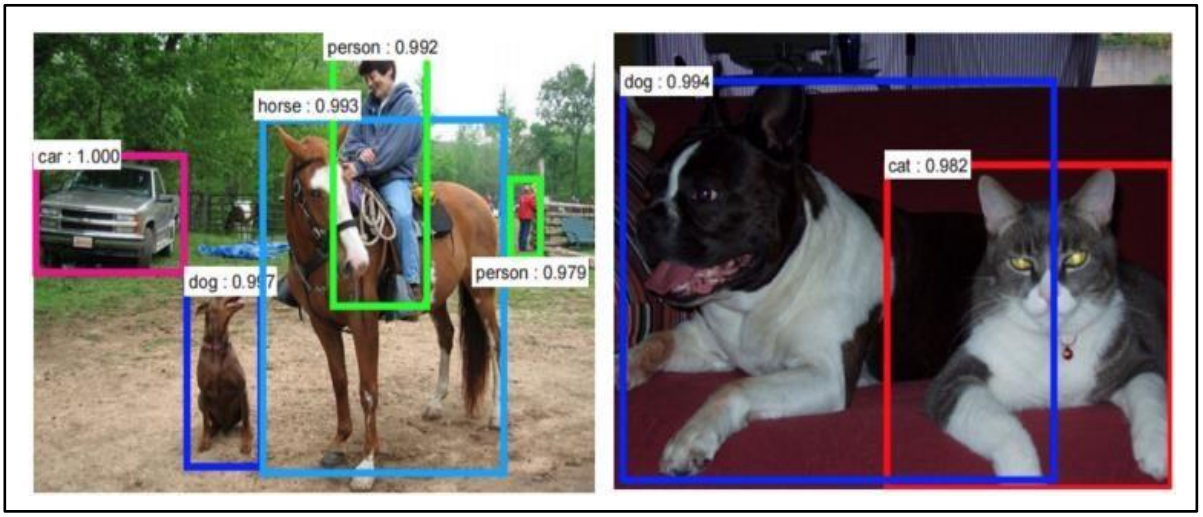


Figure 4.1: An example of ConvNets being used for recognizing everyday objects, humans and animals.

THE LENET ARCHITECTURE (1990S)

LeNet was one of the very first convolutional neural networks which helped propel the field of Deep Learning. LeNet architecture was used mainly for character recognition tasks such as reading zip codes, digits, etc.

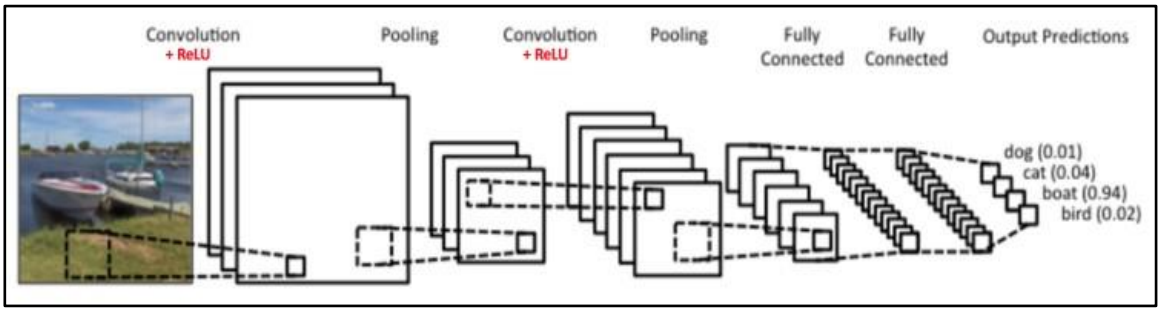


Figure 4.2: Basic setup of a Convolutional Neural Network

The Convolutional Neural Network in Figure 5 above is similar in architecture to the original LeNet and classifies an input image into four categories: dog, cat, boat or bird (the original LeNet was used mainly for character recognition tasks). As evident from the figure above, on receiving a boat image as input, the network correctly assigns the highest probability for boat (0.94) among all four categories. The sum of all probabilities in the output layer should be one There are four main operations in the ConvNet shown in figure above:

5. Convolution
6. Non-Linearity (ReLU)
7. Pooling or Sub Sampling
8. Classification (Fully Connected Layer)

RELU

ReLU stands for Rectified Linear Unit and is a non-linear operation. Its output is given by:

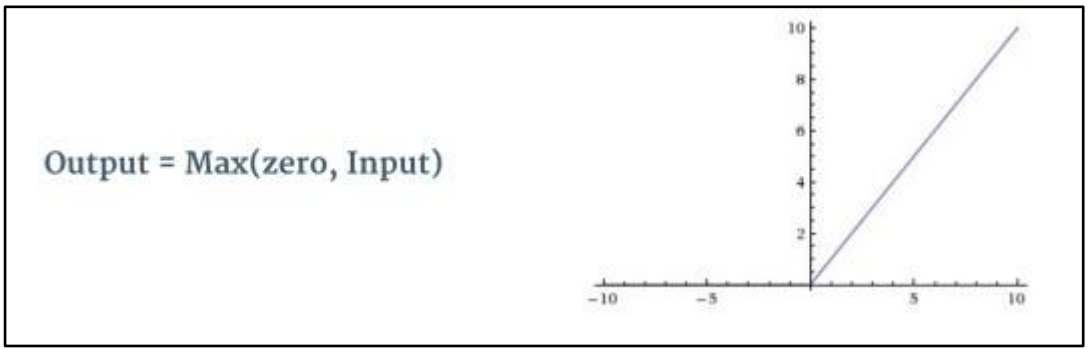


Figure 4.3: RELU Function

ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our ConvNet, since most of the real-world data we would want our ConvNet to learn would be non-linear

(Convolution is a linear operation – element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

TENSORFLOW

TensorFlow is an open-source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API. TensorFlow was originally developed by researchers and engineers working on the Google Brain Team within Google's Machine Intelligence research organization for the purposes of conducting machine learning and deep neural networks research, but the system is general enough to be applicable in a wide variety of other domains as well.

TRANSFER LEARNING

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task.

It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision and natural language processing tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems.

Transfer learning is related to problems such as multi-task learning and concept drift and is not exclusively an area of study for deep learning.

Nevertheless, transfer learning is popular in deep learning given the enormous resources required to train deep learning models or the large and challenging datasets on which deep learning models are trained.

Transfer learning only works in deep learning if the model features learned from the first task are general.

Two common approaches to use transfer learning are as follows:

1.) Develop Model Approach

2.) Pre-trained Model Approach

SIGMOID FUNCTION

A sigmoid function is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve. Often, sigmoid function refers to the special case of the logistic function.

Special cases of the sigmoid function include the Gompertz curve (used in modeling systems that saturate at large values of x) and the ogee curve (used in the spillway of some dams).

Sigmoid functions have domain of all real numbers, with return value monotonically increasing most often from 0 to 1 or alternatively from −1 to 1, depending on convention.

A wide variety of sigmoid functions including the logistic and hyperbolic tangent functions have been used as the activation function of artificial neurons. Sigmoid curves are also common in statistics as cumulative distribution functions (which go from 0 to 1), such as the integrals of the logistic distribution, the normal distribution, and Student's t probability density functions.

ADAM OPTIMIZER

The Adam optimization algorithm is an extension to stochastic gradient descent that has recently seen broader adoption for deep learning applications in computer vision and natural language processing.

Adam was presented by Diederik Kingma from Open AI and Jimmy Ba from the University of Toronto in their 2015 ICLR paper (poster) titled “Adam: A Method for Stochastic Optimization “

Adam is different to classical stochastic gradient descent. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training. A learning rate is maintained for each network weight (parameter) and separately adapted as learning unfolds.

Adam as combining the advantages of two other extensions of stochastic gradient descent. Specifically:

Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients (e.g., natural language and computer vision problems).

Root Mean Square Propagation (RMSProp) that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight (e.g., how quickly it is changing). This means the algorithm does well on online and non-stationary problems (e.g., noisy).

Adam realizes the benefits of both AdaGrad and RMSProp. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance).

Specifically, the algorithm calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages.

The initial value of the moving averages and beta1 and beta2 values close to 1.0 (recommended) result in a bias of moment estimates towards zero. This bias is overcome by first calculating the biased estimates before then calculating bias-corrected estimates.

CHAPTER 5: CONVOLUTION NEUTRAL NETWORK (CNN)

A complicated network configuration and the capability to execute convolution operations are ordinary to convolutional neural networks, which are also recognized as CNNs. In Figure 2, the image prediction network model is built from input, convolution, pooling, full link, and output layers. This technique is known as alternating convolution and pooling, and it follows the traditional configuration of a convolution followed by a pooling. In this replica, the convolution and pooling layers exchange a few magnitudes, and then when the neurons of the convolution layer are linked to the neurons of the pooling layer, no full network is established. CNN is one of the most common deep learning models used in industry. CNN's unique structural features allow it to have an edge in picture identification. Meanwhile, CNN's performance in computer vision tasks has served to enhance the rapidly increasing interest in deep learning.

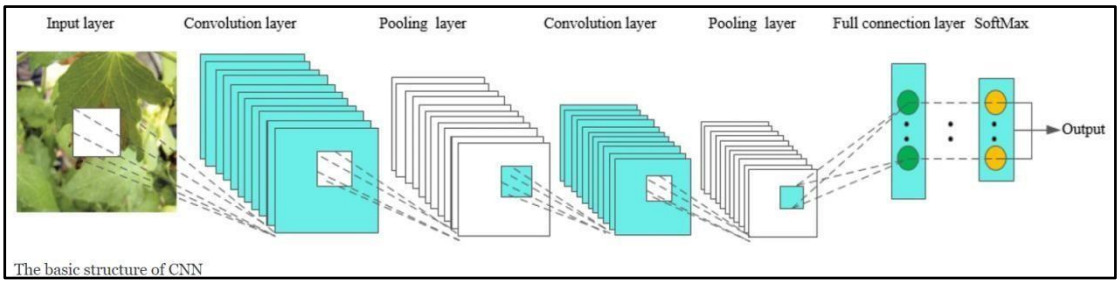


Figure 5.1

Convolution core is defined initially in the convolution layer. When used on a local receptive field, the convolution core may be regarded a local receptive field, and the biggest benefit of the convolution neural network is that it works on local receptive fields. Convolution processing occurs on the feature map, which is brought in contact with the convolution core. Extraction of features and pooling happen at the pooling layer after feature extraction is done on the

convolution layer. However, there are now three primary ways of pooling, with the most commonly worn one being to utilize the mean, maximum, and random values from the local receptive field. Data enters several convolution and pooling layers, after which they go via a full-connection layer before making it to the top layer of neurons. The SoftMax technique may be use to recognize values in the full-connection layer, and then those principles are sent to the output layer where they are used to provide results.

TECHNIQUES FOR FINDING PLANT AND FRUITS HEALTH :

This segment provides a high-level impression of numerous techniques for finding plant disease and pest outbreaks based on deep learning. Even if plant diseases and pests' detection techniques based on deep learning align with the computer vision problem, we may still consider them to be a deployment of significant classical networks in the area of agriculture. Figure 3 demonstrates the many network structure possibilities, and how they may be further split based on those structural options. According to the processing features of each kind of techniques, this article is split into many distinct sub-methods, as shown in Fig. 3.

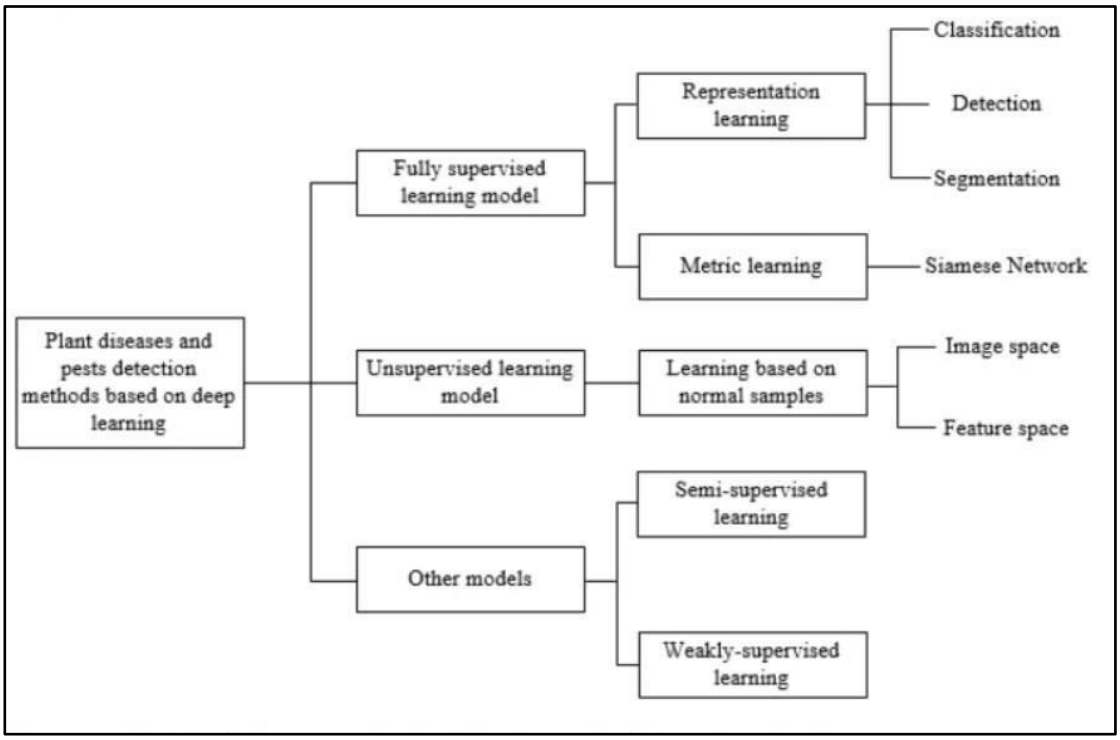


Figure 5.2

CHAPTER 6: CHALLENGES IN DEEP LEARNING

Even while embracing latest techniques and technologies such as computerized phenotyping comes with its personal difficulty and hurdles to implementation, such as obstacles to entry and user reluctance, it is still critical to go on with the effort. To help with data gathering, we now face three major types of challenges: (1) collecting training data, (2) training the model, and (3) ensuring model transferability.

When HTP platforms, such as UAS, field robots, or tractor-mounted equipment, acquire highthroughput phenotypic data, the results show greater levels of opacity and ambient noise. To complicate things further, a method of field phenotyping such as automated phenotyping introduces an even greater range of complicated data variability, such as cloud cover, light intensity, and wind speed, which interfere with algorithm operation. LIDAR (light detection and ranging) is a technology that use a pulse light source and a CCD or a CMOS array to determine distance and other characteristics. LIDAR systems typically use pulsed light sources and CCD or CMOS arrays to get information.

It is critical and difficult to keep algorithm performance optimized when using training data. Research requirements, including the need for a publicly accessible, open-source, shared database on plant stressors at leaf scale, were met by the establishment of the PlantVillage database, which included more than 54 306 pictures of 14 crop species and 26 illnesses. There have previously been numerous studies done on Plant Village data, and more sources of data will be required to build effective models. In addition to the open-source database mentioned above, other open-source databases include a maize NB-lèbre (a common disease) database including 18,222 digital pictures of maize leaves from the field, whether manually taken, placed on a boom, or by unmanned aerial vehicles (UAVs). It's the biggest publicly accessible picture collection annotated for a plant disease. There were about 105 705 NLB lesions annotated by human specialists, making this a huge image set. Over 90% accuracy using 18,000 pictures of agricultural areas in Africa, Latin America, and India, Selvaraj et al. trained a DCNN model for

banana sickness and pest classification. By release all of the pictures from their study, the author have permitted the scientific community to contact them. This collection is called the plant disease database (PDDB), and it contains over 2,300 pictures of over 170 plant diseases and other illnesses. Subdividing the pictures to amplify the number of images to 46 513 increased the database size.

To satisfy the need for big datasets for applying DL and transfer learning in illness categorization, image annotation databases comprising pictures collected from cooperating universities help meet the requirement. DL implementation is aided by hyperspectral sensors, which create enormous datasets and huge quantities of information from just a little figure of sample. Also, the FAIR principles of openness and reusability are supported because of the creation of a single database for annotated plant stress pictures. Treat training data as complicated and varied as the algorithm may face in reality; good performance accuracy is important. When using data from one year, at one location, or for one plant variety or crop stage, an algorithm cannot be used to real-life situations in various contexts since the diversity of trait expression among these factors does not span. Thus, efficient and early experiment planning, sensors, picture resolution, and end-use will need dedicated preparation.

Lastly, AL and transfer learning may aid ML models for fresh challenge and stressors by broadening ML's applicability (disease, drought, flooding, salinity, temperature, nutrient, pest, weed stresses). Image Net which is also called as a large-scale annotated plant stress collection, which has over 14 million pictures. At the leaf, canopy, and plot levels, such datasets enhance the potential of getting state-of-the-art machine learning models for use in farming fields or research settings or in plant breeding initiatives. In spite of Image Net, AlexNet, and ResNet having previously been used to transfer learning’s for plant image-based models, these networks still offer difficulties since they are non-plant image-based pertained DL models. on the other hand, training a model to detect, categorize, or compute pressure at plot/field scale does not apply to the leaf range, and inversely, a model taught to evaluate the stresses of the whole plot/field does not straightforwardly be relevant to the individual leaf scale. stress symptoms are significantly different in pictures obtained in natural settings than in conditions that are designed to mimic disease or pest pressure (i.e., greenhouses). This means that in order for an algorithm to be implemented effectively, a strategy for training data gathering must be adopted. By integrating previous big crowd-sourced datasets, the creation of extremely large and varied plant stress datasets for DL model training is possible. With these resources, multispectral/hyperspectral sensor data, with combined AL methods and/or transfer learning (in

combination with plant stress pictures), may now be used to identify, classify, and quantify plant stress at the field scale. Spectral image super-resolution, particularly dealing with spectral pictures at various scales of spatial resolution, as well as connecting spectral and trichromatic (RGB) images are coming fast with new statistical software that use ML methods. This will allow disease phenotype testing to be more cost-effective.

CHAPTER 7: CONCLUSION

Plant stress severity phenotyping is a key metric in gauging the risks of crop losses owing to diverse biotic and abiotic stressors. In terms of genetics, better disease resistant and stress-tolerant genotypes may be identified, and choices about illness management can be assessed. Present methods for measuring stress intensity use the following methods at different scales: Exact count of lesion counts, estimations of the intensity or surface area impacted by a specific stress at covering and field levels, or estimations of the group of species affected by a stress. Image-based disease phenotyping has several benefits, but increasing the speed, accuracy, reliability, and scalability is only possible if ML is used in this process. These concerns include less human error, as well as variability across raters. The purpose is to integrate ML and DL into HTP of plant characteristics in the field for real-time application. Further work is necessary in order to develop and use ML techniques to supplement sensor and phenotype platforms when it comes to quantifying and predicting disease severity. These attempts will work in tandem with recent developments in imaging technology for collecting higher-resolution and higherdimensional data.

CHAPTER 8: REFERENCES

1. Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-up robust features (surf). Comput. Vis. Image Underst. 110, 346–359. doi: 10.1016/j.cviu.2007.09.014
2. Chéné, Y., Rousseau, D., Lucidarme, P., Bertheloot, J., Caffier, V., Morel, P., et al. (2012). On the use of depth camera for 3d phenotyping of entire plants. Comput. Electron. Agric. 82, 122–127. doi: 10.1016/j.compag.2011.12.007
3. Dalal, N., and Triggs, B. (2005). "Histograms of oriented gradients for human detection," in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. (IEEE) (Washington, DC).
4. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei L. (2009). "Imagenet: A large-scale hierarchical image database," in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on. (IEEE).
5. Ehler, L. E. (2006). Integrated pest management (ipm): definition, historical development and implementation, and the other ipm. Pest Manag. Sci. 62, 787–789.
6. doi: 10.1002/ps.1247
7. Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. Int. J. Comput. Vis. 88, 303–338.
8. Garcia-Ruiz, F., Sankaran, S., Maja, J. M., Lee, W. S., Rasmussen, J., and Ehsani R. (2013). Comparison of two aerial imaging platforms for identification of huanglongbing-infected citrus trees. Comput. Electron. Agric. 91, 106–115. doi:
9. GSMA Intelligence (2016). The Mobile Economy- Africa 2016. London: GSMA.
10. Harvey, C. A., Rakotobe, Z. L., Rao, N. S., Dave, R., Razafimahatratra, H., Rabarijohn, R. H., et al. (2014). Extreme vulnerability of smallholder farmers to

agricultural risks and climate change in madagascar. Philos. Trans. R. Soc. Lond. B Biol. Sci. 369:20130089. doi: 10.1098/rstb.2013.008

11. He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. arXiv:1512.03385.

CHAPTER 12: PLAG REPORT

turnitin

Similarity Report ID: oid:16158:35302329

PAPER NAME	AUTHOR
ABSTRACT 12.pdf	SCH K

WORD COUNT	CHARACTER COUNT
3671 Words	18396 Characters
PAGE COUNT	**FILE SIZE**
80 Pages	6.0MB
SUBMISSION DATE	**REPORT DATE**
May 12, 2023 3:50 PM GMT+5:30	May 12, 2023 3:51 PM GMT+5:30

5% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 1% Internet database
- 0% Publications database
- Crossref database
- Crossref Posted Content database
- 4% Submitted Works database

Excluded from Similarity Report

- Bibliographic material
- Cited material
- Small Matches (Less then 10 words)

CHAPTER 13: RESEARCH PAPER

© 20XX IJNRD | Volume X, Issue X Month 20XX | ISSN: 2456-4184 | IJNRD.ORG

PLANT AND FRUIT DISEASE DIAGNOSIS AND TREATMENT THROUGH DEEP LEARNING

[1]Dr Ram Paul, [2]Sachin Kumar, [3]Mayank Khandelwal, [4]Ishu Khandelwal, [5]Tushar Goyal
[1]Assistant Professor-II, [2]Students,
[1]Amity School of Engineering and Technology, Noida, India

ABSTRACT

It is critical to have control over plant disease since it influences the overall quality and number of species of the plants, plus the nation's infrastructure. To avoid revenue damage and the endangerment of particular species, automated detection and sorting of leaf illness is critical. In past, numerous machine learning (ML) models have been suggested to observe and treat plant disease; nevertheless, they are not accessible because of the difficulty of procuring advanced equipment, the restricted scalability of models, and the complexity and inefficiencies of their application. Local expertise and previous experiences have historically been used to diagnose plant pathogens. A plant's health may be determined by a qualified specialist. If an unhealthy plant is discovered, signs appear on its leaves and fruits. Diagnosis of plant disease is hard because of the fact that leaves have distinct symptoms that need to be examined. Even experienced plant pathologists and agronomists have trouble differentiating among various illnesses because of the quantity of adult plants, their extensive prior phytostatic problems, and their inherent ambiguity. This research paper will undergo ML/ deep learning in the field of plant health analysis.

INTRODUCTION

To meet the difficulties confronting farmers, agrochemical firms provide a wide variety of products including those for increased yield, insect resistance, toughness, quality of water, and many other concerns. Marketers should use actual-world circumstances to calculate the effectiveness of their goods, in place of using controlled experiments alone. In single-crop farms, different fields are used and a particular procedure implemented in each. Hybrid seeds are planted in one location while a second location is fertilized, and the process is repeated for as many plots as are necessary. By observing the physical condition of crop in the plot in which the therapy was given, we are able to get a measure of the relative effectiveness of each therapy.

A widely accepted measure of plant growth is the Leaf Area Index (LAI), which has been inured to learn the effects of vegetation (grown as well as wild), climate, and surroundings in hundreds of different research papers. Fertiliser and irrigation efficiencies are critical when it comes to most agronomical and horticultural research. LAI is an essential indication of how much rainfall and sunlight is intercepted; how much energy is converted; how much water is retained; and finally, how much vegetation is formed. Crop making varies depending on the kind of crop, as well as on the phase of the plant's life.

The difficult and time-taking procedure of gathering leaves using traditional LAI measurement takes days or weeks to complete. Every each leaf must be carefully measured to ensure that the length is precise. Because the procedure is laborious and costly, measurements are rare and must be performed on limited controlled regions. Furthermore, in small numbers, the data are inadequate to construct models or train model.

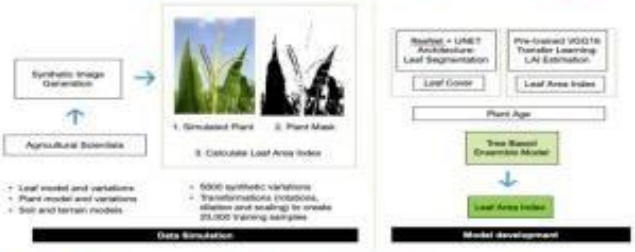
Sophisticated data science methods like drones, computer vision, and sophisticated algorithms are already being used to solve the difficulties facing agrochemical businesses. But nevertheless, such difficulties remain. In crops that develop tall and thin, such as maize, features such as leaf direction, alignment, length, form, and twisting are hard to observe. Additional complications are brought on by non constants including weather, topography, cloud refraction, and occlusion. As already mentioned, variables that

influence plant health and treatment outcomes change over time, and frequent monitoring is thus needed. Computer vision and deep learning techniques are maturing, and this is helping scientists in their assessment of LAI. With major agrochemical businesses, Tiger Analytics has created these services. We describe the many methods and difficulties in this post.

In the case of DL, the major difficulties in solving the problem is the scarcity of training data. It would take many months of human leaf measuring and a massive resource investment to train the network adequately. It must be produced (the training data, together with the model source) and the 'real' data (which is rarely acquired and used) is kept for verification.

It is vital to have strong collaboration with botanists and agricultural experts in order to analyze the drone pictures and create leaf profiles. The breadth, twist, and color saturation of each leaf profiles are reproduced for each cross-section. Various random alternatives are implemented to leaf widths, distribution, the leaf area on plants, and plant age, as well as the orientation of development.

It is possible to benefit product quality and avoid economic losses by focusing on plant health evaluation and disease detection. Early diagnosis and species-specific disease categorization are critical to ensure crop production integrity. There have been many observations that have displayed the importance of early identification of plant illnesses since, over time, the species is affected by disease, and its signs emerge on the leaves. If a crop has an illness, and visible symptoms appear on the leaves, they provide a way to recognize and determine the illness. In order to contain the spread of illness, it is difficult to regulate and survey the disease's development. Species of particular fungi or bacteria are often present in spots and blight (lesions). Fungi provide "signs" of illness, such as molds forming and fruiting bodies appearing in the region of dead plant tissue as black spots. Haphazard dark-



colored and water-soaked patches with a distinct border that may have a light-colored ring are occasionally seen when pathogenic bacteria occur during warm conditions on leaves or fruits. The decaying region appears identical to normal tissue, so it is difficult to identify illnesses at first.

Deep learning theory

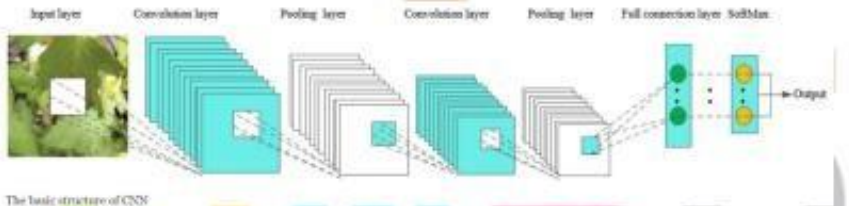
Hinton et al. first proposed the idea of Deep Learning (DL) in their article in Science in 2006. Deep learning is the process of utilizing neural networks for data survey and feature learning, where low-level characteristics are retrieved by many hidden units, and then integrate those values to produce abstract high-level features. Existing methods, relying on artificially-designed characteristics, have had their own disadvantages, and deep learning solves them by attracting increasing interest from academics. Computer vision, pattern classification, voice recognition, natural language, and reffering systems are all areas where the algorithm has recently been effectively used.

A manual feature extraction technique are only capable of extracting the feature data, and thus confines the potential of finding deeper and more complicated picture features. Additionally, DL may be used to address this issue. It is able to automatically obtain multi-level image feature material, such as limited features, intermediary characteristics, and high-level semantic, from the real image without supervision. Most conventional plant disease and pest detection algorithms use manual-designed characteristics, which is time-consuming and subject to error, and thus can't be applied mechanically. This statement means that

deep learning is not only capable of learning features from huge data sets, but it is able to do it without human modification. It is comprised of many layers, and this provides it with both a high level of learning strength to flexibly represent various characteristics. As a result, it can automatically identify picture content. Due to this, deep learning has the potential to make significant contributions to plant pest and diseases identification picture recognition. To this day, deep learning models such as deep belief network (DBN), deep Boltzmann machine (DBM), stack de-noising auto encoder (SDAE), and deep convolution neural network (DCNN) have produced several well-known models (CNN). By using deep neural network models to provide automated face detection in high-dimensional feature vector, these techniques are more suitable for the problem of picture identification than conventional human design methodologies. The accuracy of deep neural networks is increasing jointly with the quantity of training samples and the computing capacity. This surge in popularity for deep learning is being felt both within the industry and academia, with deep neural network models outperforming conventional models by a considerable margin. A (DCNN) is most often used in recent years in this field.

CNN : - (CONVOLUTION NEUTRAL NETWORK)

A complicated network configuration and the capability to execute convolution operations are ordinary to convolutional neural networks, which are also recognized as CNNs. In Figure 2, the image prediction network model is built from input, convolution, pooling, full link, and output layers. This technique is known as alternating convolution and pooling, and it follows the traditional configuration of a convolution followed by a pooling. In this replica, the convolution and pooling layers exchange a few magnitudes, and then when the neurons of the convolution layer are linked to the neurons of the pooling layer, no full network is established. CNN is one of the most common deep learning models used in industry. CNN's unique structural features allow it to have an edge in picture identification. Meanwhile, CNN's performance in computer vision tasks has served to enhance the rapidly increasing interest in deep learning.



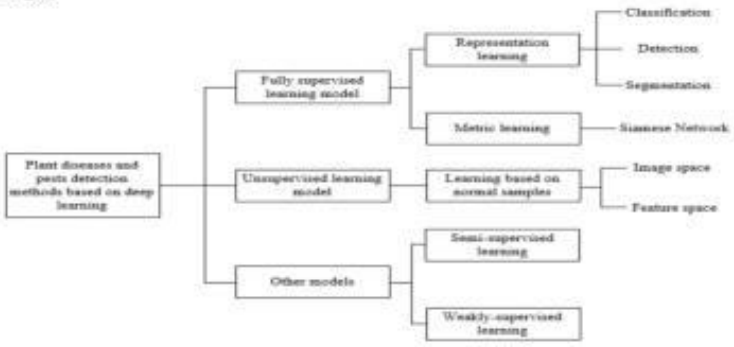
The basic structure of CNN

Convolution core is defined initially in the convolution layer. When used on a local receptive field, the convolution core may be regarded a local receptive field, and the biggest benefit of the convolution neural network is that it works on local receptive fields. Convolution processing occurs on the feature map, which is brought in contact with the convolution core. Extraction of features and pooling happen at the pooling layer after feature extraction is done on the convolution layer. However, there are now three primary ways of pooling, with the most commonly worn one being to utilize the mean, maximum, and random values from the local receptive field. Data enters several convolution and pooling layers, after which they go via a full-connection layer before making it to the top layer of neurons. The softmax technique may be use to recognize values in the full-connection layer, and then those principles are sent to the output layer where they are used to provide results.

TECHNIQUES FOR FINDING PLANT SICKNESS AND PEST OUTBREAKS:

This segment provide a high-level impression of numerous techniques for finding plant disease and pest outbreaks based on deep learning. Even if plant diseases and pests detection techniques based on deep learning align with the computer vision problem, we may still consider them to be a deployment of significant classical networks in the area of agriculture. Figure 3 demonstrates the many network structure possibilities, and how they may be further split based on those structural options. According to the processing features of each kind of techniques, this article is split into many distinct sub-methods, as shown in Fig. 3.

Fig. 3



Framework of plant diseases and pests detection methods based on deep learning

CHALLENGES IN ML/DEEP LEARNING :

To help with data gathering, we now face three major types of challenges: (1) collecting training data, (2) training the model, and (3) ensuring model transferability.

It is critical and difficult to keep algorithm performance optimized when using training data. Research requirements, counting the call for a openly accessible, open-source, joint database on plant stressors at leaf scale, were met by the establishment of the PlantVillage database, which included more than 54 306 pictures of 14 crop species and 26 illnesses. There have previously been numerous studies done on Plant Village data, and more sources of data will be required to build effective models. In addition to the open-source database mentioned above, other open-source databases include a maize NB-lèbre (a common disease) database including 18,222 digital pictures of maize leaves from the field, whether manually taken, placed on a boom, or by unmanned aerial vehicles (UAVs). It's the biggest publicly accessible picture collection annotated for a plant disease. There were about 105 705 NLB lesions annotated by human specialists, making this a huge image set. Over 90% accuracy using 18,000 pictures of agricultural areas in Africa, Latin America, and India, Selvaraj et al. trained a DCNN model for banana sickness and pest classification. By release all of the pictures from their study, the author have permitted the scientific community to contact them. This collection is called the plant disease database (PDDB), and it contains over 2,300 pictures of over 170 plant diseases and other illnesses. Subdividing the pictures to amplify the number of images to 46 513 increased the database size.

CONCLUSION

Plant stress severity phenotyping is a key metric in gauging the risks of crop losses owing to diverse biotic and abiotic stressors. In terms of genetics, better disease-resistant and stress-tolerant genotypes may be identified, and choices about illness management can be assessed. Present methods for measuring stress intensity use the following methods at different scales: Exact count of lesion counts, estimations of the intensity or surface area impacted by a specific stress at covering and field levels, or estimations of the group of species affected by a stress. Image-based disease phenotyping has several benefits, but increasing the speed, accuracy, reliability, and scalability is only possible if ML is used in this process. These concerns include less human error, as well as variability across raters. The purpose is to integrate ML and DL into HTP of plant characteristics in the field for real-time application.

International Research Journal
IJNRD
Research Through Innovation

REFERENCES

[1] Stephane Georges Mallat, Understanding Deep Convolutional Networks, Philosophical Transaction of The Royal Society: A mathematical, physical and Engineering Science 374(2065), 2016.

[2] Sharada Prasanna Mohanty, David Hughes, and Marcel Salath, Using Deep Learning for Image-Based Plant Disease Detection, Front. Plant Sci. 7:1419. doi: 10.3389/fpls.2016.01419, 2016.

[3] Amanda Ramcharan, Kelsee Baranowski, Peter McCloskey, Babuali Ahmed, James Legg, and David Hughes, Deep learning for Image-Based Cassava Disease Detection, Front. Plant Sci. 8:1852 doi: 10.3389/fpls.2017.01852, 2017.

[4] Philipp Lottes, Jens Behley, NivedChebrolu, Andres Milioto, CyrillStachniss, Joint Stem Detection and Crop-Weed Classi_cation for Plant-specific Treatment in Precision Farming, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018.

COMMENTS