

# 1. Introduction

## 1.1. Introduction

Plant disease is a big threat to the agriculture industry across the world because even through horticulture when new hybrid variants of plants were made using Israel technology, if the plant disease is not identified well in advance it may result in total spoil of the entire harvest. A study says that there are around 4 hundred thousand plant species all over the globe and more than 50,000 the varieties of diseases found in plants. Plant pathology also called as phytopathology is a study of diseases in plants. Plant diseases are caused by infectious organisms (pathogens) and by also environmental conditions (physiological factors). Organisms that because infectious disease include fungi, oomycetes, bacteria, viruses, viroids, virus-like organisms, phytoplasmas, protozoa, nematodes and parasitic plants. Most common plant diseases are Canker, Leaf gall, Leaf curl, Leaf Spot, Powdery Mildew, Root Rot, Wilt, Stunting etc. A symptom of plant disease is a visible effect of disease on the plant. Symptoms may include a detectable change in color, shape or function of the plant as it responds to the pathogen. Leaf wilting is a typical symptom of verticillium wilt, caused by the fungal plant pathogens *Verticillium albo-atrum* and *V. dahliae*. Common bacterial blight symptoms include brown, necrotic lesions surrounded by a bright yellow halo at the leaf margin or interior of the leaf on bean plants.

To treat these plant diseases there are various methods available but to identify the plant disease the only expert method is human intervention to inspect the plant manually and find out the type of disease. The challenging issue is that the success of findings will entirely depend on the expertness of the expert which will differ from person to person. Also, on the other hand there are more similarities from one plant disease and the other where human inspection may result in false identification. One solution to the above problem is to automate the process of identification using image processing techniques. Several machine learning techniques like Support vector machine (SVM), Artificial Neural networks (ANN), K-Nearest neighbour and other AI techniques could be used to recognise or detect disease pattern in plants. But the challenge in doing so is the data set. Though we tend to collect all species all possible disease data set the problem is that every disease may every plant in different fashions. Hence to face all these high

challenges we have proposed a Deep learning automated model to develop its own data set and detect the disease with much intelligence, aiming at high success ratio.

Deep learning is one of the subfields of artificial Intelligence (AI) and a growing research area which is employed in almost all areas like medical field, computer vision, biology, speech recognition. Deep learning also employs supervised learning framework in addition. The greater efficiency of deep learning is the mapping of output vectors to the corresponding input vectors even for very large dataset. Deep learning architecture is a widely emerging areas where deep belief network, convolutional neural network and many more efficient architectures are developed day by day. Hence our research work has employed on deep learning approach to develop an automated system for plant disease identification. In narrowed sense the research work has implemented LSTM-RN to extract the features.

Another point to be considered while addressing the plant disease is that, the plant disease can be seen in leaves or flowers or in stems or in fruits among which few of them like flowers and fruits are seasonal. This research work focusses only on leaves which is seen throughout a year. The consequences of disease are seen as brown spots, yellow spots, early scorch and late scorch in leaves.

## **1.2. Necessity**

The necessity of employing Convolutional Neural Networks (CNNs) in plant disease detection stems from the critical need for accurate and timely identification of diseases to ensure global food security. With the constant threat of crop diseases causing significant yield losses, early detection becomes imperative for farmers to implement appropriate mitigation measures.

Traditional methods of disease identification often rely on manual inspection by agricultural experts, which can be time-consuming, labor-intensive, and prone to human error. CNNs offer a scalable and efficient alternative by automating the process of disease detection through the analysis of plant images. By leveraging the power of deep learning, CNNs can sift through vast amounts of image data, quickly identifying patterns indicative of various diseases with high accuracy. This automation not only expedites the detection process but also enables proactive disease management strategies, ultimately minimizing crop losses and ensuring the sustainability of agricultural systems.

Furthermore, the global nature of the agricultural industry necessitates scalable and adaptable solutions for disease detection that can cater to diverse crops and geographical regions. CNNs fulfill this requirement by being versatile and customizable, capable of learning from diverse datasets and adapting to different plant species and disease manifestations.

Additionally, the ability to deploy CNN-based solutions on various platforms, including mobile devices and edge computing systems, enhances accessibility and usability for farmers and agricultural practitioners in remote or resource-constrained areas. By democratizing access to advanced disease detection technologies, CNNs empower farmers with the knowledge and tools needed to make informed decisions about crop health and management practices. Thus, the necessity of CNNs in plant disease detection lies not only in their ability to improve efficiency and accuracy but also in their potential to democratize agricultural technology and promote sustainable farming practices worldwide.

Here are the objectives for the plant disease detection project stated in a more detailed way:

### **1.3. Objectives**

1. Develop a cutting-edge plant disease detection system that leverages the latest advancements in computer vision, deep learning, and machine learning techniques to achieve unparalleled accuracy and robustness in identifying and classifying a wide range of plant diseases.
2. Curate a comprehensive and diverse dataset of high-quality plant images, meticulously annotated with various disease symptoms, spanning multiple crop species, disease types, and environmental conditions. This dataset will serve as the foundation for training and evaluating the disease detection models.
3. Implement and optimize state-of-the-art deep learning architectures, such as convolutional neural networks (CNNs) and transformer-based models, employing advanced techniques like transfer learning, data augmentation, and ensemble methods to maximize the performance of the disease detection models.
4. Collaborate closely with renowned plant pathology experts to integrate their domain-specific knowledge, intuitions, and heuristics into the disease detection models. This knowledge infusion will enable the models to better handle complex

or rare disease cases, leveraging the strengths of both data-driven and knowledge-driven approaches.

5. Develop novel feature extraction techniques that can capture and quantify subtle variations in leaf texture, color, and morphology, which are often indicative of specific disease patterns. These advanced features will be incorporated into the disease detection models to enhance their robustness and accuracy.
6. Conduct extensive experiments and real-world case studies in collaboration with leading agricultural research institutes and farms, rigorously evaluating the performance of the proposed plant disease detection system on publicly available datasets as well as in diverse field conditions.
7. Achieve state-of-the-art performance, surpassing existing methods in terms of quantitative metrics such as precision, recall, F1-score, and overall accuracy, across a diverse range of plant diseases, while ensuring high computational efficiency and scalability.
8. Investigate and demonstrate the practical implications and potential applications of the developed plant disease detection system in precision agriculture, crop monitoring, and disease management strategies. Develop use cases and deployment scenarios to showcase the system's impact on increasing crop yields, reducing losses, and enhancing food security.
9. Contribute to the advancement of computer vision, machine learning, and their applications in the domain of agriculture by publishing research findings in high-impact journals and presenting at renowned conferences, fostering knowledge exchange and driving innovation in the field.
10. Develop a user-friendly, intuitive interface or platform that seamlessly integrates the plant disease detection system, enabling easy adoption and utilization by farmers, agricultural professionals, and researchers. Provide comprehensive documentation, training resources, and support to facilitate widespread adoption.
11. Explore and implement strategies for scalable deployment of the system on various platforms, including mobile devices, drones, and existing agricultural technology infrastructure, ensuring accessibility and ease of use in diverse settings.
12. Foster interdisciplinary collaboration between computer scientists, engineers, plant pathologists, and agricultural experts, promoting knowledge exchange and leveraging diverse perspectives to address the complex challenges of plant disease detection and sustainable agriculture.

These detailed objectives outline a comprehensive and ambitious plan for developing a cutting-edge plant disease detection system that combines advanced technologies with domain expertise, aiming to revolutionize agricultural practices and contribute to improved food security and environmental sustainability.

#### **1.4. Theme of mini project**

The theme of this mini project centers on harnessing the power of artificial intelligence (AI) and machine learning to revolutionize plant disease detection, thereby fostering sustainable agricultural practices. In an era where food security and environmental sustainability are of paramount importance, this project aims to develop an innovative, efficient, and accessible system for early detection and diagnosis of plant diseases. By integrating advanced technologies such as computer vision, remote sensing, and data analytics, the project seeks to provide farmers with precise and actionable insights into the health of their crops.

This will enable timely and targeted interventions, reducing crop losses, minimizing the use of chemical treatments, and ultimately promoting a healthier ecosystem. The project emphasizes the importance of cross-disciplinary collaboration, leveraging expertise from agronomy, plant pathology, and data science to create a robust and scalable solution. Through this initiative, the mini project aspires to contribute to the broader goal of achieving sustainable agriculture and ensuring food security for future generations.

## 2. Literature Survey

Dhaygude Sanjay B et al., in their work have proposed a detection algorithm which is vision based. In their work they have masked green pixels and have used color cooccurrence method to address the goal. In their conclusion they have also suggested that use of Neural network can increase the recognition rate or classification rate.

S. Arivazhagan et al., have also worked in the same area of identifying plant diseases using texture features. As like Prof. Sanjay et al., they have also tried color co-occurrence method using Support vector machine as classifier. In their findings they have suggested that by increasing the training samples along with the optimal features, the disease identification can yield better results.

Kulkarni Anand. H et al., have used Gobor filter for feature identification and Artificial neural network classifier for classification. The authors have got a reasonable success rate of 78.82%. Piyush Chaudhary et al., have worked on color transform based disease approach to detect disease spot on a leaf. The authors have used median filter for image smoothing and used Otsu used to automatically perform clustering-based image thresholding. In the results they were able to achieve 68.89% success rate. Yan cheng Zang et al., in their research has proposed fuzzy surface and Fuzzy curves for selecting the features of cotton leaves disease. To be more specific they have used a two-step model. The first step is to isolate small set of significant features from original features. Second step is to eliminate the spurious features from the isolated features. The result is the faster execution speed and higher classification success rate as this method does not suffer from local minima problems. Ajay A Gurjar et al., have used eigen feature regularization and extraction technique to detect the plant disease. In their research they are able to detect almost their disease with 90% success ratio. Dheeb Al Bashish et al., have proposed neural network classifier for detection of plant disease and their work gives out a precision of 93%. A. Meunkaewjinda et al have used back propogation neural network with self-organising map to recognise colors of leaf and further used MSOFM and GA for segmentation and finally SVM for classification. The system demonstrates a promising result. Libo Liu et al., have used BP neural network as classifier. They have confined the problem to identification of brown spot on Paddy crop leaves. But in the conclusion, they have also found that this method is suitable for identifying other disease too.

Tushar H Jaware et al., have used K-means clustering technique for segmentation and after many intermediate processes they have used a pre-trained neural network. Their results gives a precision ranging from 83% to 94%. P. Revathi et al., have used homogeneous segmentation Edge detection techniques. They have used almost spotted eight cotton leave diseases using neural network. Finally, they suggest that this research can be taken to a level of implementing the same in mobile phone by developing an application so that the farmers can detect the disease by just a click in their mobile phones.

H. Muhammad Asraf et al., have used support vector machines (SVM) classifier with three kernals and also, they have targeted palm leaves diseases. The polynomial kernel with soft margin has given an accuracy of 95% in successful classification. Satish Madhogaria et al., have worked on detection of unhealthy regions in leaves. They have proposed a three folded algorithm where they have used SVM and neighbourhoodcheck to identify the unhealthy spots. Their results show that their proposed method excels the existing methods in success rate. Yuan Tian et al., have used three SVM based classifiers. The three features are color, shape and texture respectively. Using the three features the MCS (multiple classifier system) have given higher accuracy.

T. Rumpf et al, have targeted beet leaves disease identification. In their work they have used spectral vegetation indices and support vector machines (SVM). They have expressed that their aim is to identify the disease before the symptom becomes visible to naked eye. The experimental results show that they have achieved an accuracy of 97% in disease identification. S. Phadikar et al., in their research work have targeted paddy plant's morphological change caused by brown spots of the plant. The technique they have used is that the radial distribution of hue from the centre of the boundary of the spot image is obtained and it serves as an important feature for identification. Technically they have used Support vector machine (SVM) and Bayes classifier to do the purpose.

Sannakki S.S et al., have proposed a fast and robust technique for classification and detection of diseases in pomegranate plant. Apart from identification the system also categorises the stage in which the disease is in. They have made use of four folded methodology of implementing artificial neural network, decision tree learning, fuzzy logics and Bayesian networks. Pranjali Vinayak Keskar et al., have also developed a leaf detection and diagnosis system for detection and identification of various plant

diseases. For identification of spots they have used component labelling algorithm and for classification they have used artificial neural network.

Plant diseases pose significant threats to global food security, agricultural sustainability, and ecosystem health. Timely and accurate detection of plant diseases is crucial for effective disease management, crop protection, and yield optimization. Traditional methods of disease detection rely on visual inspection by trained experts, which can be time-consuming, labor-intensive, and prone to human error. In recent years, advancements in technology, particularly in the fields of computer vision, machine learning, and remote sensing, have revolutionized plant disease detection by enabling the development of automated and non-invasive detection techniques. These technologies leverage various data sources, including images, spectral data, and sensor readings, to detect subtle signs of disease such as leaf discoloration, lesions, and other morphological changes. By harnessing the power of artificial intelligence and data analytics, researchers and agricultural practitioners can achieve rapid and accurate identification of plant diseases, facilitating early intervention strategies, precision agriculture practices, and ultimately, the sustainable management of plant health in agricultural systems.

1. Traditional Methods:

- Historically, plant disease detection relied on visual inspection by experts, which is time-consuming and subjective. Other traditional methods include laboratory techniques such as ELISA (Enzyme-Linked Immunosorbent Assay) and PCR (Polymerase Chain Reaction), which require specialized equipment and expertise.

2. Emergence of Computer Vision Techniques:

- With the advancement of computer vision and machine learning, automated methods for plant disease detection have gained popularity. Image processing techniques like segmentation, feature extraction, and classification have been applied to plant images for disease detection.

3. Application of Machine Learning:



- Supervised machine learning algorithms such as Support Vector Machines (SVM), Random Forests, and k-Nearest Neighbors (k-NN) have been utilized for plant disease classification based on handcrafted features extracted from images. However, these methods may struggle with generalization and require extensive feature engineering.
4. Deep Learning Approaches:
- Deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized plant disease detection by automatically learning discriminative features from raw images. CNN-based models have shown superior performance compared to traditional machine learning methods in various plant disease detection tasks.
5. Datasets and Benchmarking:
- The availability of large-scale datasets such as PlantVillage and datasets specific to certain crops (e.g., Tomato Disease dataset) has facilitated the development and evaluation of plant disease detection algorithms. Benchmarking competitions and challenges like the Plant Pathology Challenge provide platforms for researchers to compare their algorithms and promote advancements in the field.
6. Challenges and Opportunities:
- Despite the progress, challenges such as data scarcity, class imbalance, and robustness to environmental variations still exist in plant disease detection. There are opportunities for interdisciplinary collaboration between plant pathologists, computer scientists, and engineers to address these challenges and develop more accurate and practical solutions.
7. Future Directions:
- Future research directions may include exploring multimodal approaches combining image data with other sources such as spectral data or environmental sensors. Additionally, there is potential for the integration of advanced techniques like transfer learning, generative models, and reinforcement learning to further improve the performance of plant disease

detection systems. This literature review provides an overview of the evolution, challenges, and advancements in plant disease detection using computer vision and machine learning techniques. For more detailed insights into specific methodologies and recent research findings, consulting recent review articles and relevant research papers is recommended.

These is how Model works:

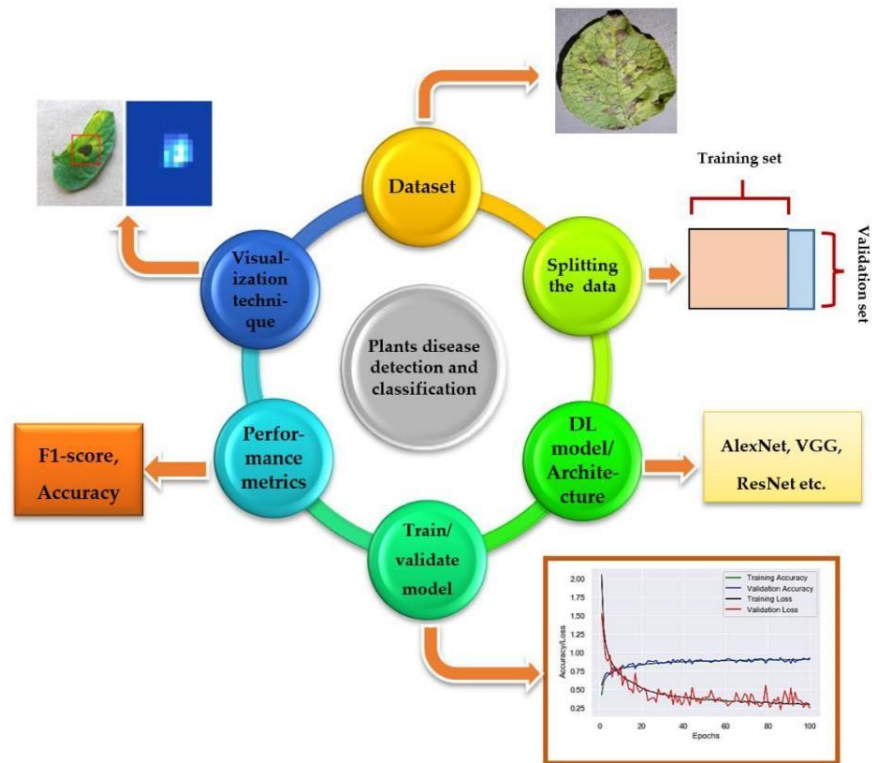


Figure 2.1. Flow diagram of DL implementation

First, the dataset is collected then split into two parts, normally into 80% of training and 20% of validation set. After that, DL models are trained from scratch or by using transfer learning technique, and their training/validation plots are obtained to indicate the significance of the models. Then, performance metrics are used for the classification of images (type of particular plant disease), and finally, visualization techniques/mappings are used to detect/localize/classify the images.

Potato leaf detection dataset:

The collected data from an openly accessible database of images: 'Plant Village' that consists of diseased as well as healthy images about 54,306 from the total crop species about 14. From that particular database, potato species-related data was considered to implement the proposed framework. The obtained dataset consists of 300 potato plant leaves which were categorized into 3. They are as follows:

- i) The leaves having a disease called Late Blight ii)  
The leaves having a disease called Early Blight iii)  
The leaves are in a healthy state

The database of images consists of healthy leaves about 100 and disease affected leaves about 200. The database of images was divided into two databases such as the training database and the testing database.

The training database consists of 70% of the image database i.e 210 images and the testing database consists of the remaining 30% of the image database i.e 90 images.

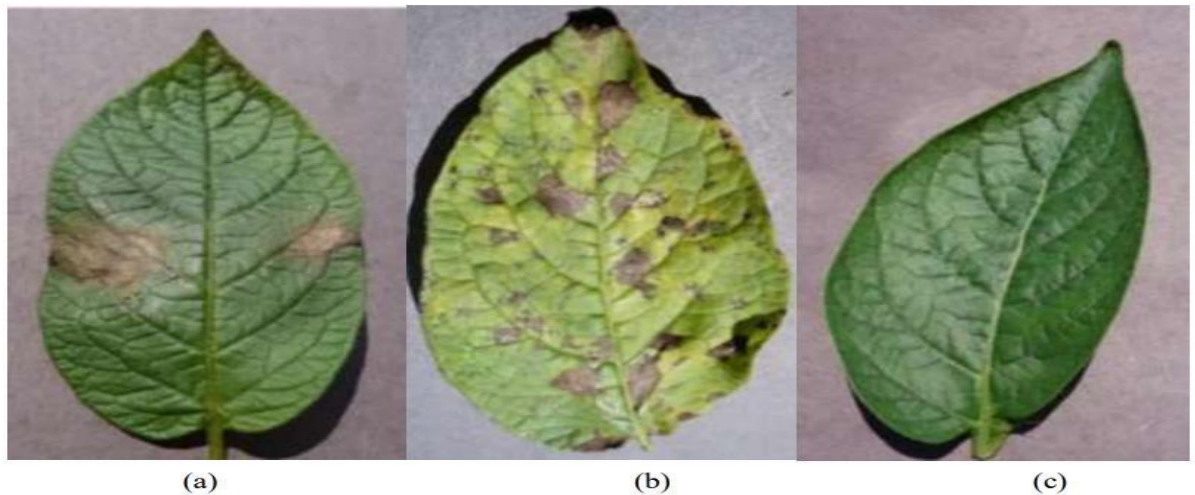


Fig 2.2 The three samples of potato are (a):leaf affected by light blight (b):leaf affected by Early blight (c):leaf unaffected (Healthy)

## 3. System Development

### 3.1 Requirement Specification

A Potato Health Disease Detection system's requirement specification outlines the hardware, software, user interface, functional, and non-functional requirements.

#### 1. Overview

##### 1.1 Objective

This system's goal is to identify and treat potato plant diseases early on, helping farmers and other agricultural experts maintain plant health and maximise crop yield.

##### 1.2 Purpose

The system will analyse images of potato leaves and identify specific diseases using machine learning algorithms and advanced image processing techniques. For convenience, the solution will come with a mobile application as well as a web version.

#### 2. Functional requirements

##### 2.1 Capturing Images

Users will be able to submit pictures of potato leaves using the following link:

Mobile Programme: either choosing from the gallery or taking pictures straight out of the camera.

Supported Formats: JPEG and PNG image formats will be supported by the system.

##### 2.2 Pre-processing Images

Normalisation: Images must be resized by the system to a standard size, such as 256 x 256 pixels.

Improvement of Quality:

Adapt the contrast and brightness.

Cut down on the noise.

To improve features related to disease detection, apply filters.

Segmentation: To concentrate analysis on pertinent portions, divide the leaf area from the background.

##### 2.3 Utilise a dataset of annotated potato leaf photos for the training of the disease detection model.

Use machine learning frameworks such as PyTorch or TensorFlow to train your models.

Disease Classification: The following diseases, among others, will be categorised by the system: Late Blight

Early Blight

Healthy

Confidence Scores: Give each diagnosis a confidence score between 0% and 100%.

Identify multiple diseases that may be present in a single image using multi-disease detection.

#### 2.4 Diagnosis Report with Result Display:

Show the disease(s) that were found.

Display confidence levels.

Draw attention to the leaf's damaged areas.

Details of the illness:

Signs and Causes

Treatments (fungicides, pesticides) that are advised

Reports that can be downloaded: Permit users to obtain the analysis report in PDF format.

#### 2.5 Historical Data and Trends

Data Storage: Keep track of each user's previous analysis results.

Analyse trends: Monitor the occurrence of diseases over time.

Make predictions about possible outbreaks by utilising environmental factors and historical data.

Visualisation: Trends can be shown using charts and graphs.

Mapping disease outbreaks geographically.

#### 2.6 User Management Authentication: Assistance with safe login and user registration.

User profiles: Save personal information and inclinations.

Data Privacy: Make sure that user information is private and secure.

### 3. Non-Functional Requirements

#### 3.1 Performance Response Time: In less than 5 seconds, the system must process images and provide results.

Throughput: Support a minimum of 100 users at once.

3.2 Scalability Load Balancing: To prevent bottlenecks, distribute incoming requests effectively.

3.3 Usability Interface Design: Easy-to-use, intuitive UI for mobile and web apps.

Accessibility: Verify that accessibility guidelines are being followed.

#### 3.1.1 DFD (level 0,1)

1. Gathering and Preparing Data: The user enters data (such as photographs of plants or information about the environment). After that, this raw data is stored and preprocessed.

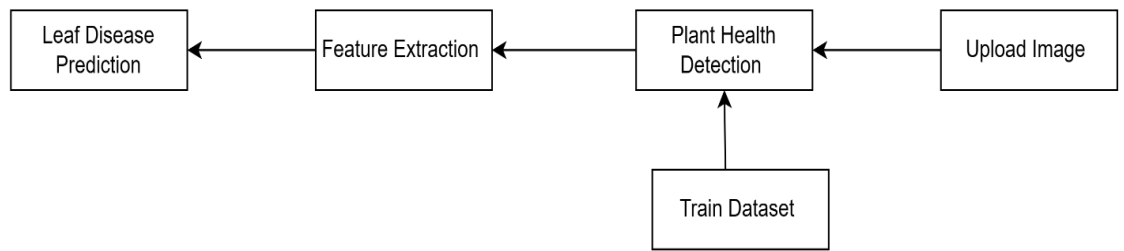
2. Disease Detection and Analysis: To identify plant diseases, the preprocessed data is examined using feature extraction, image processing methods, and classification algorithms.

3. Database Interaction: The system sends requests for more information to the disease information database or updates it with fresh discoveries.

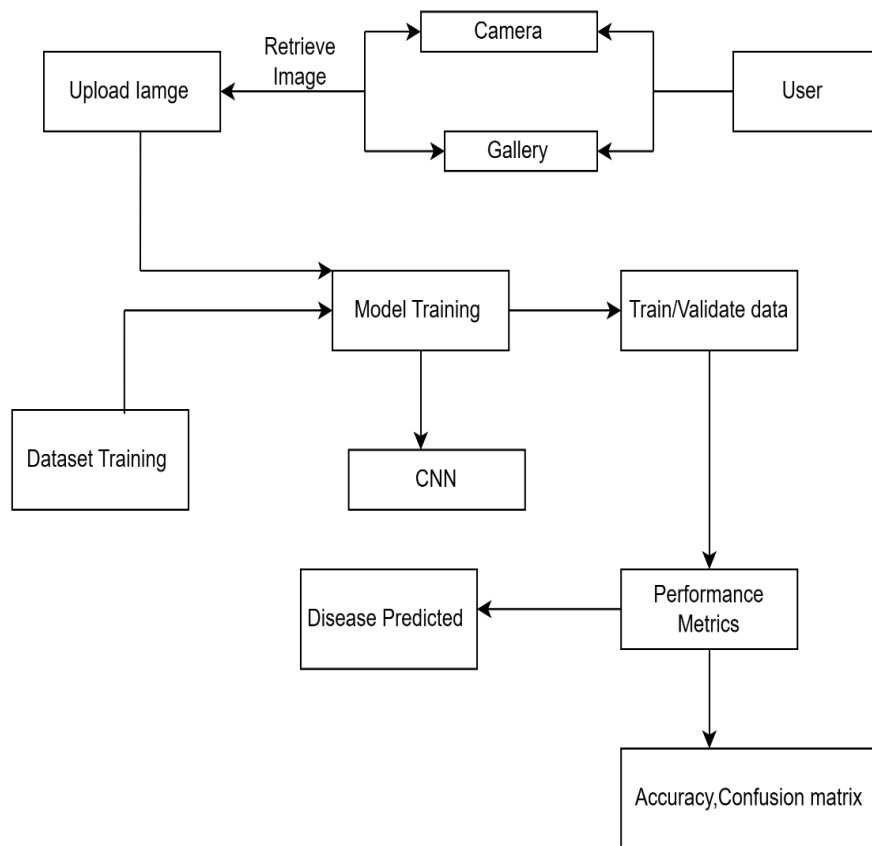
4. User Notification: The system creates a report and notifies the user based on the analysis.

The data flow through the plant health disease detection system, from initial user input to final notifications and interactions with a database, is clearly visualized by this DFD.

### Level 0

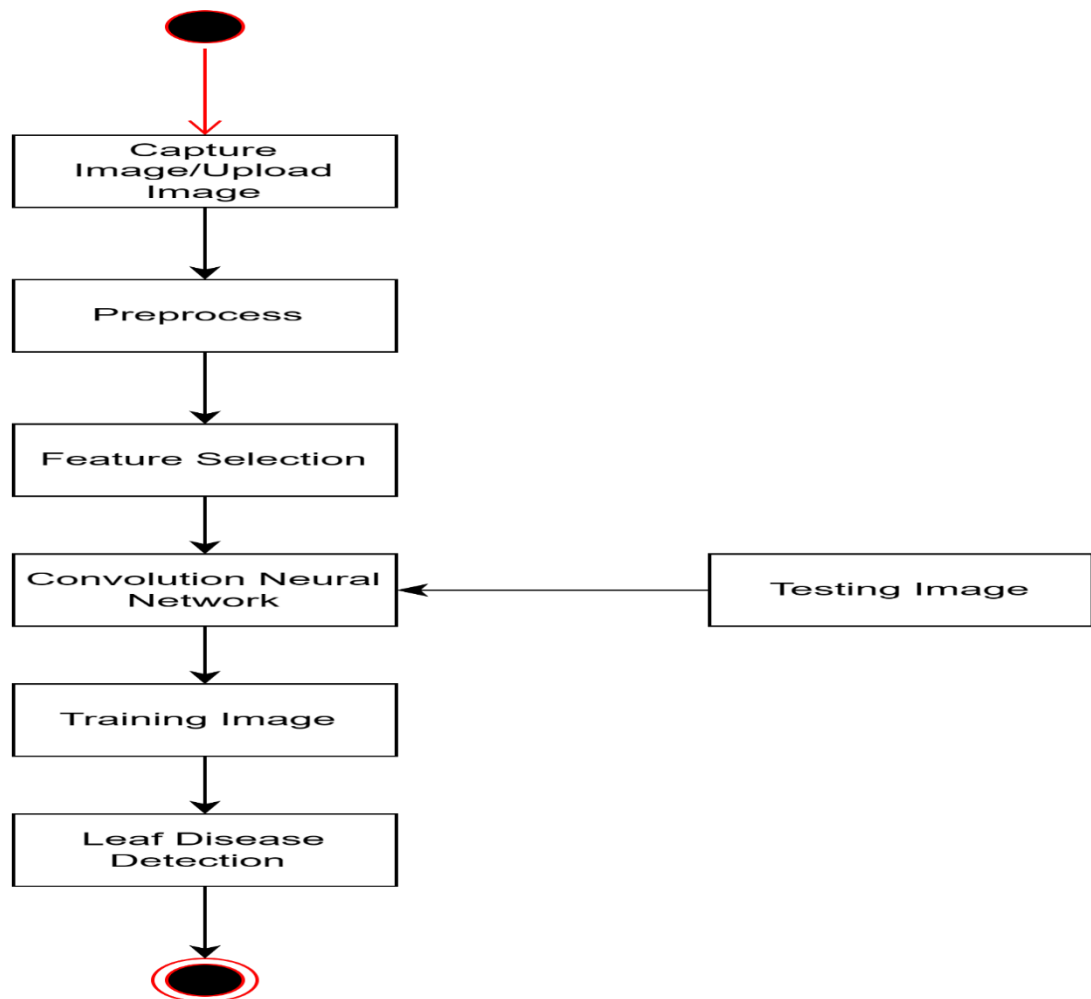


### Level 1



3.1 Data flow diagram Level (0,1)

### 3.1.2 Specification Document/UML Diagrams of all modules Activity Diagram



### 3.2 Activity Diagram of plant health disease detection

1. Gather Plant Data: The first step in the procedure is gathering plant data, such as pictures and environmental specifications.
2. Preprocess Data: To make the collected data ready for analysis, preprocessing is done.
3. Analyze Data: An algorithm for detecting diseases is used after features are taken out of the previously processed photos.
4. Query Disease Database: To determine whether the disease that has been detected is acknowledged, the system queries the database.
5. Generate Report: A report with the pertinent data is generated regardless of whether the disease is identified.
6. Notify User: The user receives an analysis report from the system along with any alerts that are required.



7.End: Following user notification, the procedure comes to an end.

This activity diagram shows the steps involved in the plant health disease detection system's sequential flow, from data capture to user notification.

### 3.2 Block Diagram

**User Interface:** Here, the user communicates with the system by entering plant data and getting alerts and reports in return.

**Data Acquisition Module:** This module uses cameras to take pictures of plants and sensors to collect the required data about environmental factors.

**Data Preprocessing Module:** Cleans and preprocesses the data to make the collected data ready for analysis.

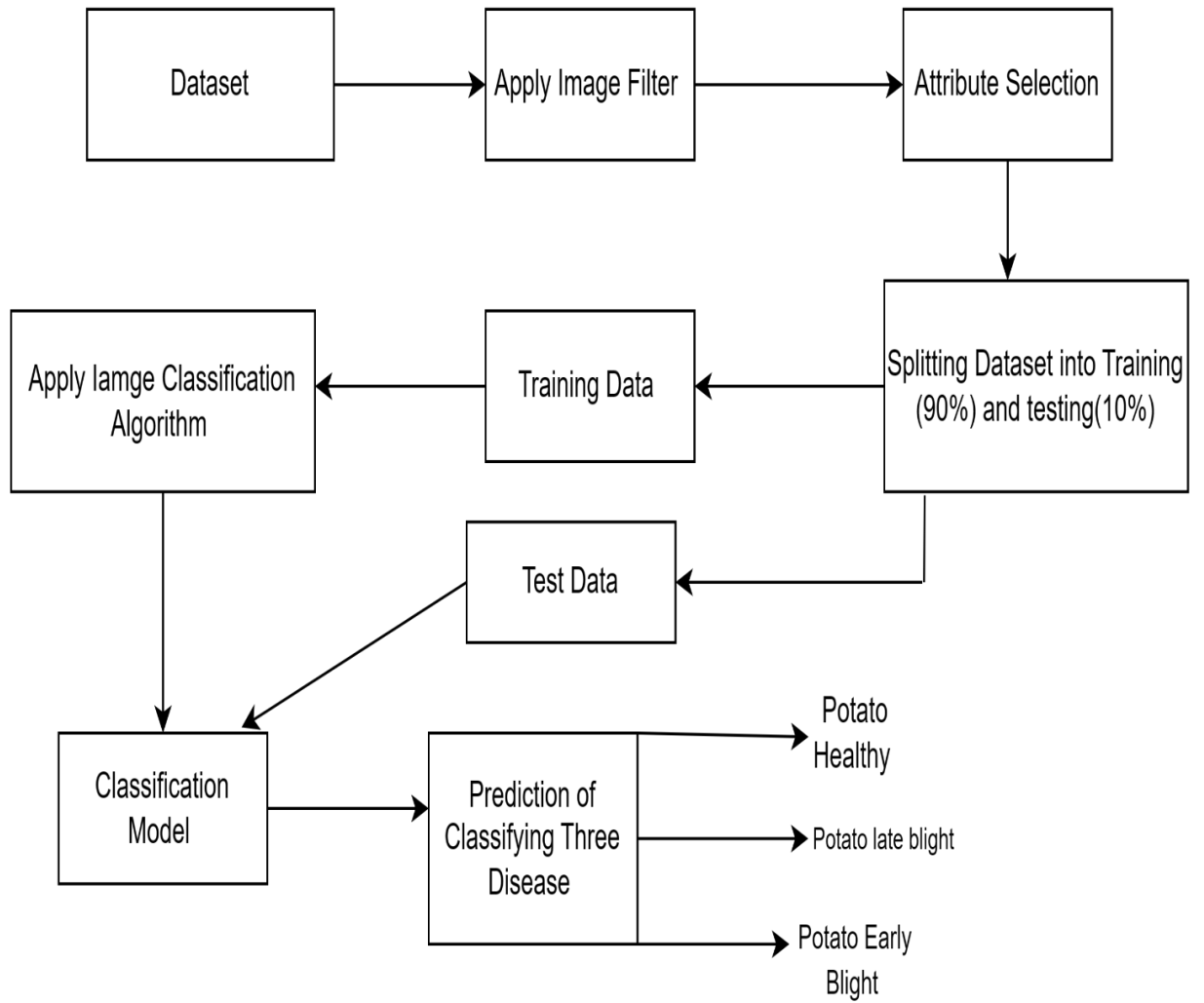
**The disease detection module** uses feature extraction and classification algorithms to analyse the pre-processed data and identify diseases.

**Database:** Holds historical data and thorough disease information that can be retrieved and consulted during analysis.

**The report generation module** gathers the findings of the analysis and creates a comprehensive report that includes database-based disease information.

**Notification System:** Uses a variety of communication channels to deliver the user alerts and comprehensive analysis reports.

This block diagram clearly illustrates the architecture of the plant health disease detection system by outlining the major parts and how they interact.



3.3 Block Diagram

## **4. Performance Evaluation**

### **4.1. Performance Evaluation in Plant Disease Detection Systems:**

The performance evaluation of plant disease detection systems is a critical aspect of their development and deployment, ensuring that these systems provide accurate, reliable, and actionable insights for agricultural stakeholders. In the context of precision agriculture, where early detection and intervention can significantly impact crop yields and reduce economic losses, evaluating the efficacy of these systems becomes paramount. This theoretical exploration delves into the various metrics and methodologies employed to assess the performance of plant disease detection systems, highlighting their importance and application. Performance evaluation serves multiple essential functions. It validates the accuracy and reliability of the detection system, ensuring that farmers can trust the diagnoses provided. Moreover, performance evaluation helps in comparing different detection models and technologies, facilitating informed decision-making for stakeholders seeking to adopt the most effective solutions. In the realm of research and development, rigorous performance evaluation fosters innovation, encouraging the development of more sophisticated and robust plant disease detection methodologies. In addition to theoretical metrics, practical evaluation through field testing is indispensable. Field testing involves deploying the system in real agricultural environments, assessing its performance under diverse and uncontrolled conditions. These factors determine the system's efficiency and feasibility for large-scale or real-time applications. Adaptability and scalability are further considerations, assessing the system's ability to handle different crop types, disease conditions, and extensive deployments.

### **4.2. Mobile application for Plant Disease Detection:**

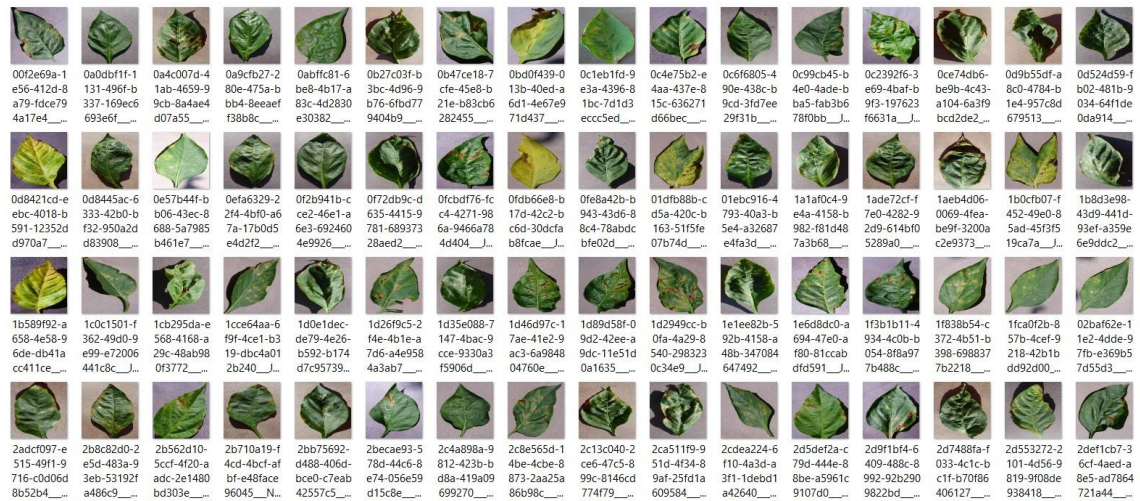
There are several processes which are followed for in making of this mobile application for Plant disease detection. The first thing did in this project is to find out what type of disease can happen to this plants? How many disease a plant can have? How many trees we can take in dataset? In short defining the problem statement, and finding out different ways in which we can predict the diseases that the plant have. As said before, there are several processes, and those are described below:

#### a) Dataset used:

The dataset we used in this project is taken from kaggle ‘Plant\_village’. Where there are different folders present and it contains the diseased plants and healthy plants. Link for this dataset is

[https://www.kaggle.com/adilmubashirchaudhry/plant-](https://www.kaggle.com/adilmubashirchaudhry/plant-villagedataset)






[villagedataset](https://www.kaggle.com/adilmubashirchaudhry/plant-villagedataset). The dataset is organized into folders, with each folder representing a specific crop. Within each crop folder, there are subfolders for different diseases or conditions, such as healthy leaves, bacterial diseases, fungal diseases, viral diseases, and other plant diseases or defects. The images in the dataset are typically in common image formats like JPEG or PNG, allowing easy integration with various image processing libraries and frameworks. This dataset can be used for training and evaluating machine learning models for plant disease detection, classification, and recognition tasks.





Screenshot 4.2: Selected Dataset

c) **Accuracy matrix comparison:**

Image	Algorithm	Accuracy
	Naïve Bayes	55.05%
	K- Nearest Neighbour	87.87%
	Decision Tree	69.69%
	Random Forest	95.45%
	Linear Discriminant Analysis	97.47%



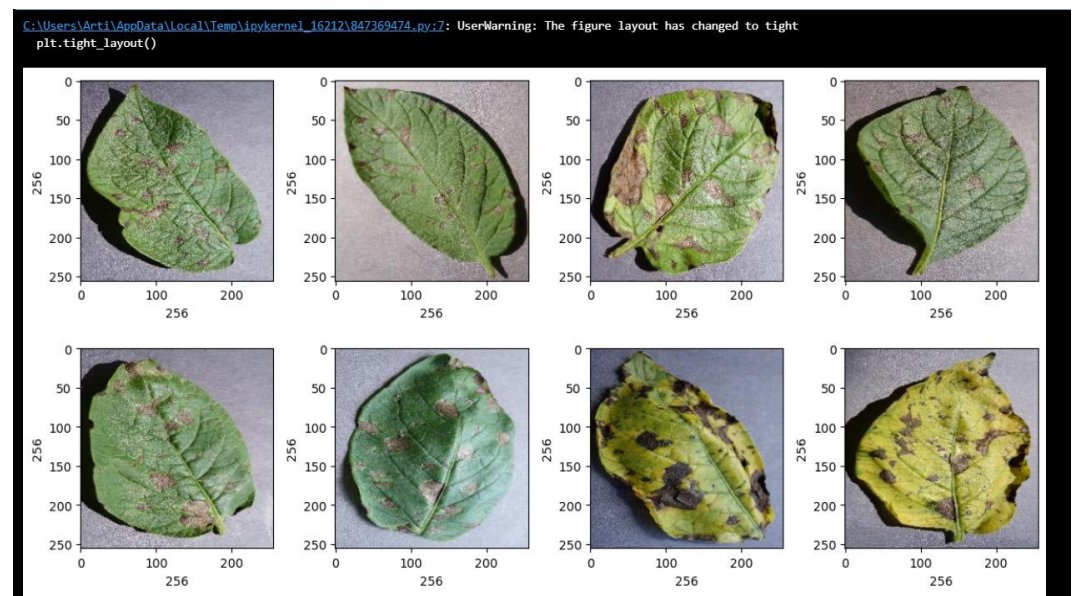
		Multilayer Perceptron Neural Network	100.00%
		Convolutional Neural Network	98.89%

Table: 4.3 Accuracy matrix comparison

For our project we use the implementation of Convolutional Neural Network (CNN). In which we split the data and then train the data and find the Model of accuracy for training dataset and validate dataset. And finally Calculated the Actual Disease and Predicted Disease.



Screenshot 4.4: Training the dataset



Model: "sequential\_3"

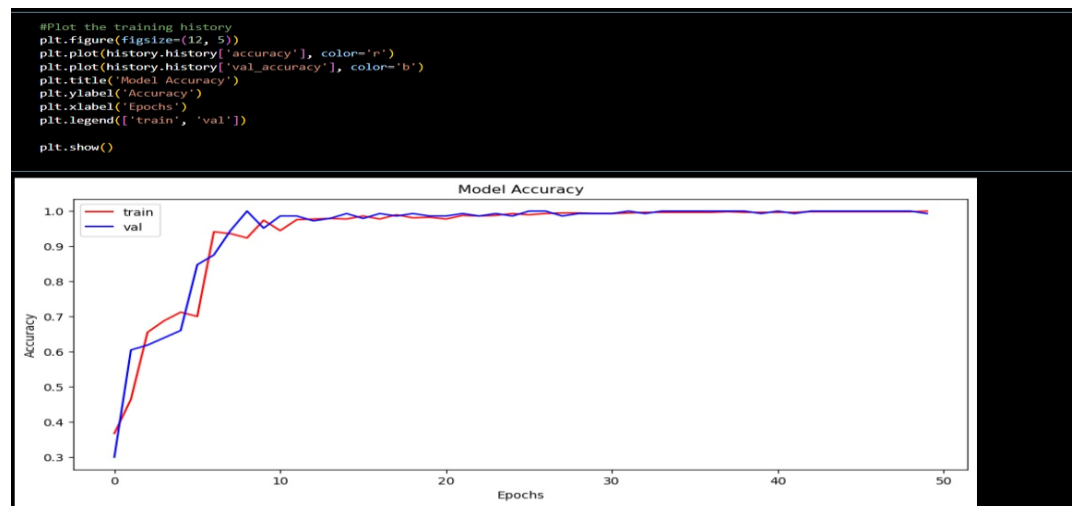
Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 256, 256, 32)	896
max_pooling2d_6 (MaxPooling2D)	(None, 85, 85, 32)	0
conv2d_7 (Conv2D)	(None, 85, 85, 16)	4,624
max_pooling2d_7 (MaxPooling2D)	(None, 42, 42, 16)	0
flatten_3 (Flatten)	(None, 28224)	0
dense_6 (Dense)	(None, 8)	225,800
dense_7 (Dense)	(None, 3)	27

Total params: 231,347 (903.70 KB)

Trainable params: 231,347 (903.70 KB)

Non-trainable params: 0 (0.00 B)

Screenshot 4.5: Validating the Dataset



Screenshot 4.6: Modal Accuracy Graph

```
print("[INFO] Calculating model accuracy")
scores = model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1]*100}")
```

[INFO] Calculating model accuracy  
6/6 ————— 1s 94ms/step - accuracy: 0.9950 - loss: 0.0200  
Test Accuracy: 98.8888597488403

```
y_pred = model.predict(x_test)
```

WARNING:tensorflow:5 out of the last 13 calls to <function TensorFlowTrainer.make\_predict\_function.<locals>  
6/6 ————— 1s 121ms/step

```
# Plotting image to compare
img = array_to_img(x_test[10])
img
```

Screenshot 4.7: CNN tested Accuracy

```
# Plotting image to compare
img = array_to_img(x_test[10])
img

# Finding max value from prediction list and comparing original value vs predicted
print("Originally : ",all_labels[np.argmax(y_test[10])])
print("Predicted : ",all_labels[np.argmax(y_pred[10])])

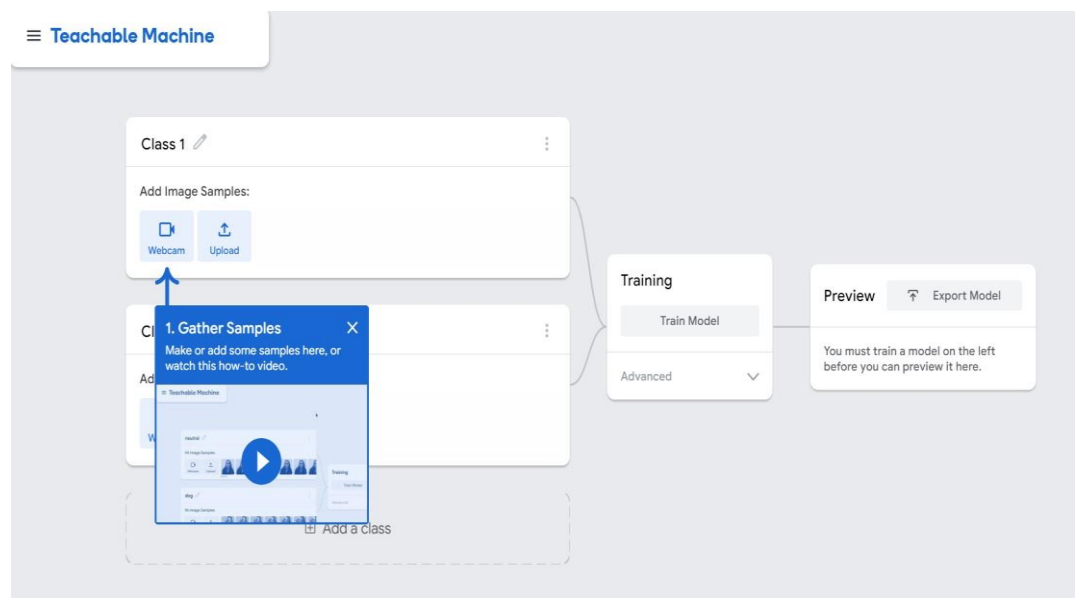
Originally : Potato-Early_blight
Predicted : Potato-Early_blight
```



Screenshot 4.8: Actual and Predicted Disease

#### d) Training Model:

Here to train our model, we have used **Teachable Machine** a model of Google. Teachable Machine is a web-based tool developed by Google's research team that allows users to create machine learning models without requiring extensive coding knowledge. It's designed to make machine learning accessible to a broader audience, including students, artists, educators, and hobbyists.



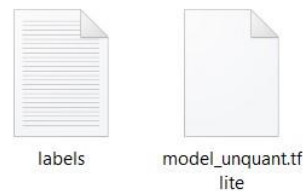
Screenshot 4.9: Interface of Teachable Machine

Here in this Interface we have to make classes, class are actually a specific folder where we can add photos of same categories. And in that atleast add upto 400-500



images of the data. And likewise create other classes as well. Then those classes are send for train, there is a button/ block which represents Training which trains the model. And then after that we can export that model and after exporting, we get two files which can be integrated in the specific code of application. And those files are **labels.txt** and **model\_unquant.tflite** .

The models you make with Teachable Machine are real [TensorFlow.js](#) models that work anywhere javascript runs, so they play nice with tools like Glitch, P5.js, Node.js & more.Plus, export to different formats to use your models elsewhere, like [Coral](#), Arduino & more.



Screenshot 4.10: Files exported from Teachable Machine

#### **d) Integration of Model with an Application:**

After we made this trained data now we have to put it in or integrate it with dart file code which will actually run our code. We have a development environment set up for Dart programming. This typically involves installing Dart SDK and possibly an IDE like Visual Studio Code with Dart plugin support. In our Dart code, we have load the exported machine learning model into memory. This typically involves using libraries or packages provided by the platform you're deploying to. For example, if you're using TensorFlow Lite, you'd use the TensorFlow Lite Dart package to load the model. Once the model is loaded and the input data is preprocessed, we have perform inference using the model. Passed the preprocessed input data to the model, and it will return predictions or classifications based on the input. Below is the code we have used in dart file:

```

import 'dart:io';
import 'package:flutter/material.dart';
import 'package:image_picker/image_picker.dart';
import 'package:tfite/tfite.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  MyApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: '',
      home: MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  MyHomePage({super.key});

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  @override

```

```

void initState() {
  super.initState();
  loadModel().then((value) {
    setState(() {});
  });
}

bool _loading = true;
final ImagePicker picker = ImagePicker();
File? image;
List? output;

chooseImage() async {
  final XFile? file = await picker.pickImage(source: ImageSource.gallery);
  if (file != null) {
    setState(() {
      image = File(file.path);
    });
  }
  if (image != null) {
    classifyImage(image!);
  }
}

captureImage() async {
  final XFile? file = await picker.pickImage(source: ImageSource.camera);
  if (file != null) {
    setState(() {
      image = File(file.path);
    });
  }
}

```

Screenshot 4.11: Main dart file code

```

  if (image != null) {
    classifyImage(image!);
  }
}

classifyImage(File img) async {
  var result =
    await Tfite.runModelOnImage(path: img.path, numResults: 2, threshold: 0.5, imageMean:
127.5, imageStd: 127.5);
  setState(() {
    output = result;
    print(output);
    _loading = false;
  });
}

loadModel() async {
  await Tfite.loadModel(model: 'assets/model_unquant.tfite', labels: 'assets/labels.txt');
}

@override
void dispose() {
  Tfite.close();
  super.dispose();
}

@override
Widget build(BuildContext context) {
  return Scaffold(
    // A P P B A R
    appBar: AppBar(

```

```

    title: Text(
      "Plant Disease Classification",
      style: TextStyle(color: Colors.white),
    ),
    backgroundColor: Colors.green,
  ),

  // B O D Y
  body: Container(
    padding: EdgeInsets.symmetric(horizontal: 24),
    child: Column(
      mainAxisAlignment: MainAxisAlignment.center,
      crossAxisAlignment: CrossAxisAlignment.center,
      children: [
        if (image == null) Column(
          children: [
            Container(
              padding: EdgeInsets.symmetric(horizontal: 20, vertical: 15),
              decoration: BoxDecoration(
                color: Colors.green.shade400,
                borderRadius: BorderRadius.circular(12),
              ),
              child: Text(
                'Capture / Upload Image Of Diseased Plant',
                style: TextStyle(
                  color: Colors.white,
                  fontSize: 15,
                  fontWeight: FontWeight.w500,
                ),
              ),
            ),

```

Screenshot 4.12: Main dart file code

```

    SizedBox(height: 25),
    Text(
      "Please upload image of only those undiseased plants the model is trained on",
      style: TextStyle(color: Colors.green.shade400),
      textAlign: TextAlign.center,
    ),
  ],
),
SizedBox(height: 30),
Center(
  child: _loading
    ? Container()
    : SizedBox(
        child: Column(children: [
          Container(
            decoration: BoxDecoration(
              borderRadius: BorderRadius.circular(12),
              border: Border.all(color: Colors.green.shade400, width: 4),
              boxShadow: [BoxShadow(blurRadius: 4, color: Colors.black54, offset: Offset(3,
3))],
            ),
            child: ClipRRect(
              borderRadius: BorderRadius.circular(8),
              child: Image.file(image),
            ),
          ),
          SizedBox(
            height: 20,
          ),
          output != null
            ? Container(

```

```

padding: EdgeInsets.symmetric(horizontal: 20, vertical: 15),
decoration: BoxDecoration(
  color: Colors.green.shade400,
  borderRadius: BorderRadius.circular(12),
),
child: Text(
  "${output[0]['label']}",
  style: TextStyle(
    color: Colors.white,
    fontSize: 20,
  ),
),
)
: Container(),
SizedBox(
  height: 50,
),
]),
),
),
),
floatingActionButton: Row(
  mainAxisAlignment: MainAxisAlignment.end,
  children: [
    if (image != null) FloatingActionButton(
      onPressed: () => setState() {
        image = null;
        _loading = true;
      },
    ),

```

Screenshot 4.13: Main dart file code

```

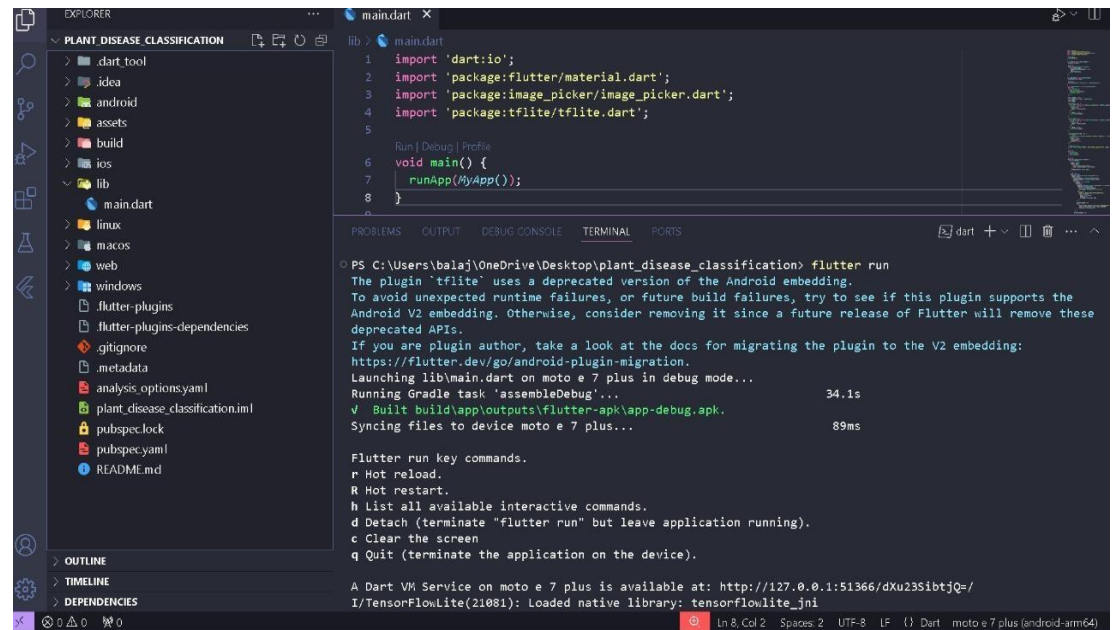
      backgroundColor: Colors.green.shade400,
      child: Icon(Icons.delete, color: Colors.white),
      tooltip: "Deselect Image",
    ),
    SizedBox(width: 20),
    FloatingActionButton(
      onPressed: captureImage,
      backgroundColor: Colors.green.shade400,
      child: Icon(Icons.camera_alt, color: Colors.white),
      tooltip: "Capture Image",
    ),
    SizedBox(width: 20),
    FloatingActionButton(
      onPressed: chooseImage,
      backgroundColor: Colors.green.shade400,
      child: Icon(Icons.photo_library, color: Colors.white),
      tooltip: "Pick Image",
    ),
    SizedBox(width: 20),
  ],
),
);
}
}

```

Screenshot 4.14: Main dart file code

### e) Run commands on Terminal:

After training model, and then implementing code we have to Run it on the terminal with specific commands.

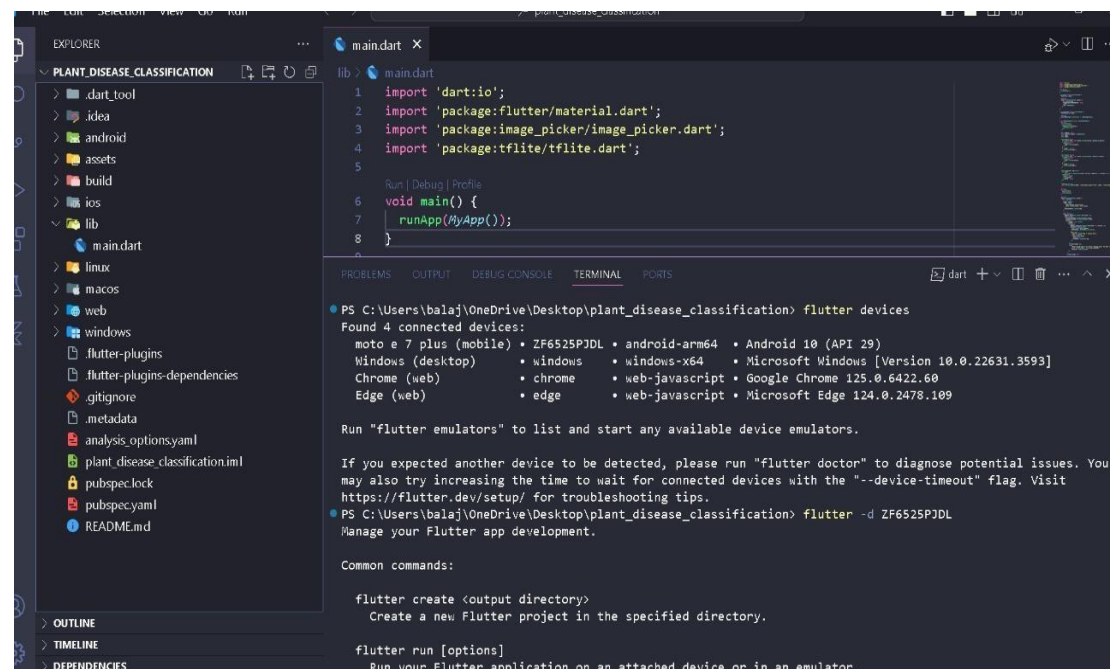


```
PS C:\Users\balaj\OneDrive\Desktop\plant_disease_classification> flutter run
The plugin 'tflite' uses a deprecated version of the Android embedding.
To avoid unexpected runtime failures, or future build failures, try to see if this plugin supports the
Android V2 embedding. Otherwise, consider removing it since a future release of Flutter will remove these
deprecated APIs.
If you are plugin author, take a look at the docs for migrating the plugin to the V2 embedding:
https://flutter.dev/go/android-plugin-migration.
Launching lib/main.dart on moto e 7 plus in debug mode...
Running Gradle task 'assembleDebug'... 34.1s
✓ Built build/app/outputs/flutter-apk/app-debug.apk.
Syncing files to device moto e 7 plus... 89ms

Flutter run key commands.
r Hot reload.
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).

A Dart VM Service on moto e 7 plus is available at: http://127.0.0.1:51366/dXu23SibtjQ=/
I/TensorFlowLite(21091): Loaded native library: tensorflowlite_jni
```

Screenshot 4.15: Run code through command.



```
PS C:\Users\balaj\OneDrive\Desktop\plant_disease_classification> flutter devices
Found 4 connected devices:
moto e 7 plus (mobile) • ZF6525PJDL • android-arm64 • Android 10 (API 29)
Windows (desktop) • windows • windows-x64 • Microsoft Windows [Version 10.0.22631.3593]
Chrome (web) • chrome • web-javascript • Google Chrome 125.0.6422.60
Edge (web) • edge • web-javascript • Microsoft Edge 124.0.2478.109

Run "flutter emulators" to list and start any available device emulators.

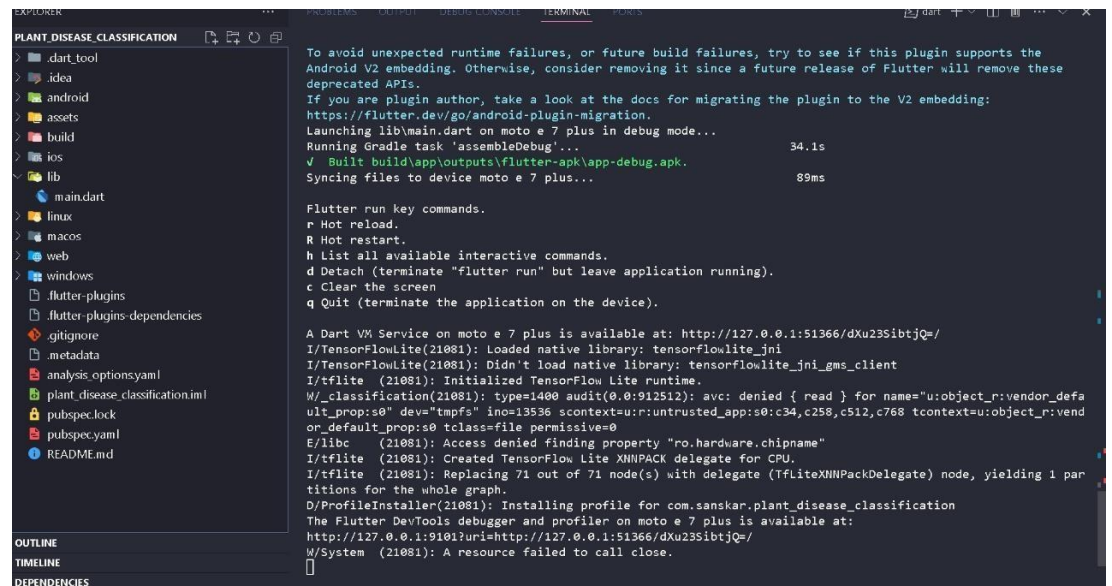
If you expected another device to be detected, please run "flutter doctor" to diagnose potential issues. You
may also try increasing the time to wait for connected devices with the "--device-timeout" flag. Visit
https://flutter.dev/setup/ for troubleshooting tips.
PS C:\Users\balaj\OneDrive\Desktop\plant_disease_classification> flutter -d ZF6525PJDL
Manage your Flutter app development.

Common commands:

flutter create <output directory>
Create a new Flutter project in the specified directory.

flutter run [options]
Run your Flutter application on an attached device or in an emulator.
```

Screenshot 4.16: Identifying the device.



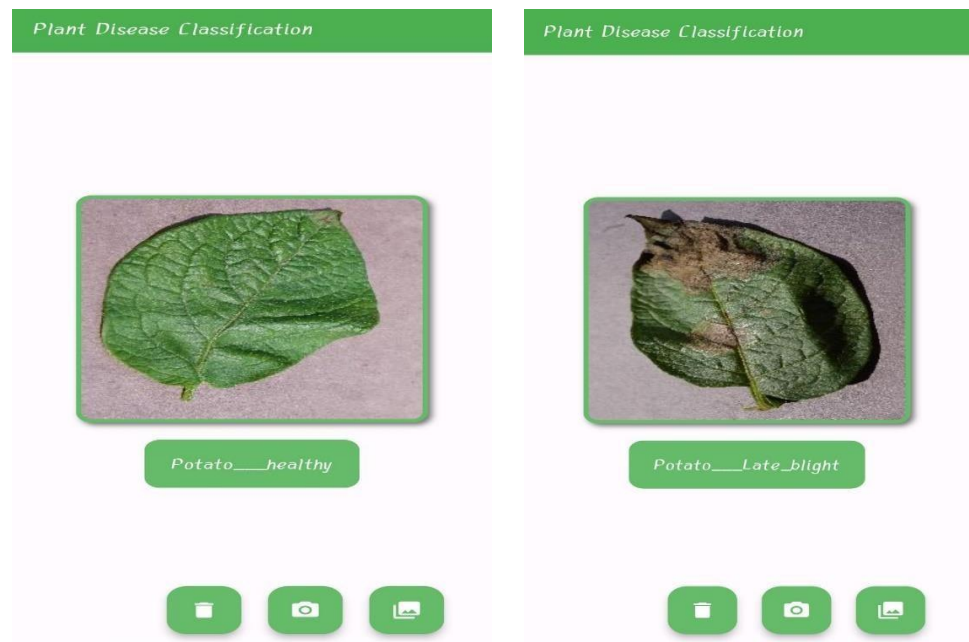
Screenshot 4.17: Executing the code.

#### f) Mobile Application:

Trained the model, integrated it with code then executed it in the terminal and we got an mobile application in our phone with an default symbol which is of Flutter.



Screenshot 4.18: Interface of Application



Screenshot 4.19: Plant disease detected.

This is the process which we have followed throughout and represented our all the work through screenshots.

## **5. Conclusion**

In conclusion, the development and implementation of plant disease detection technologies represent a crucial step forward in modern agriculture. By leveraging advancements in computer vision, machine learning, and sensor technologies, these systems offer a powerful means to monitor and manage crop health more effectively.

Through the systematic evaluation of performance we can produce and find out or predict the plant has, and save it before it gets completely wasted and get other crops harmed too. The integration of plant disease detection technologies into agricultural practices holds significant promise for enhancing crop productivity, minimizing yield losses, reducing reliance on chemical interventions, and ultimately contributing to global food security. Continued research, innovation, and collaboration are essential to further refine these technologies, expand their applicability, and address the evolving challenges facing modern agriculture.

By harnessing the potential of plant disease detection, we can empower farmers with timely and accurate information to make informed decisions, optimize resource allocation, and mitigate the impact of plant diseases on crop yields. Ultimately, the widespread adoption of these technologies has the potential to revolutionize agricultural practices, promoting sustainable and resilient food production systems for generations to come.

## References

1. Dhaygude, S.B., Kumbhar, N.P., (2013). Agricultural plant leaf disease detection using image processing, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, 2(1), 599-602.
2. Arivazhagan, S., Newlin Shebiah, R., Ananthi, S., Vishnu Varthini S., (2013). Detection of unhealthy region of plant leaves and classification of plant leaf diseases using texture features, Agricultural Engineering International: CIGR Journal, 15(1), 211–217.
3. Kulkarni Anand, H., Ashwin Patil, R.K., (2012). Applying image processing technique to detect plant diseases, International Journal of Modern Engineering Research, 2(5), 3661– 3664.
4. Yan Cheng Zhang, Han Ping Mao, Bo Hu, Ming Xili., (2007). features selection of Cotton disease leaves image based on fuzzy feature selection techniques, 2007 International Conference on Wavelet Analysis and Pattern Recognition, IEEE, Beijing.  
<https://doi.org/10.1109/ICWAPR.2007.4420649>
5. Dheeb Al Bashish, Malik Braik, Sulieman Bani-Ahmad, (2010). A Framework for Detection and Classification of Plant Leaf and Stem Diseases, International Conference on Signal and Image Processing, IEEE, India.  
<https://doi.org/10.1109/ICSIP.2010.5697452>



## **ACKNOWLEDGEMENT**

We would like to place on record our deep sense of gratitude to Prof. S. B. Kalyankar, HOD-Dept. of Computer Science and Engineering, Deogiri Institute of Engineering and management Studies Chatrapati Sambhajinagar, for his generous guidance, help and useful suggestions.

We express our sincere gratitude to Dr. Pramod Bhalerao, Dept. of Computer Science and Engineering, Deogiri Institute of Engineering and management Studies Chatrapati Sambhajinagar, for his stimulating guidance, continuous encouragement and supervision throughout the course of present work.

We are extremely thankful to Dr. Ulhas Shiurkar, Director and Dr. S V. Lahane, Dean Academics Deogiri Institute of Engineering, and management Studies Chatrapati Sambhajinagar, for providing me infrastructural facilities to work in, without which this work would not have been possible.

### **Signature(s) of Students**

Sonali Shakhawar  
Jidnayasa Pawar  
Mrunal Zambre