LEAF FUNGICIDE RECOMMENDATION

A Project report submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Submitted by

K. VENKATA SAI 318126511134 **P. JYOTHI SRAVANTHI 318126511145**

M. VYSHNAVI 318126511137 **K.V.S.S. VIJAY 318126511128**

Under the Guidance of

G. VIJAYA LAKSHMI

Associate Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES (UGC AUTONOMOUS)

(Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade)

Sangivalasa, Bheemili Mandal, Visakhapatnam dist. (A.P)

2021-2022

DEPARTMENT OF INFORMATION TECHNOLOGY

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES (UGC AUTONOMOUS)

(Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade)

Sangivalasa, Bheemili Mandal, Visakhapatnam dist. (A.P)

CERTIFICATE

This is to certify that the project report entitled "**LEAF FUNGICIDE RECOMENDATION**" submitted by **P. JYOTHI SRAVANTHI (318126511145), K. VENKATA SAI (318126511134), M. VYSHNAVI (318126511137), K.V.S.S.VIJAY (318126511128)** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in **Information Technology** of Anil Neerukonda Institute of technology and sciences, Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

Project Guide	**Head of the Department**
G. Vijaya Laskhmi	**Poosapati Padmaja**
Associate Professor	Professor
Department of IT	Department of IT
ANITS	ANITS

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **G. Vijaya lakshmi** Associate Professor, Department of Information Technology, ANITS, for her guidance with unsurpassed knowledge and immense encouragement. We are grateful to **Professor Poosapati Padmaja**, Head of the Department, Information Technology, for providing us with the required facilities for the completion of the project work.

We are very much thankful to the **Principal and Management, ANITS**, Sangivalasa, for their encouragement and cooperation to carry out this work.

We express our thanks to all **teaching faculty** of Department of IT, whose suggestions during reviews helped us in accomplishment of our project. We would like to thank all **non-teaching staff** of the Department of IT, ANITS for providing great assistance in accomplishment of our project.

We would like to thank our parents, friends, and classmates for their encouragement throughout our project period. At last, but not the least, we thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT STUDENTS:

P. Jyothi Sravanthi	**(318126511145)**
K.Venkata Sai	**(318126511134)**
M. Vyshnavi	**(318126511137)**
K.V.S.S Vijay	**(318126511128)**

DECLARATION

We hereby declare that the project work entitled "**LEAF FUNGICIDE RECOMENDATION**" submitted to the Anil Neerukonda Institute of Technology and Sciences is a record of an original work done by **P. Jyothi Sravanthi (318126511145) , K. Venkata Sai (318126511134) , M. Vyshnavi (318126511137),K.V.S.S Vijay (318126511128)** under the esteemed guidance of **G. Vijaya Lakshmi**, Associate Professor of Information Technology, Anil Neerukonda Institute of Technology and Sciences, and this project work is submitted in the partial fulfillment of the requirements for the award of degree Bachelor of Technology in Information Technology. This entire project is done with the best of our knowledge and have not been submitted for the award of any other degree in any other universities.

PROJECT STUDENTS:

P. Jyothi Sravanthi	**(318126511145)**
K.Venkata Sai	**(318126511134)**
M. Vyshnavi	**(318126511137)**
K.V.S.S Vijay	**(318126511128)**

INDEX

I.ABSTRACT

Currently 20-40 percent of the world's crop is lost due to plant diseases and lack of care. Plants are very susceptible to disease and often require care. Work has been going on in this area for a long time. However, there is no effective accuracy solution. Approximately one third of crop is lost every year to pest and diseases. The lack of awareness about the disease and cures is the main issue. In total millions of dollars every year are being used in wrong fungicides.

Early detection and diagnosing is the most significant practice for controlling. To achieve this, we discuss a solution that identifies the plant disease and provides a solution for cure using Transfer learning-based Convolution Neural Network (CNN). EfficientNetV2B0 is used which is smaller and faster while training. In this paper we classify different diseases between apple, potato, tomato. Early detection of leaf disease will reduce the amount of loss in crop. Some farmers are using wrong and excessive fungicides which results in polluting soil and food. This will help in using suitable solution in suitable amount. So, we can have healthy crop and reduce soil pollution.

Convolution Neural Network (CNN) is a type of artificial neural network used in image recognition and processing, specifically designed to process pixel data. Transfer learning is a research problem in the field of machine learning. It stores the knowledge gained while solving one problem and applies it to a different but related problem. For example, the knowledge gained while learning to recognize cats could apply when trying to recognize cheetahs. In deep learning, transfer learning is a technique whereby a neural network model is first trained on a problem similar to the problem that is being solved. Transfer learning has the advantage of decreasing the training time for a learning model and can result in lower generalization error.

II. LIST OF FIGURES

III. LIST OF ABBREVIATIONS

Serial No	Abbreviation Name	Page No
1	LFR- Leaf Fungicide Recommendation	
2	CNN-Convolutional Neural Network	
3	RGB-red-green-blue	
4	ANN-artificial neural network	**12**
5	GLCM-gray level Co-occurrence matrix	**13**
6	BPNN-back propagation neural network	**13**
7	API-application programming interface	**14**
8	GUI-graphical user interface	**21**

CHAPTER-1

INTRODUCTION

1. INTRODUCTION

India is a fast-growing country, and agriculture is the backbone of the country's early development, making it a growing economic giant and more than 65% of the population directly engaged in agriculture or agricultural products. Agriculture is struggling to meet its needs as the world population is growing rapidly. It is very important to bring technological advancement in the fields related to crop productivity in India.

Food and oxygen are key resources for livelihood. Plants contribute a major part in producing them. Every year 20-40 percent of crop production is affected by pests and diseases. Each year globally the plant diseases cost around $290 Billion. On the other hand, the human population is increasing at an average growth of 83 Million each year. If the population increases at this rate and the crop is being lost to diseases and pests there will be an imbalance in food chain and food crisis.

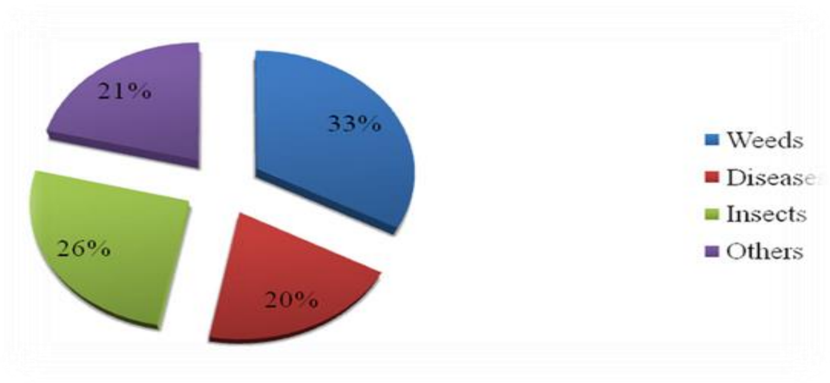


Figure 1: Graph representing causes of loss in crop

The diseases must be diagnosed in an early stage and cured with respective fungicides. It is difficult for farmers to identify the disease. The lack of knowledge of diseases and cures is a major problem. In some cases, fertilizers and fungicides are excessively used which is harmful for the soil and the humans eating them. So, for faster and accurate finding of disease and cures better techniques should be adopted.

Accordingly, the initial stage of plant diagnosis is an important task. Due to non-detection and treatment of the disease at an early stage, crops worth Rs.50,000 crore are lost annually due to pests and diseases. Crop disease reduces the quantity and quality of food produced. Crop diseases not only affect global food security, but also adversely affect small-scale farmers who depend on safe agriculture for their livelihood. Crops suffer a lot due to plant diseases and pest damage. Estimated figures show that tons of annual production worldwide was lost due to various pests in the early 21st century.

Plant diseases account for 15 to 17% of the total accumulated losses in the annual production range. 68% of the total average annual loss in crop production is lost in the form of losses due to various factors such as pests, weeds and leaf diseases of plants. Using this method we can distinguish between normal leaves and diseased leaves. Farmers need regular supervision by experts, which can be very costly and time consuming. Therefore, it is of great practical importance to look for quick, low-cost and accurate ways to intelligently identify diseases from leaf signs on plants.

1.1 PROBLEM STATEMENT:

Agriculture is an integral part of the Indian economy. The Indian agricultural sector employs almost half of the country's labor force. India is the largest producer of pulses, rice, wheat, spices and condiments in the world. The economic growth of farmers is determined by the quality of the produce they produce, which depends on the growth and yield of the plants. Consequently, in agriculture, it is important to identify the disease in plants.
Plants are susceptible to diseases that inhibit plant growth, which affects the ecology of the farmer. The use of automated disease detection techniques in early detection of plant disease is beneficial. Plant diseases are found in different parts of the plant like leaves. It is very difficult to manually detect plant disease using leaf images.
Therefore, there is a need to develop computational methods that automate the disease detection and classification process using leaf images. In our proposed solution we identify the percentage of the affected area and identify the disease and propose a solution. Our approach provides results in the shortest possible time with maximum accuracy compared to other existing approaches.

1.2 MOTIVATION:

It is very difficult for farmers to detect various diseases in plants. Worldwide, annual crop damage due to plant disease is estimated at $ 60 billion. Traditional tools and methods are not very useful because it takes a lot of time and manual work. However, these difficulties do not stop with the disease detection system. Therefore, it is necessary to evaluate agricultural products, increase market value and classify leaf diseases in accordance with quality standards. It can also be used to identify and take further action for the further spread of diseases.

1.3 RESEARCH OBJECTIVES:

- The objective of this research is detection and classification of tomato, potato and apple leaf diseases.
- To make sure that the farmers are suggested with the correct solution for their problem.
- To minimize the time taken for the analyzing and identifying the disease of the leaf
- To make sure that the disease is found in early stages
- This objective is achieved with five stages.
    - ➔ Stage 1: Data Exploration and Data Visualization
    - ➔ Stage 2: Image Segmentation
    - ➔ Stage 3: Making and Training the models
    - ➔ Stage 4: Ensembling the models
    - ➔ Stage 5: the disease detected and classified.

1.4 PROJECT SCOPE:

There have been advances in computer vision with the development of in-depth learning. It is widely used to identify plant diseases and is a satisfying alternative to the classification of plant diseases.The whole process of developing specimens for plant disease detection using deep CNNs is described in more detail. Beginning with the collection of images for the classification process using deep neural networks, the whole process is divided into several important stages in the following subsections.

CHAPTER-2

LITERATURE SURVEY

2. LITERATURE SURVEY

Till now many research were held for automation in classification of leaf diseases.

SUPPORT VECTOR MECHANISM

- N.Kanaka Durga has performed plant disease identification Using SVM and ANN algorithm. Same dataset is used to train with SVM and ANN .
- Support Vector Machine is a supervised machine learning algorithm used for classification problems.
- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
- The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.
- SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.
- Artificial neural networks or neural networks are computational algorithms inspired by the central nervous system of an animal.
- Results were 70 -75 %, 80-85% for SVM and ANN respectively. But KNN outperformed SVM and ANN.

K -NEAREST NEIGHBOURS

- K -Nearest Neighbors: Sandeep kumar has performed leaf disease detection using KNN. It is a supervised machine learning algorithm.
- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.
- K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.
- The number of immediate neighbors of a new unknown variable, which is classified, is

denoted by the 'K' symbol. This has performed pretty well for less classes.

- But based on CNN outperforms KNN, SVM in classification for higher number of classes.

CONVOLUTION NEURAL NETWORK

- Convolution Neural Network: Melike Sardogan has used CNN for plant leaf disease detection..
- Convolutional Neural Network is one of the main categories to do image classification and image recognition in neural networks. Scene labeling, objects detections, and face recognition, etc., are some of the areas where convolutional neural networks are widely used.
- In CNN, each input image will pass through a sequence of convolution layers along with pooling, fully connected layers, filters (Also known as kernels). After that, we will apply the Soft-max function to classify an object with probabilistic values 0 and 1.
- CNN works good for plant disease classification But, Transfer learning works better.
- Over the past few years many good models were developed like Inception, ResNet, MobileNet, EfficientNet. Among these EfficientNetB0 works better for coloured images dataset
- Previously people worked with ResNet , Inception, MobileNet but according to a study in 2021 EfficientNetB0 has highest accuracy among all the pretrained models.

CHAPTER 3

SYSTEM ANALYSIS

3.SYSTEM ANALYSIS

3.1 SYSTEM REQUIREMENTS:

3.1.1 Hardware Requirements:

- RAM: 4GB and Higher
- Processor: Intel i3 and above
- Hard Disk:500GB Minimum

3.1.2 Software Requirements:

- OS: Windows family
- Python IDE: Python 3.8
- Online IDE: Google Collaboratory
- Browser version: Google Browser 8 and above

3.2 System Environment:

1.Python
2.Tensorflow
3.Keras
4.Matplotlib
5.Numpy
6.Open CV
7.Tkinter

1.Python

Python is a computer programming language often used to build websites and software, automate tasks, and conduct data analysis. Python is a general purpose language, meaning it can be used to create a variety of different programs and isn't specialized for any specific problems. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on.

2.Tensorflow

It is an open source library of artificial intelligence, which uses data flow graphs to build models. Allows developers to create large neural networks with multiple layers. TensorFlow is widely used: Planning, Visibility, Understanding, Discovering, Predicting and Creating. Features of tensorflow are:

- Auto Differentiation: the process of automatically calculating the gradient vector of a model in relation to each of its parameters. With this feature, TensorFlow can automatically calculate gradients in parameters in the model, which is useful for algorithms such as backpropagation that require gradients to improve performance.
- Distribute: Both arbitrarily and graphically, TensorFlow provides spreadsheet API for multiple devices with various distribution techniques.
- Loss: To train and test models, TensorFlow offers a set of loss functions (also known as cost functions).
- Metrics: To test the performance of machine learning models, TensorFlow provides API access to the most widely used metrics. Examples include various metrics for accuracy (binary, phased, partial).
- TF.nn: TensorFlow.nn is a module for using older neural network functions in models.To install

```
$ pip install tensorflow
```

3.Keras

It is a high-end neural network library operating over TensorFlow, CNTK, and Theano. Using Keras for in-depth learning allows for easy and fast prototyping and seamless operation on the CPU and GPU. This framework is coded with Python code that is easy to edit and allows for easy expansion.

Apart from this, Keras has features such as :

- It runs smoothly on both CPU and GPU.
- It supports almost all neural network models.
- It is modular in nature, which makes it expressive, flexible, and apt for innovative research.

Keras is a deep learning API written in Python, running on top of the machine learning platform TensorFlow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result as fast as possible is key to doing good research.Keras is

- Simple -- but not simplistic. Keras reduces developer cognitive load to free you to focus on the parts of the problem that really matter.
- Flexible -- Keras adopts the principle of progressive disclosure of complexity: simple workflows should be quick and easy, while arbitrarily advanced workflows should be possible via a clear path that builds upon what you've already learned.
- Powerful -- Keras provides industry-strength performance and scalability: it is used by organizations and companies including NASA, YouTube, or Waymo.

To install keras

```
$ pip install tensorflow
```

4.Matplotlib

Matplotlib. Pyplot is a set of functions that make matplotlib work as MATLAB. Each pyplot function makes a certain change in the image: eg, creates a figure, creates a layout in the image, edits certain lines in the editing area, decorates the structure with labels, etc.
Matplotlib is an excellent visualization library in Python for 2D arrays. Matplotlib is a multi-platform data visualization library built on the NumPy array and designed to work with a wide range of SciPy stacks. It was introduced in 2002 by John Hunter. One of the biggest advantages of visualization is that it allows visual access to large amounts of data in easily digestible sequences. Matplotlib contains many platforms such as lines, bars, scatters, histograms, etc
Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats .

.

To install

$pip install matplotlib

5.Numpy

NumPy can be used to perform various mathematical tasks in arrows. Add powerful data structures to Python that ensure effective statistics with arrays and matrices and provide a large library of high-level mathematical operations that work on these same components and matrices. NumPy (Numerical Python) is an open source Python library used in almost every field of science and engineering. It is the universal standard for working with numeric data in Python and is a key component of Scientific Python and Pydata ecosystems. NumPy's customers range from beginner coders to experienced researchers conducting advanced scientific and industrial research and development.
Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.
It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.
To install

python -m pip install -U matplotlib

6.Open CV

OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source Apache 2 License.
It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.
In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and videos.The purpose of computer vision is to understand the content of the images. It extracts the description from the pictures, which may be an object, a text description, and three-dimension model.
Features of OpenCV Library

- Using OpenCV library, you can −
- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection

To install OpenCv

pip install opencv-python

7.Tkinter

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.
Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps :

- Import the Tkinter module.
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

pip install tk

CHAPTER-4

SYSTEM DEVELOPMENT

4. SYSTEM DEVELOPMENT

The main goal of the system is to make Machine learning model to classify the 17 diseases of Apple, Potato Tomato. Later integrate the model with a front end. Transfer learnin based CNN model is imported and customized to the problem.

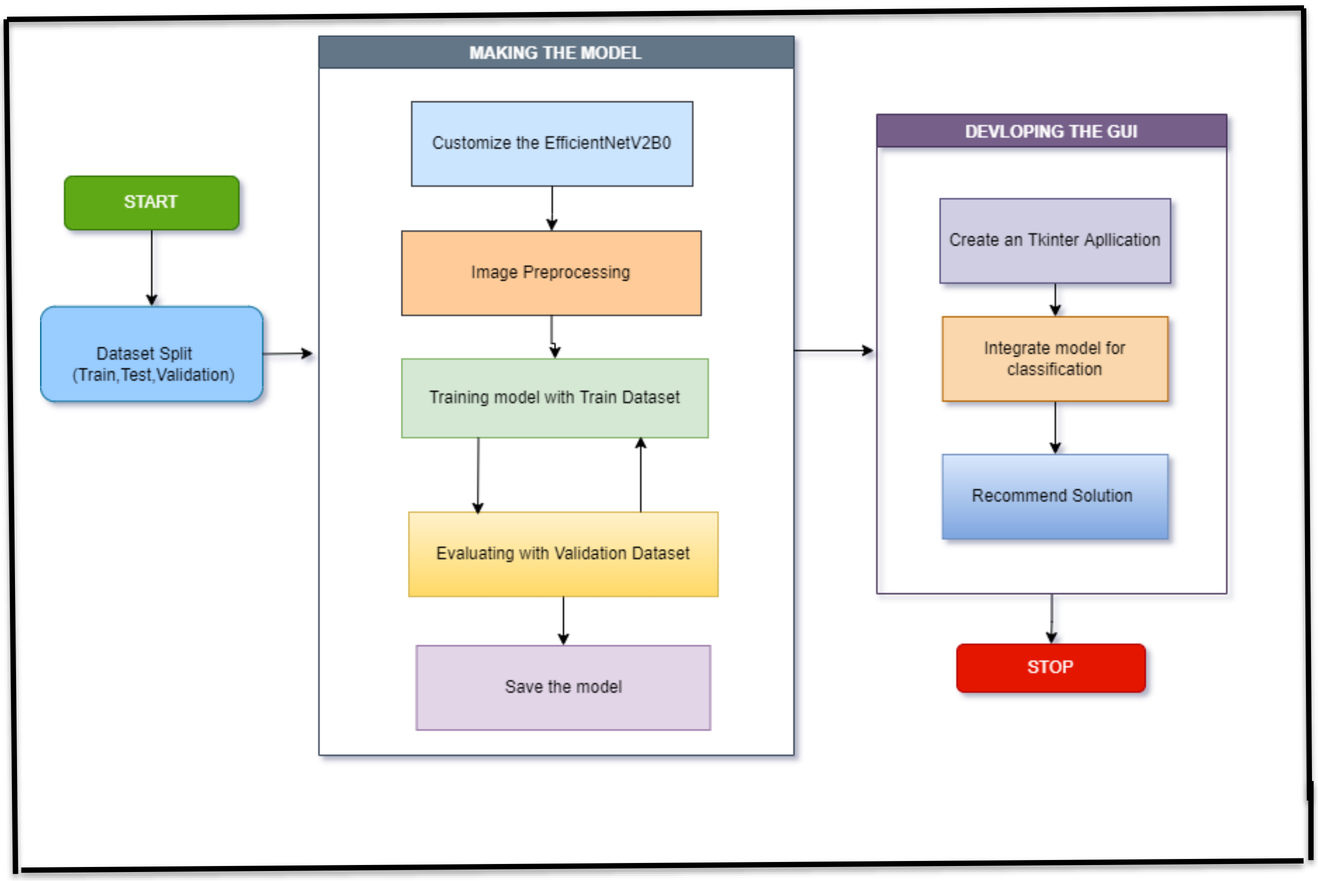

START
Dataset Split (Train,Test,Validation)
MAKING THE MODEL
Customize the EfficientNetV2B0
Image Preprocessing
Training model with Train Dataset
Evaluating with Validation Dataset
Save the model
DEVLOPING THE GUI
Create an Tkinter Apllication
Integrate model for classification
Recommend Solution
STOP

Figure 4.1 : Flow chart of System Development

4.1 Making the model

Our main goal is to make a better model than the models present out there. Model should have faster training speeds and high classification accuracy. Based on a research paper by [5] Sk mahumudl the EfficientNetB0 has higher accuracy. So, we have considered EfficieNetB0 for training the model. Recently Google has launched EfficieNetV2 models. These models have surpassed all the models in training speed and accuracy. Small models tend to have less training time. Transfer learning based EfficieNetV2B0 is imported from Tensorflow hub.
We have considered an dataset of 17 plant diseases. These are the diseases found in Apple, Tomato, Potato. The images are unaugmented and has pixel size of 256x256 with 3 channels(RGB). The dataset is split into 3 parts Train, validation, test with 80%,10%,10% respectively. The Train dataset is used to train the model. The Train dataset is augmented before

training the images are rotated, zoomed randomly so the model doesn't become overfitting.
The Model is imported and the input layer and output layer are customized. The model is pretrained with ImageNet and CIFAR dataset so it classifies daily ordinary objects like flowers, cars, airplanes, dogs, cats etc. Models all training features are made false. It can no more detect the common objects. Now we make the model input layer with our leaf images. The output class is defines as 17 classes. The images are randomly rotated and trained. The model is trained for 8 epochs the model is evaluated with validation dataset after each epoch and prints the accuracy. We are using the validation dataset instead of test data while training because we are preventing the model from training the test images. We have observed accretion in accuracy and diminishing loss over the epochs .

4.2 Developing the GUI

Tkinter is used to make an windows based application. It provides widgets like buttons, Labels, Message. We have used the buttons to perform actions like open camera, brows files, predict disease. The Message is used to show the instructions of how to use the application. The application offers 2 types of importing image.

1. Browse files from system,
2. Capture image using camera

By browsing the files in the system we can import the images and classify. This helps us to easy classify the images if present in system. The model is integrated with OpenCV to capture an images and classify the image. When we use the capture from camera function the system saves an image for every second. The function terminates when we hit the space bar. The last saved image is then classified. After importing another image or closing the application the captured image is terminated.
The Application is fast and easy to use. When we click on the Predict button it basically uses the previously trained model for classification. While loading the tkinter application it simultaneously loads the trained model also. This model is integrated with the GUI application. This process of integration with GUI makes the process easier to classify an imager with out understanding the internal working of the model. Any person with basic knowledge can work with the GUI application and classify the model. It classifies the model is very less time. It also trains 5x faster than other models. We can customize the model to classify more number of classes also.

CHAPTER-5

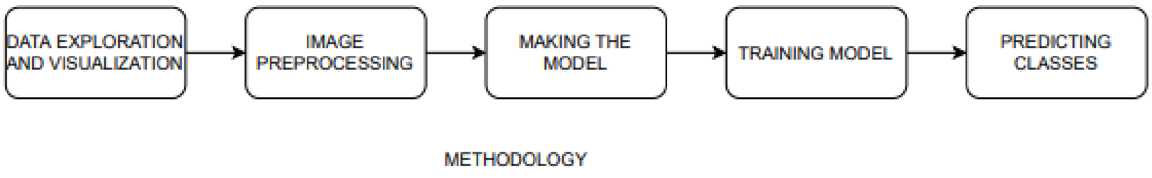
METHODOLOGY

5.METHODOLOGY

5.1 GENERAL:

In this chapter we describe the general process of Implementation details of algorithmic steps. LFR uses CNN to classify the images and Recommend a solution for the Disease.It includes Following steps

1. Data Exploration and Visualization



2. Image Preprocessing
3. Making the models
4. Training Model
5. Predicting classes

5.2 DATA EXPLORAATION AND VISUALIZATION:

Importing the data and visualization is done to check if the data is correctly imported or not. Visualization and exploring sets the understanding about the data is an important skill .Data exploration is the initial step in data analysis, where users explore a large data set in an unstructured way to uncover initial patterns, characteristics, and points of interest. This process isn't meant to reveal every bit of information a dataset holds, but rather to help create a broad picture of important trends and major points to study in greater detail. Data exploration can use a combination of manual methods and automated tools such as data visualizations, charts, and initial reports.

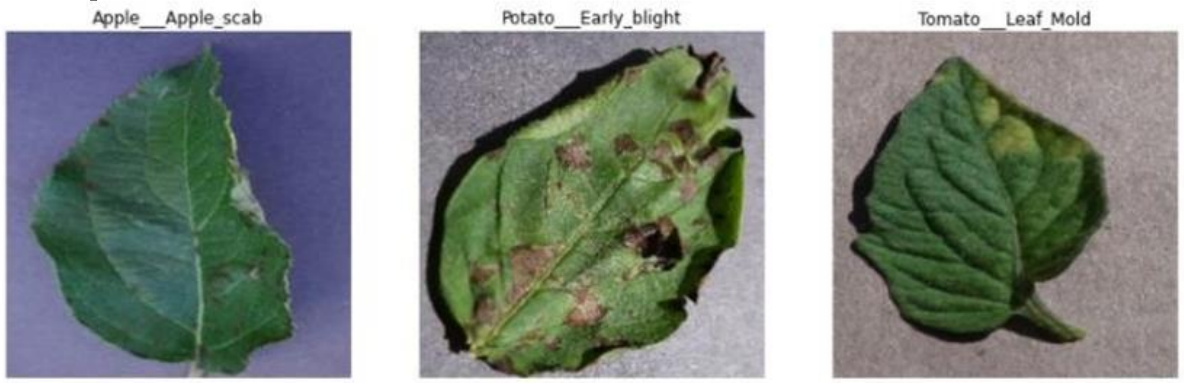


Figure 5.1: Sample images from Dataset

The dataset covers 17 classes with dimensions of 256 x 256 pixels color images (RGB). There are a total of 23483 images. The dataset is free from noise and has the same background for all images.

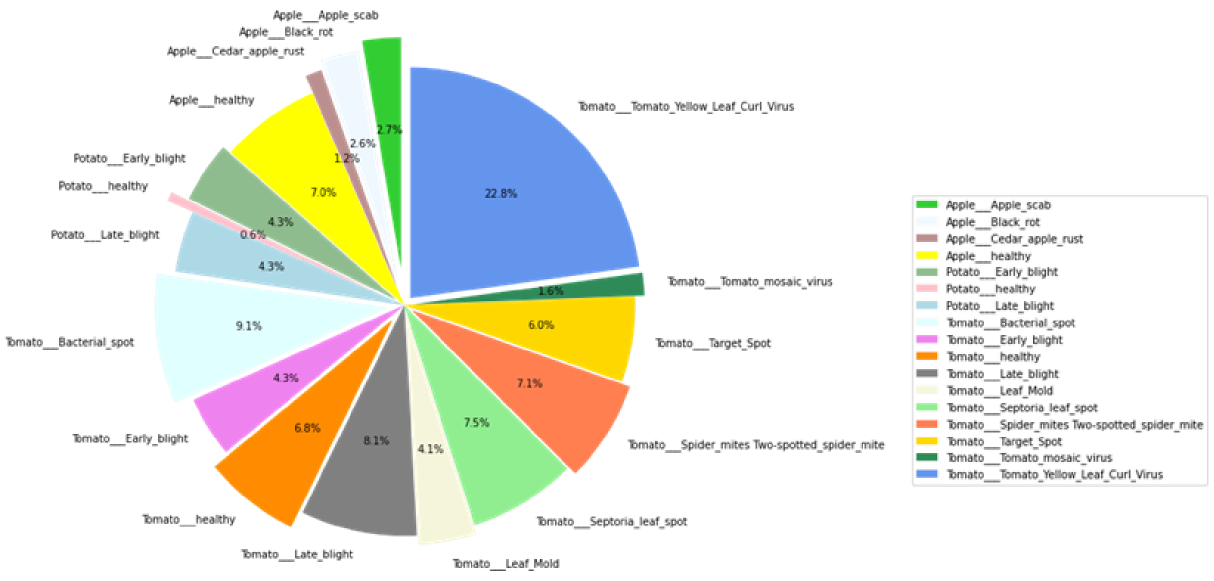


Figure 5.2: Pie chart representing the size of each class in dataset

5.3 IMAGE PREPROCESSING:

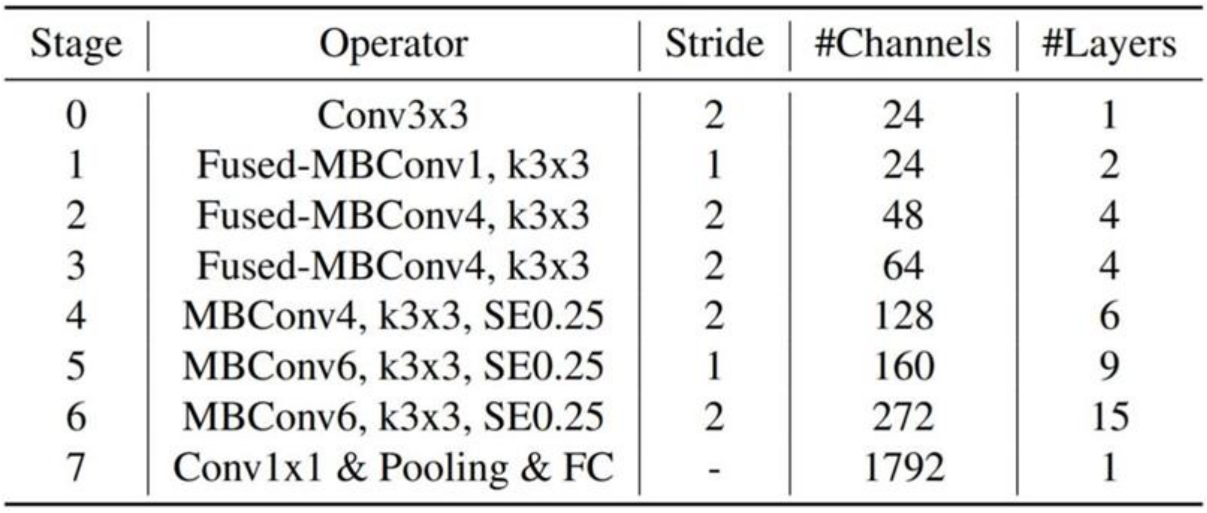
Pre-image processing is the steps that are taken to format images before they are used in modeling training and description. This includes, but is not limited to, changing the size, shape, and color correction.

Adding an image is a trick used in images to create different versions of the same content to present a model in a wide range of training examples. For example, a random change in rotation, brightness, or scale of an input image requires the model to pay attention to how the image title looks in different situations.

The illusion of image enhancement is a form of pre-image processing, but there is an important difference: while image processing steps are used in training and testing sets, image enhancement is used only in training data. Therefore, a change that may be an increase in some cases may be a step forward in others.

5.4 TRAINING MODEL:

Training the model is the most important part of the algorithm. After a few layers of integration and large, the final separation is done with fully integrated layers. Neurons in a fully integrated layer have a connection to everything activated in the previous layer, as evidenced by normal (fixed) activity networks of artificial senses.
Deep learning is a class of machine learning. Convolution Neural Network is one of the Deep learning Algorithms. In this paper transfer learning based CNN model is used. Transfer learning is a process of taking the relevant parts of a pre-trained learning model and using it to solve a new similar problem. A pre-trained CNN model is taken and input, output layers are customized. Sk Mahmudul Hassan [5] used Inception V3, InceptionResNetV2, MobileNetV2, EfficientNetB0 on leaf diseases data set . It resulted that EfficientNetB0 has better performance than others in coloured leaf images in all splits of Dataset.



Stage	Operator	Stride	#Channels	#Layers
0	Conv3x3	2	24	1
1	Fused-MBConv1, k3x3	1	24	2
2	Fused-MBConv4, k3x3	2	48	4
3	Fused-MBConv4, k3x3	2	64	4
4	MBConv4, k3x3, SE0.25	2	128	6
5	MBConv6, k3x3, SE0.25	1	160	9
6	MBConv6, k3x3, SE0.25	2	272	15
7	Conv1x1 & Pooling & FC	-	1792	1

Figure 5:.3 EfficientNetV2 Architecture

EfficientNetV2 is a new family of convolutional networks with faster training speeds and better parameter capabilities than previous models. This was developed with a combination of Training aware Neural Architecture Discovery and scaling to optimize training speed and parameter capabilities. With progressive learning, EfficientNetV2 has surpassed previous models in ImageNet and CIFAR / Car / Flower dataset. By pre-training on the same ImageNet21k, our EfficientNetV2 achieved 87.3% Top-1 accuracy in ImageNet ILSVRC2012, surpassing the recent ViT with 2.0% accuracy while training 5x-11x faster using the same computing resources .

EfficientNet are known for high quality and quick image classification. They arevery popular for their scaling method which makes their training much faster than other networks. Recently google released the EfficientNetV2, which is a huge improvement over EfficientNet in terms of training speed and better in terms of accuracy.

EfficientNetV2 makes extensive use of both MBConv and the newly added Fuse-MBConv in the opening layers. EfficientNetV2 prefers smaller expansion ratios for MBConv because smaller expansion ratios can lead to lower memory access overhead.

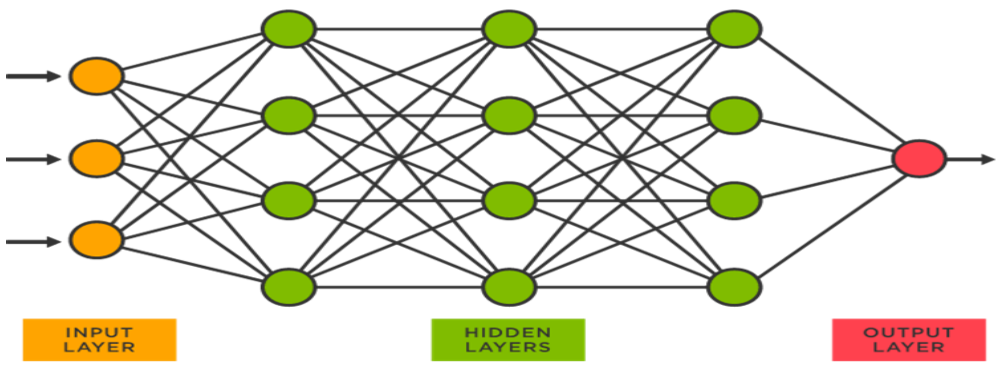
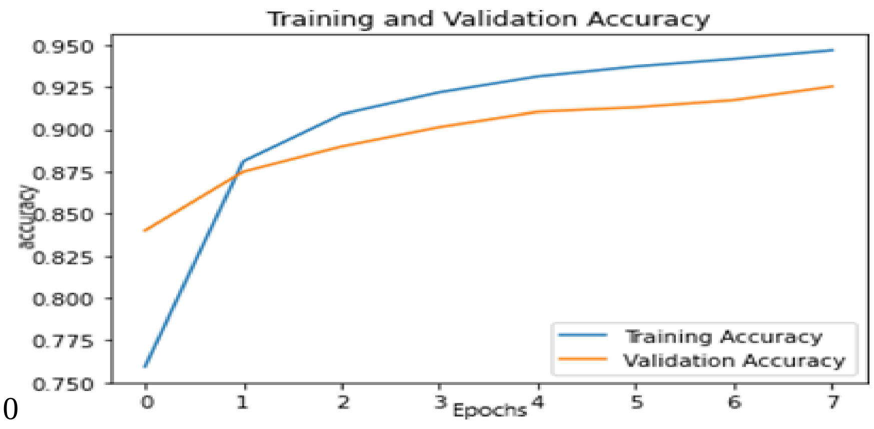


Figure 5.4: CNN Structure

Results of post training the model have significantly increased accuracy over the epochs and decrease in loss. Below are the graphs depicting the significant progress over the epochs. Accuracy is the number of correct predicted images over the total number of images. Training accuracy is the accuracy on training dataset after each epoch. Validation accuracy is the accuracy on validation dataset after each epoch.



0

Figure 5.5 : Training ,Validation accuracy vs Epcohs

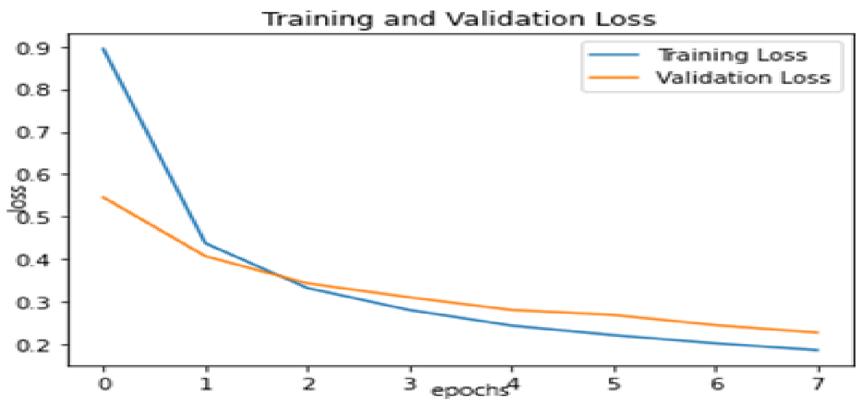


Figure 5.6: Training ,Validation Loss vs Epochs

CHAPTER -6

CODING

6.CODING

6.1 LFR_TRAINING.IPYNB

```
In [1]: import tensorflow as tf
        import matplotlib.pyplot as plt
        import tensorflow_hub as hub
        from tensorflow.keras import layers
```

```
In [2]: BATCH_SIZE = 30
        IMAGE_HEIGHT = 224
        IMAGE_WIDTH=224
        CHANNELS=3
        EPOCHS=8
```

```
In [3]: # importing training Dataset

        TrainDataset = tf.keras.preprocessing.image_dataset_from_directory(
            "E:\\leaf disease\\split dataset\\train",
            shuffle=True,
            image_size=(IMAGE_HEIGHT,IMAGE_WIDTH),
            batch_size=BATCH_SIZE
        )

        Found 18780 files belonging to 17 classes.
```

```
In [4]: ValidationDataset= tf.keras.preprocessing.image_dataset_from_directory(
            "E:\\leaf disease\\split dataset\\val",
            image_size=(IMAGE_HEIGHT,IMAGE_WIDTH),
            batch_size=BATCH_SIZE
        )

        Found 2343 files belonging to 17 classes.
```

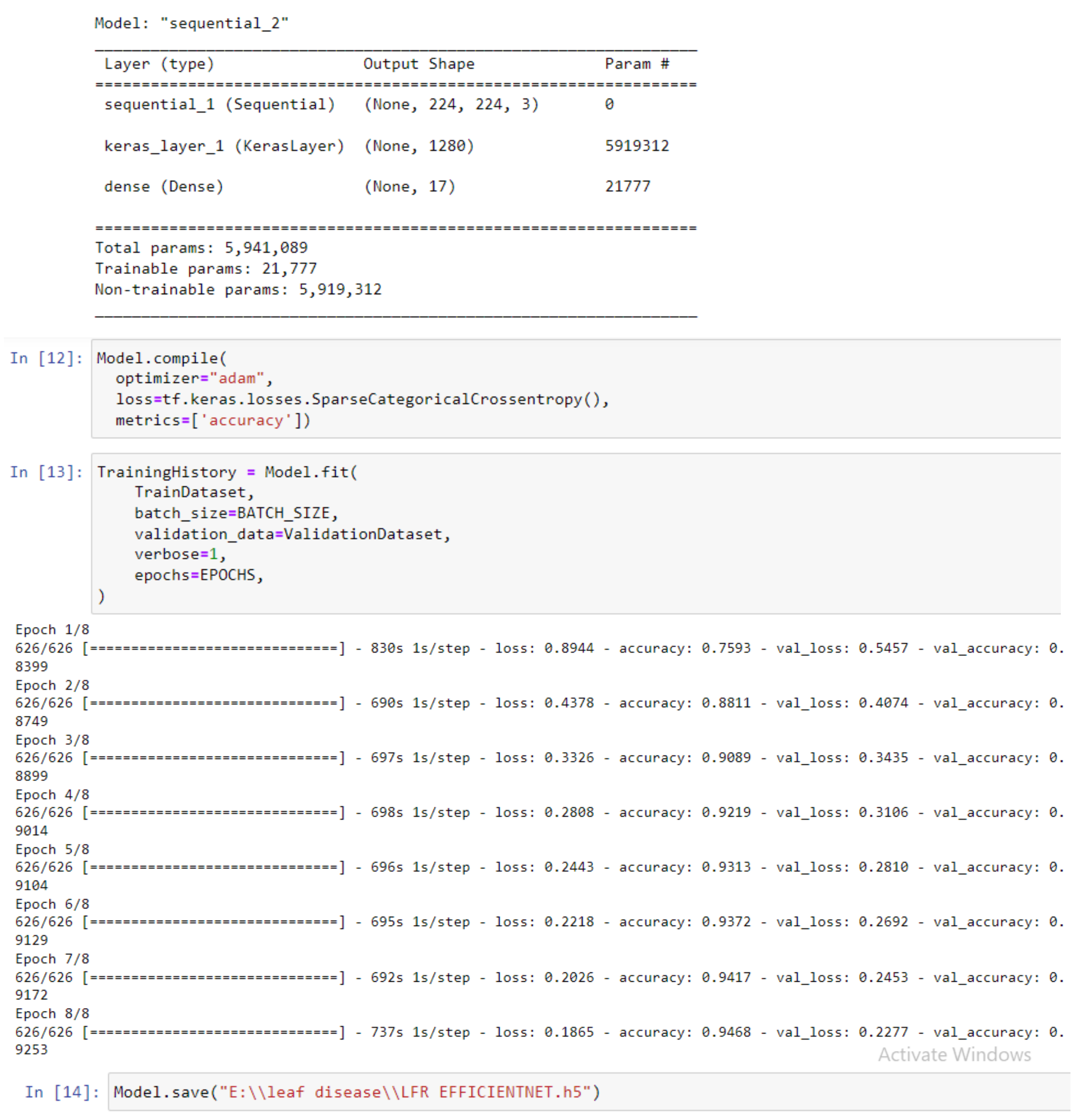
```
In [5]: classes= TrainDataset.class_names
```

```
In [8]: preprocess = tf.keras.Sequential([
            layers.experimental.preprocessing.Resizing(IMAGE_HEIGHT,IMAGE_WIDTH),
            layers.experimental.preprocessing.RandomRotation(0.3),
            layers.experimental.preprocessing.RandomZoom(0.1)
        ])
```

```
In [9]: InputShape = (IMAGE_HEIGHT,IMAGE_WIDTH, CHANNELS)

        efficientnet_v2_model = "https://tfhub.dev/google/imagenet/efficientnet_v2_imagenet1k_b0/feature_vector"

        efficientnet_v2_model_without_top_layer = hub.KerasLayer(
                                                    efficientnet_v2_model,
                                                    input_shape=InputShape,
                                                    trainable=False
                                                )
```

```
In [10]: OutputClasses=len(classes)
```

```
In [11]: Model = tf.keras.Sequential([
             preprocess,
             efficientnet_v2_model_without_top_layer,
             tf.keras.layers.Dense(OutputClasses, activation='softmax')
         ])

         Model.build([None, IMAGE_HEIGHT,IMAGE_WIDTH,CHANNELS])
         Model.summary()
```



```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_1 (Sequential)   (None, 224, 224, 3)       0

 keras_layer_1 (KerasLayer)  (None, 1280)              5919312

 dense (Dense)               (None, 17)                21777

=================================================================
Total params: 5,941,089
Trainable params: 21,777
Non-trainable params: 5,919,312
_________________________________________________________________
```

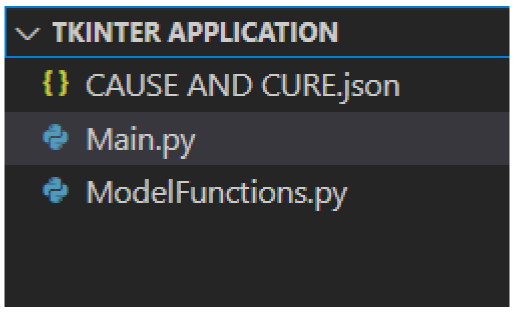
```
In [12]: Model.compile(
           optimizer="adam",
           loss=tf.keras.losses.SparseCategoricalCrossentropy(),
           metrics=['accuracy'])
```

```
In [13]: TrainingHistory = Model.fit(
             TrainDataset,
             batch_size=BATCH_SIZE,
             validation_data=ValidationDataset,
             verbose=1,
             epochs=EPOCHS,
         )
```

```
Epoch 1/8
626/626 [==============================] - 830s 1s/step - loss: 0.8944 - accuracy: 0.7593 - val_loss: 0.5457 - val_accuracy: 0.
8399
Epoch 2/8
626/626 [==============================] - 690s 1s/step - loss: 0.4378 - accuracy: 0.8811 - val_loss: 0.4074 - val_accuracy: 0.
8749
Epoch 3/8
626/626 [==============================] - 697s 1s/step - loss: 0.3326 - accuracy: 0.9089 - val_loss: 0.3435 - val_accuracy: 0.
8899
Epoch 4/8
626/626 [==============================] - 698s 1s/step - loss: 0.2808 - accuracy: 0.9219 - val_loss: 0.3106 - val_accuracy: 0.
9014
Epoch 5/8
626/626 [==============================] - 696s 1s/step - loss: 0.2443 - accuracy: 0.9313 - val_loss: 0.2810 - val_accuracy: 0.
9104
Epoch 6/8
626/626 [==============================] - 695s 1s/step - loss: 0.2218 - accuracy: 0.9372 - val_loss: 0.2692 - val_accuracy: 0.
9129
Epoch 7/8
626/626 [==============================] - 692s 1s/step - loss: 0.2026 - accuracy: 0.9417 - val_loss: 0.2453 - val_accuracy: 0.
9172
Epoch 8/8
626/626 [==============================] - 737s 1s/step - loss: 0.1865 - accuracy: 0.9468 - val_loss: 0.2277 - val_accuracy: 0.
9253
```

```
In [14]: Model.save("E:\\leaf disease\\LFR EFFICIENTNET.h5")
```

6.2 TKINTER APPLICATION



```
TKINTER APPLICATION
  {} CAUSE AND CURE.json
  Main.py
  ModelFunctions.py
```

6.2.1 MAIN.PY

from tkinter import *

```
from PIL import ImageTk, Image
from tkinter import filedialog
import os
import json
from datetime import date,datetime
import cv2
import ModelFunctions as MF

if os.path.exists("input.jpg"):
  os.remove("input.jpg")
today = date.today()
root = Tk()
def open_img():
	x = openfilename()
	img = Image.open(x)
	img = img.resize((250, 250), Image.ANTIALIAS)
	img.save("input.jpg")
	for widgets in can2.winfo_children():
		widgets.destroy()
	img = ImageTk.PhotoImage(img)
	for widgets in can3.winfo_children():
		widgets.destroy()
	panel = Label(can2, image = img)
	panel.image = img
	panel.grid(row=1,column=0,padx=5,pady=5)
	l = Label(left_frame_center_left, text="Input Image",bg="white")
	l.grid(row=1,column=0)
	l = Label(left_frame_center_right, text="Input Image Details",bg="white")
	l.grid(row=1,column=0)
	image = Image.open("input.jpg")
	info_dict = {
    "Image Size": image.size, "Image Format": image.format,
    "Image Mode": image.mode, "Frames in Image": getattr(image, "n_frames", 1),
	"Date":str(today), "Time":str(datetime.now().time())}
	count=1
	for label,value in info_dict.items():
		t=f"{label:25}: {value}"
		p = Message(can3,text=t,width=300,bg="white", fg="#000",justify=LEFT)
		p.grid(row=count,column=1)
		count+=1
def openfilename():
	filename = filedialog.askopenfilename(title ='pen')
	return filename
def predict_btn():
	if os.path.exists("input.jpg"):
		for widgets in right_can.winfo_children():
			widgets.destroy()
		predicted_class,confidence=MF.predict()
		print(predicted_class,confidence)
		Name=Label(right_can,text=" "+str(predicted_class),font= ('Helvetica 13 '))
		Name.grid(row=3,column=0)
		Name=Label(right_can,text=" Predicted Disease:"+str(predicted_class),font= ('Helvetica 13 '))
		Name.grid(row=3,column=0)
		Confidence=Label(right_can,text=" Confidence:"+str(confidence),font= ('Helvetica 13'))
		Confidence.grid(row=4,column=0)
```

```
                path="CAUSE AND CURE.json"
                for widgets in right_frame_bottom.winfo_children():
                        widgets.destroy()
                f=open(path)
                data=json.load(f)
                f.close()
                frame1=LabelFrame(right_frame_bottom  ,text=predicted_class,font= ('Helvetica 16'),)
                frame1.grid(row=0,column=0,padx=15,pady=15)
                About=Label(frame1,text="About",font= ('Helvetica 14 underline'))
                About.grid(row=1,column=0)
                l = Message(frame1,
                text= data[predicted_class]["About"],font= ('Helvetica 13'), width=610)
                l.grid(row=2,column=0,padx=10)
                Cause=Label(frame1,text="Cause",font= ('Helvetica 14 underline'))
                Cause.grid(row=3,column=0)
                l2 = Message(frame1, text= data[predicted_class]["Cause"],font= ('Helvetica 13'), width=610)
                l2.grid(row=4,column=0,padx=10)
                Cure=Label(frame1,text="Cure",font= ('Helvetica 14 underline'))
                Cure.grid(row=5,column=0)
                l3 = Message(frame1,text= data[predicted_class]["Cure"],font= ('Helvetica 13'),   width=610)
                l3.grid(row=6,column=0,padx=10)
        else:
                l4 = Label(right_can,
                text= "*UPLOAD A IMAGE*",
                font= ('Helvetica 12 underline'),
                fg="red")
                l4.grid(row=0,column=0)
def open_cam():
        cam = cv2.VideoCapture(0)
        cv2.namedWindow("PRESS SPACE TO SAVE IMAGE")
        while True:
                ret, frame = cam.read()
                if not ret:
                        print("failed to grab frame")
                        break
                cv2.imshow("PRESS SPACE TO SAVE IMAGE", frame)
                k = cv2.waitKey(1)
                if k%256 == 27:
                        print("Escape hit, closing...")
                        cv2.destroyAllWindows()
                        break
                elif k%256 == 32:
                        img_name = "input.JPG"
                        cv2.imwrite(img_name, frame)
                        print("Saved")
                        cv2.destroyAllWindows()
                        break
        img=Image.open("input.jpg")
        img = img.resize((250, 250), Image.ANTIALIAS)
        img.save("input.jpg")
        for widgets in can2.winfo_children():
                widgets.destroy()
        img = ImageTk.PhotoImage(img)
        for widgets in can3.winfo_children():
                widgets.destroy()
        panel = Label(can2, image = img)
        panel.image = img
        panel.grid(row=1,column=0,padx=5,pady=5)

        l = Label(left_frame_center_left, text="Input Image",bg="white")
        l.grid(row=1,column=0)
        l = Label(left_frame_center_right, text="Input Image Details",bg="white")
        l.grid(row=1,column=0)
        image = Image.open("input.jpg")
        info_dict = {    "Image Size": image.size,"Image Format": image.format,
    "Image Mode": image.mode,"Frames in Image": getattr(image, "n_frames", 1),
        "Date":str(today),"Time":str(datetime.now().time())}
        count=1
        for label,value in info_dict.items():
                t=f"{label:25}: {value}"
                p = Message(can3,text=t,width=300,bg="white", fg="#000",justify=LEFT)
                p.grid(row=count,column=1)
                count+=1
root.title("Leaf Funcigide Recommendation")
w= root.winfo_screenwidth()
h= root.winfo_screenheight()
root.geometry("%dx%d" % (w, h))
root.minsize(w, h)
root.grid_columnconfigure(0, weight = 1)
root.grid_columnconfigure(1, weight = 1)
root.grid_rowconfigure(0, weight = 1)
left_frame = Frame(root)
left_frame.grid(row = 0, column = 0, sticky = "nesw")
right_frame = Frame(root)
right_frame.grid(row = 0, column = 1, sticky = "nesw")
left_frame.grid_rowconfigure(0, weight = 1)
left_frame.grid_rowconfigure(1, weight = 2)
left_frame.grid_columnconfigure(0, weight = 1)
left_frame_top = Frame(left_frame)
left_frame_top.grid(row = 0, column = 0, sticky = "nesw")
left_frame_top.grid_propagate(False)
left_frame_center = Frame(left_frame)
left_frame_center.grid(row = 1, column = 0, sticky = "nesw")
left_frame_center.grid_propagate(False)
left_frame_center.grid_columnconfigure(0, weight = 1)
left_frame_center.grid_columnconfigure(1, weight = 1)
left_frame_center.grid_rowconfigure(0, weight = 1)
left_frame_center_left = Frame(left_frame_center, bg = "white", highlightthickness=2)
left_frame_center_left.grid(row = 0, column = 0, sticky = "nesw")
left_frame_center_left.grid_propagate(False)
left_frame_center_right = Frame(left_frame_center, bg="white", highlightthickness=2)
left_frame_center_right.grid(row = 0, column = 1, sticky = "nesw")
left_frame_center_right.grid_propagate(False)
can = Canvas(left_frame_top)
can.place(relx=0.5, rely=0.5, anchor=CENTER)
frame1=LabelFrame(can,text="About",font= ('Helvetica 14'),)
frame1.grid(row=0,column=0)
About= Message(frame1,
text= "\n This is a app can load images from your device and can predict its disease in Minimal Possible
time,Either Click the below 'Browse Files' button  and select the required Image  or use 'Capture' to take
images from camera then click on 'Predict' button to find About ,Cause,Cure of Disease \n",
width=600,font= ('Helvetica 13'),justify=CENTER)
About.grid(row=0,column=0,padx=10)
btn1 = Button(can, text ='Browse Files', command =
open_img,height=1,width=10).grid(row=1,column=0,pady=10)
btn2=Button(can, text ='Capture ',command=open_cam,height=1,width=10).grid(row=3,column=0,pady=10)
can2 = Canvas(left_frame_center_left,bg = 'white',highlightthickness=0)
can2.place(relx=0.5, rely=0.5, anchor=CENTER  )
l1=Label(can2,text="No Input Image")
l1.grid(row=0,column=0)
can3 = Canvas(left_frame_center_right,bg = 'white',highlightthickness=0)
can3.place(relx=0.5, rely=0.5, anchor=CENTER)
l2=Label(can3,text="No Input Image")
l2.grid(row=0,column=0)
right_frame.grid_rowconfigure(0, weight = 1)
right_frame.grid_rowconfigure(1, weight = 1)
right_frame.grid_rowconfigure(2, weight = 5)
right_frame.grid_columnconfigure(0, weight = 1)
right_frame.grid_propagate(False)
right_frame_top = Frame(right_frame)
right_frame_top.grid(row = 0, column = 0, sticky = "nesw")
right_frame_top.grid_propagate(False)
right_frame_center = Frame(right_frame)
right_frame_center.grid(row = 1, column = 0, sticky = "nesw")
right_frame_center.grid_propagate(False)
right_frame_bottom = Frame(right_frame)
right_frame_bottom.grid(row = 2, column = 0, sticky = "nesw")
right_frame_bottom.grid_propagate(False)
frame1=LabelFrame(right_frame_bottom,text="Disease Name" ,font= ('Helvetica 14'))
frame1.grid(row=0,column=0,padx=10,pady=10)
About=Label(frame1,text="About",font= ('Helvetica 13 underline'))
About.grid(row=0,column=0)
l = Message(frame1,text= " "*200,width=610,)
l.grid(row=1,column=0,padx=10)
Cause=Label(frame1,text="Cause",font= ('Helvetica 13 underline'))
Cause.grid(row=2,column=0)
l2 = Message(frame1,text="\n"*2,width=610)
l2.grid(row=3,column=0,padx=10)
Cure=Label(frame1,text="Cure",font= ('Helvetica 13 underline'))
Cure.grid(row=4,column=0)
l3 = Message(frame1,text= "\n"*3,width=610)
l3.grid(row=5,column=0,padx=10)
canv = Canvas(right_frame_top)
canv.place(relx=0.5, rely=0.5, anchor=CENTER)
right_can= Canvas(right_frame_center)
right_can.place(relx=0.5, rely=0.5, anchor=CENTER)
btn3=Button(canv, text ='predict',command = predict_btn,height=3,width=10).grid(row=1,column=0)
root.mainloop()
if os.path.exists("input.jpg"):
  os.remove("input.jpg")
```

6.2.2 ModelFunctions.py

```
import tensorflow as tf
import tensorflow_hub as hub
import numpy as np
from tensorflow.keras.preprocessing import image
import os

os.environ["TFHUB_CACHE_DIR"] = "Folder path”
model = tf.keras.models.load_model("Path to model",custom_objects={'KerasLayer':hub.KerasLayer})

def predict():
        class_names=[ “Names of all classes”]
        img_path ="input.jpg"
        img = image.load_img(img_path, target_size=(224, 224))
        x = image.img_to_array(img)
        x = np.expand_dims(x, axis=0)
        predictions = model.predict(x)
        predicted_class = class_names[np.argmax(predictions[0])]
        confidence = round(100 * (np.max(predictions[0])), 2)
        return predicted_class,confidence
```

6.2.3 CAUSE and CURE.json

Sample of single diesease

```
 "Apple___Apple_scab":
                          {
                           "About" : "Apple Scab is a fungal disease which can affect the leaves,
                          fruit, and twigs of flowering, ornamental trees such as crabapple,
                          hawthorn (Venturia inequalis) and pear (V. pirina).",

                          "Cause" : "Apple Scab is caused by the ascomycete fungus Venturia
                          inaequalis.",

                          "Cure"  : "For best control, apple scab spray that is liquid copper soap
                          early can be used two weeks before symptoms appear normally.
                          Alternatively, begin the applications when the disease first appears, and
                          repeat it at least seven to ten days' interval till the blossom drops."

                          },
```

CHAPTER 7

TESTING

7.TESTING

TESTING MODEL:

Testing is done once the code is developed. This stage is usually a subset of all the stages as in the project development. The testing activities are mostly involved in all the stages. However,this stage refers to the testing only stage of the disease where disease defects are reported ,tracked,fixed and retested ,until the result reaches the quality standards.

To verify the performance of proposed method, we have conducted a set of experiments on healthy and diseased tomato leaf image databases and have performed classification .One of the main challenges in disease detection and classification for this study is that leaves with different diseases are very similar to each other .Therefore,this similarity can cause some leaves to be folded into to wrong classes.
To verify the performance of the model . We need to check its classification on the test dataset. Input an image and let the model predict then compare with the actual name of the image. We can calculate accuracy by formula

Accuracy (%) = ((No. of correctly Predicted) / (Total no of leaves in Datasets))*100

The accuracy of some classes is low because of less data for the classes.

Below are some sample results from the testing.

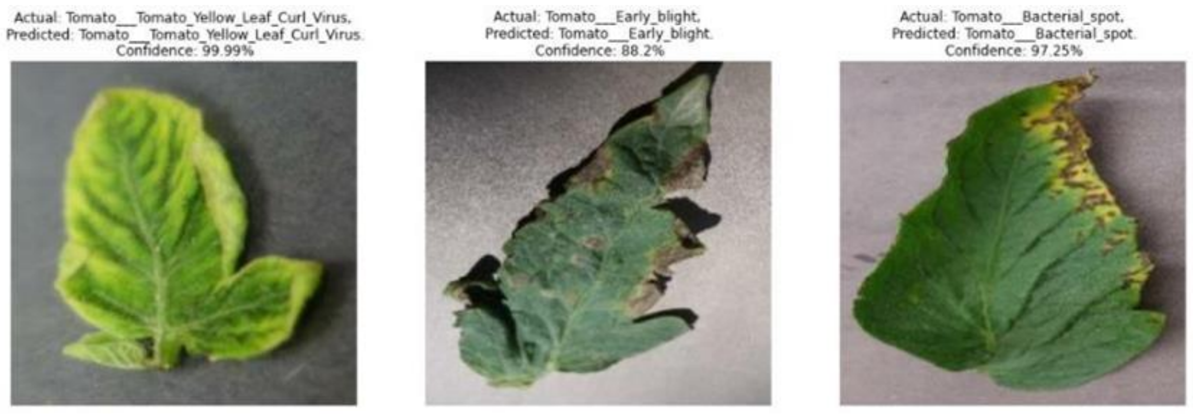

Actual: Tomato___Tomato_Yellow_Leaf_Curl_Virus,
Predicted: Tomato___Tomato_Yellow_Leaf_Curl_Virus.
Confidence: 99.99%
Actual: Tomato___Early_blight,
Predicted: Tomato___Early_blight.
Confidence: 88.2%
Actual: Tomato___Bacterial_spot,
Predicted: Tomato___Bacterial_spot.
Confidence: 97.25%

Figure 7.1 : Sample images of testing

We have trained the model for 8 epochs and evaluated the model on train and test datasets and observed the graphs of performance metrics accuracy and loss.The prediction for test leaf images are compared with disease dataset.

The custom CNN gives the highest training accuracy of all the models that we trained but got the lowest validation accuracy of all. The difference in the training and validation accuracy of the custom CNN may suggest that the model may be slightly overfitting to the data. This model is trained for only 8 epochs with all the callbacks applied of other implementation techniques.

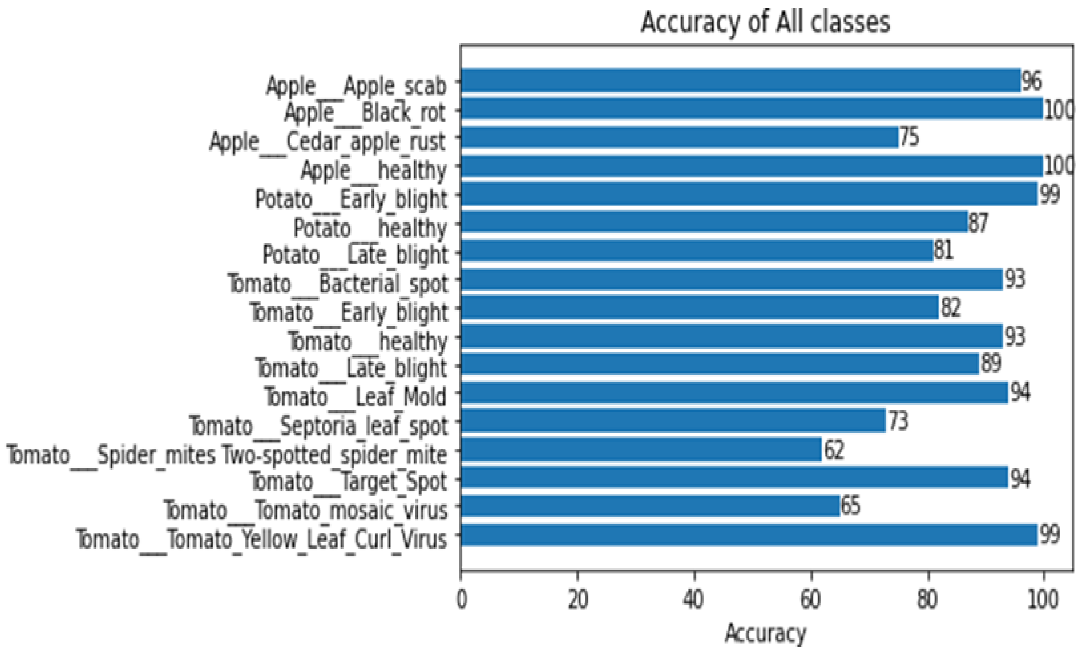


Figure 7.2: Testing results

Since the samples are less in Tomato spider mites two spotted sider mite and mosaic virus are less the respective accuracy is also less when compared to others.

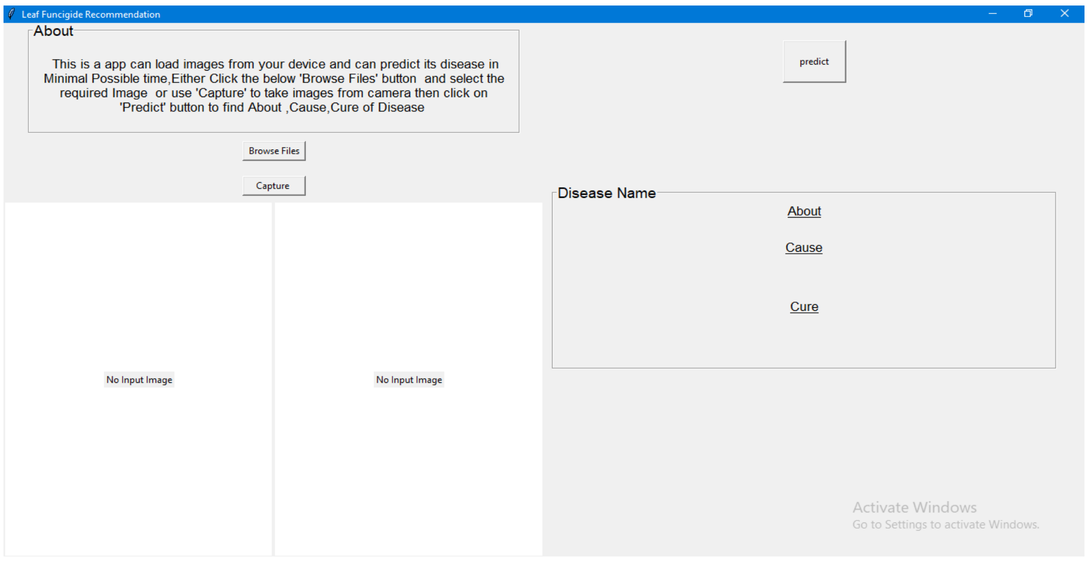
Accuracy and precision are two measures of observational errors. Accuracy is how close or far off a given set of measurements (observations or readings) are to their true value, while precision is how close or dispersed the measurements are to each other.

CHAPTER 8

RESULTS

8.RESULTS

This is a windows application integrated with the model. This application is capable of browsing files from the system and capturing images from webcam. Then predicts disease in images and recommends a suitable solution for the disease. This is a simple GUI. Since the people are unaware of the disease this application suggests it with a cure and educates them about the cause.



This is a app can load images from your device and can predict its disease in Minimal Possible time,Either Click the below 'Browse Files' button and select the required Image or use 'Capture' to take images from camera then click on 'Predict' button to find About ,Cause,Cure of Disease

Figure 8.1: Main page of LFR

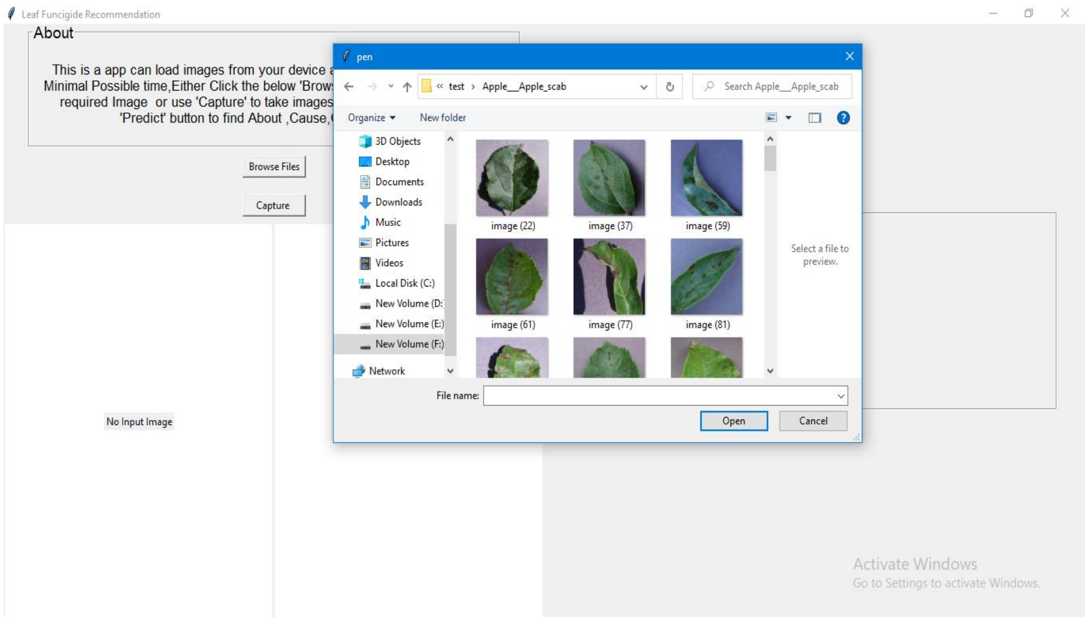
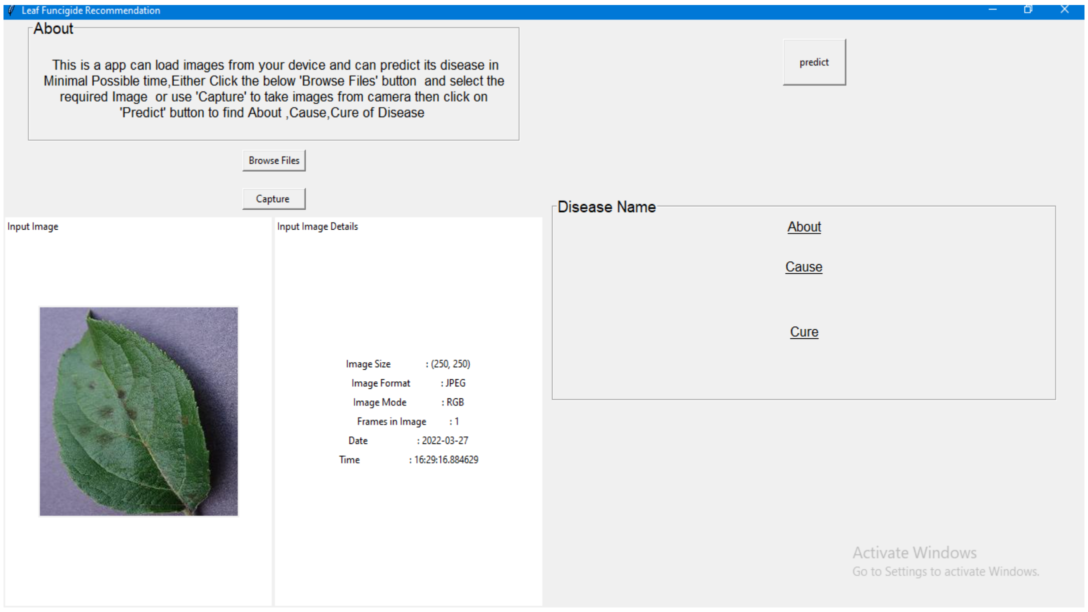


Figure 8.2: After clicking on browse files


Leaf Funcigide Recommendation
predict
Browse Files
Capture
Activate Windows
Go to Settings to activate Windows.

About

This is a app can load images from your device and can predict its disease in Minimal Possible time,Either Click the below 'Browse Files' button and select the required Image or use 'Capture' to take images from camera then click on 'Predict' button to find About ,Cause,Cure of Disease

Input Image

Input Image Details

```
Image Size        : (250, 250)
Image Format      : JPEG
Image Mode        : RGB
Frames in Image   : 1
Date              : 2022-03-27
Time              : 16:29:16.884629
```

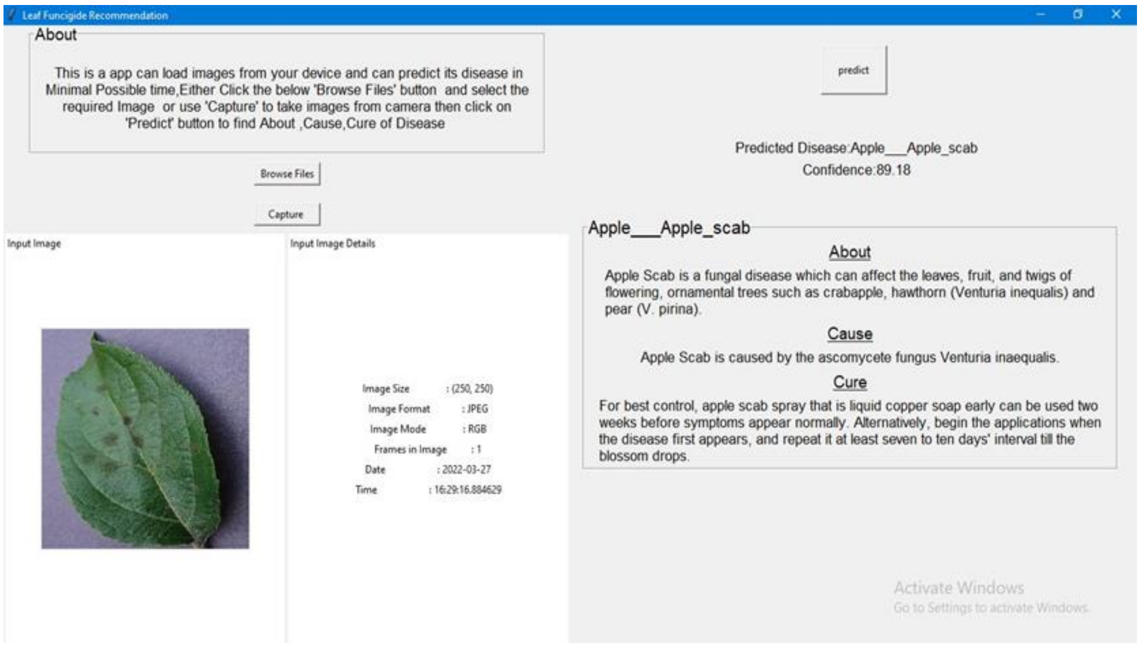
Disease Name

About

Cause

Cure

Figure 8.3 : selecting image or capture image from camera then click on ‘Predict’ button


Leaf Funcigide Recommendation
predict
Browse Files
Capture
Activate Windows
Go to Settings to activate Windows.

About

This is a app can load images from your device and can predict its disease in Minimal Possible time,Either Click the below 'Browse Files' button and select the required Image or use 'Capture' to take images from camera then click on 'Predict' button to find About ,Cause,Cure of Disease

Predicted Disease:Apple___Apple_scab
Confidence:89.18

Input Image

Input Image Details

```
Image Size        : (250, 250)
Image Format      : JPEG
Image Mode        : RGB
Frames in Image   : 1
Date              : 2022-03-27
Time              : 16:29:16.884629
```

Apple___Apple_scab

About

Apple Scab is a fungal disease which can affect the leaves, fruit, and twigs of flowering, ornamental trees such as crabapple, hawthorn (Venturia inequalis) and pear (V. pirina).

Cause

Apple Scab is caused by the ascomycete fungus Venturia inaequalis.

Cure

For best control, apple scab spray that is liquid copper soap early can be used two weeks before symptoms appear normally. Alternatively, begin the applications when the disease first appears, and repeat it at least seven to ten days' interval till the blossom drops.

Figure8.4 :showing about the disease of image,cause and cure

CHAPTER-9

CONCLUSION

9. CONCLUSION

Protecting crops in organic farming is not an easy task .In our system,a special deep learning model is developed based on a special architectural convolution network to detect plant diseases through images of healthy or diseased plant leaves. Studies show that managing plant diseases can help increase yields by about 50%.This will help the Farmers to Find and cure them in less time,more accuracy,less labour.

In this application we provide a GUI which helps to easily identify the plant disease of Apple, Potato, Tomato diseases. In this paper the model used is EfficientNetV2B0 which is a smaller and faster training model. This application helps to find the disease, cure the disease and educate about the cause of the disease. The CNN algorithm is implemented successfully to train the system. This present work can contribute to agricultural domain and can be used to help people for tracking their house plants and also enables the farmers to keep a track of the harvest. This work can be expanded into further to develop an app through which one would also know the remedy to a leaf disease.

Augmentation and transfer learning in this case, proved beneficial to the model, helping the CNN to generalize more reliability. While this improved the model's ability to extract features are, it was not enough when the model was presented with'in field' imagery. In this case, the classifier ranked an accuracy Above all, this highlights the importance of diversifying the training dataset to include alternative background data, additional plant anatomy and varying stages of disease.

Overall, this study is conclusive in demonstrating how transfer learning CNNs may be applied to empower small-holder farmers in their fight against leaf disease. In the future, work should be focused on diversifying training datasets and also in testing similar applications in real life situations. Without such developments, the struggle against leaf disease will continue. This application predicts faster, more accurately, which will reduce the usage of wrong pesticides over these fields. Usage of correct dosage and correct Fungicide is recommended. This will reduce the pollution from the crop and result in healthy food. This application can also be extended and integrated with drones which will reduce the manpower. It can also reduce time along with covering humongous areas of crops and frequent diagnosis helps to stop diseases at early stages

CHAPTER-10

REFERENCES

10.REFERENCES

[1]. Manohar lal, Budhi Ram and Prabhat Tiwari "Botanicals to Cops Stored Grain Insect Pests" 2017

[2]. N. Kanaka Durga, G. Anuradha, Plant Disease Identification Using
SVM and ANN Algorithms published in IJRET in February 2019
[3]. Sandeep kumar, KMVV Prasad, A. Srilekha, T.Suman, B. Pranav Rao, J.Naga
Vamshi Krishna in " Leaf Disease Detection and Classification based on Machine
Learning"
[4]. Bijaya Kumar hatuwal, Aman Shakya and Basanta joshi ,"Plant Leaf Disease
Recognition using Random Forest ,KNN,SVM,CNN
[5].Sk Mahmudul Hassan,Arnab Kumar Maji, Michał Jasiński , Zbigniew Leonowicz
and Elzbieta Jasinska Identification of Plant-Leaf Diseases Using CNN and Transfer-
Learning Approach
[6]. Dataset of Plant Leaf Disease data.mendeley.com/datasets/tywbtsjrjv/1[7]. Mingxing Tan,Quoc v.Le ," EfficientNetV2: Smaller Models and Faster Training",june 2021[8].Leaf Fungicide Recomendation-github.com/venkatasai7/Leaf-Fungicide-Recomendation
[9]. Melike Sardogan, Adem Tuncer, Yunus Ozen Plant Leaf Disease Detection and
Classification Based on CNN with LVQ Algorithm
[10].Aman Arora "EfficientNetV2",https://wandb.ai/wandb_fc/pytorchss-image-
models/reports/EfficientNetV2--Vmlldzo2NTkwNTQ