Leaf Disease Identification using Deep Learning

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the requirement for the award of the Degree of

BACHELOR OF TECHNOLOGY
in
Computer Science Engineering

by

Roopa Chowdary Cherukuri (19BCE7049)
Allu Meghana Chowdary (19BCD7105)

Under the Guidance of

Dr. Udit Narayana Kar

SCHOOL OF COMPUTER SCIENCE ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

DECEMBER 2022

CERTIFICATE

This is to certify that the Capstone Project work titled "**Leaf Disease Prediction using Deep Learning**" that is being submitted by **Roopa Chowdary Cherukuri (19BCE7049) and Allu Meghana Chowdary (19BCD7105)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. Udit Narayana Kar
Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner **External Examiner**

Approved by

PROGRAM CHAIR	**PROGRAM CHAIR**	**DEAN**
B. Tech. CSE	B.Tech. CSE-DA	School of Computer Science Engineering

ACKNOWLEDGEMENTS

I would like to express my gratitude to **Dr. G. Viswanathan,** Chancellor, **Dr. Sekar Viswanathan** Vice president, **Dr. S. V. Kota Reddy,** Vice Chancellor and Dean **Dr. Sudha S V**, School of Computer Science and Engineering, for providing a conducive environment to work during the tenure of the course.

In a jubilant mood, I express ingeniously my whole-hearted thanks to Dr Saroj Kumar Panigraphy(Program chair of CSE-Core) and Dr Mehfooza Munvar Basha(Program chair of CSE – DA) the Professors of the School of Computer Science and Engineering and it's my pleasure to express the deep sense of gratitude to Dr Udit Narayana Kar, Assistant Professor, School of Computer Science and Engineering, VIT-AP, for his constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavour.

It is indeed my pleasure to thank all faculty members working as limbs of our university for their selfless enthusiasm coupled with timely encouragement showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly towards the successful completion of this project.

Place: Amaravati
Date: 29-12-2022

Roopa Chowdary Cherukuri
Allu Meghana Chowdary

ABSTRACT

Plant diseases pose a significant risk to farmers, consumers, the environment, and the global economy. In India alone, viruses and pests destroy 35% of field crops, forcing farmers to lose money. Pesticide indiscriminate usage is also a severe health hazard, as many are poisonous and biomagnified. These negative consequences are avoidable by early disease identification, crop surveillance, and tailored therapies. Agricultural professionals examine exterior signs to detect the majority of ailments. Farmers, on the other hand, have limited access to professionals. Ours is the first platform for automated illness detection and tracking. Farmers may use a web tool to quickly and precisely diagnose illnesses and find treatments by photographing afflicted plant portions. The newest Deep Learning CNN algorithm enables real-time diagnostics. The CNN model in our study was trained using ten different types of Tomato leaf sick photos. The automatic CNN model was used to diagnose test pictures. Our solution is a unique, scalable, and accessible method for disease control of a wide range of agricultural crop plants that can be deployed as a web service for farmers and professionals to ensure environmentally sustainable crop production.

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

Agriculture is necessary for human life. Increased agricultural, fruit, and vegetable yield is especially important in densely populated developing nations like India. Not only must output be maintained, but so must generate quality in order to improve public health. However, problems like as disease transmission that may have been avoided with early detection impede both production and food quality. Many of these illnesses are contagious, resulting in total crop output loss.

Human aided disease diagnosis is ineffective and cannot keep up with the extravagant demand due to the huge geographical spread of agricultural areas, poor education levels of farmers, limited knowledge, and lack of access to plant pathologists. To address the shortage of human-assisted disease diagnosis, it is critical to use technology to automate crop disease diagnosis and make low-cost, accurate machine-assisted disease diagnosis available to farmers. Some progress has been made in utilising technology such as robotics and computer vision systems to tackle a wide range of agricultural concerns.

Image processing has been investigated as a tool to aid in precision agricultural techniques, weed and pesticide technology, plant growth monitoring, and plant nutrition management. Despite the fact that many plant diseases may be diagnosed by plant pathologists by visual inspection of physical signs such as noticeable colour change, wilting, and the emergence of spots and lesions, progress on automating plant disease detection is still primitive.

Overall, commercial investment in connecting agriculture and technology remains lower than investment in more profitable domains such as human health and education. Due to difficulties such as access and connectivity for farmers to plant pathologists, high cost of implementation, and scalability of solution, promising research initiatives have not been able to productize. Recent advancements in Web Technology and Deep Learning provide an ideal chance to produce a scalable, low-cost crop disease treatment that can be widely adopted. Mobile phones with internet access have become commonplace in developing nations such as India. Low-cost mobile phones with cameras and GPS are commonly accessible and may be used by users to post photographs with geolocation. They can interface with more complex Cloud-based backend services that can do compute-intensive activities, manage a centralized database, and perform data analytics across widely available mobile networks. Another technological advancement in recent years is AI-based image analysis, which has outperformed human vision and can reliably recognize and categorize pictures.

The underlying AI algorithms employ Neural Networks (NN), which consist of layers of neurons connected in a way inspired by the visual brain. These networks are "trained" on a large collection of pre-classified "labelled" photos in order to obtain high image classification accuracy on fresh unseen images. The breakthrough in CNN capabilities has resulted from a combination of enhanced computational capabilities, massive visual data sets, and improved NN algorithms.

Aside from accuracy, open source platforms such as TensorFlow have enabled AI to improve and become more inexpensive and accessible. Prior work for our research includes endeavors to collect healthy and sick crop photos, image analysis with feature extraction, RGB images, spectral patterns, and fluorescence imaging spectroscopy. In the past, neural networks were used to diagnose plant diseases, but the strategy was to recognize textural traits. Our proposal leverages the advancements in Deep Learning and Web Technology to provide an end-to-end crop diagnosis system that duplicates the knowledge ("intelligence") of plant pathologists and makes it available to farmers.

1.1 OBJECTIVES

The following are the objectives of this project:

- ➢ To develop a platform for farmers that is
- ▪ low cost
- ▪ reliable
- ▪ accessible way of identifying leaf disease.
- ➢ To train the networks on a set of pre-classified "labeled" images to achieve high accuracy of image classification on new unseen images.
- ➢ To make the real-time diagnosis possible using CNN algorithm using web application.

1.2 BACKGROUND AND LITERATURE SURVEY

We have gone through a research paper that presents a process in which it combines a self-organizing map as well as a Bayesian classifier for categorizing sick plants growing in dynamic scenarios like as greenhouses, where the lack of control over lighting and the presence of background pose significant challenges. During the training phase, two of them are used: one that forms color groups of photos, which are then classified into two groups using K -means and labelled as vegetation and non-vegetation using norms, and a second one that corrects the first one's classification faults. During the testing phase, the Bayesian classifier segments an input picture and converts it to a binary image, from which outlines are extracted and evaluated to recover sick regions that were mistakenly categorized as non-vegetation. The experimental outcomes utilizing suggested method outperformed some of the most widely commonly used color score algorithms.

We also scrutinized a report that stated Convolutional networks serve as the heart of most cutting-edge computer vision systems for a wide range of jobs. CNNs have become commonplace, providing significant advances in a variety of benchmarks. Even though enhanced data size and computational cost seem to translate to instant top notch profits in most tasks (as long as enough labelled data is made available for training), computational efficiency and minimal variable count remain enabling factors for a variety of use cases, including mobile vision and big-data scenarios. We look at how to scale up networks in ways that use the extra computing as effectively as possible, using appropriately factorized convolutions and aggressive regularization.

Another research paper focuses that the Computer technology have been demonstrated to increase agricultural output in a variety of ways. Image processing is one technology that is developing as a helpful tool. This study provides a brief overview of how image processing techniques can help academics and farmers improve agricultural operations. Precision agricultural procedures, weed and pesticide technology, monitoring plant development, and plant nutrition management have all benefited from image processing. This research focuses on the future possibilities of image processing in various agricultural business scenarios.

The practicality of using IR spectroscopy with in identification of yellow dragon disease in fruit trees is investigated in this research. From the refined original data, three datasets were created:

first derivatives, second derivatives, and a merged data which was generated by integrating resampled original data, first and second derivatives. To minimize the amount of features utilized as inputs in the categorization, the resampled datasets were evaluated via principal component analysis , Linear and quadratic discriminant analysis, k-nearest neighbor, and soft independent parameterization of classification analogies were the classification techniques studied . The algorithms' claimed classification accuracies are the mean of these runs.

1.3 ORGANIZATION OF THE REPORT

The remaining chapters of the project report are described as follows:

- Chapter 2 contains the proposed system, methodology, and software details.
- Chapter 3 discusses the results obtained after the project was implemented.
- Chapter 4 concludes the report.
- Chapter 5 consists of codes.
- Chapter 6 gives references.

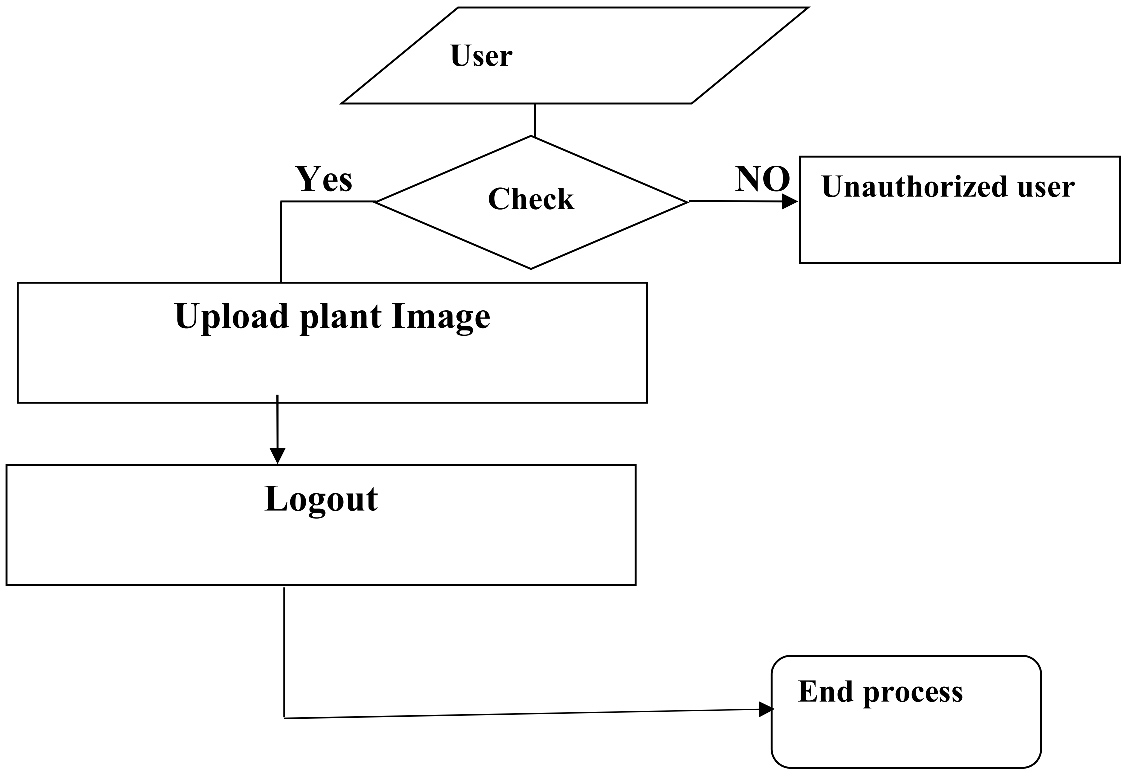
CHAPTER 2

LEAF DISEASE IDENTIFICATION USING DEEP LEARNING

This Chapter describes the proposed system, working methodology, software details.

2.1 PROPOSED SYSTEM

In this project, we utilized convolution neural networks as artificial intelligence to train all plant illness photographs, and then when fresh images are uploaded, CNN would forecast plant disease based on the images. We use cloud services to store the CNN train model and photos. As a result, Al author predicts plant illness, and the cloud is utilized to store data.
It is possible to upload images using a smart phone, but building an android application would incur additional costs and labor, thus we built it as a python online application. Using this online application, the CNN model is trained, and the user may input photographs, after which the programme applies the CNN model to the provided images to forecast illnesses.



2.2 WORKING METHODOLOGY

Numpy, pandas, TensorFlow, SciPy, Scikit-learn, and matplotlib are among the libraries utilized. Flask is a framework for developing web applications in Python with a built-in development server and rapid debugging. First the user uploads the image of leaf upon which the deep learning model identifies the leaf disease with the help of 4 layers.

CONVOLUTION LAYER:

A CNN's fundamental building block is a convolutional layer. It consists of a series of filters (or kernels), the parameters of which must be learned throughout the training process. The filters are often smaller in size than the real image. Each filter interacts with the picture to generate an activation map. The filter is slid over the image's height and breadth for convolution, and the dot product between each element of the filter and the input is calculated at each spatial location. The first activation map entry is calculated by converging the filter with the blue component of the input picture. This technique is repeated for each element of the input picture to build the activation map. The convolutional layer's output volume is formed by stacking the activation maps of each filter along the depth dimension. Every component of the activation map may be thought of as a neuron's output. As a result, each neuron is linked to a tiny local region in the input picture, the size of which equals the size of the filter. All of the neurons in an activation map share parameters as well. Because of the convolutional layer's local connection, the network is driven to train filters that have the best response to a limited region of the input.

POOLING LAYER:

Pooling is a type of down sampling that decreases the dimensionality of a feature map. The corrected feature map is now processed by a pooling layer to produce a pooled feature map. As a result, it decreases the number of parameters to learn as well as the amount of computation done in the network. The pooling layer summarizes the characteristics in an area of the feature map produced by a convolution layer. As a result, subsequent operations are carried out on summarized features rather than precisely positioned features created by the convolution layer. As a result, the model is more resistant to changes in the location of the features in the input picture. We utilised maxpooling in this case. Max pooling is a pooling procedure that chooses the most elements from the covered region of the feature map covered by the filter. As a result, the output of the max-pooling layer would be a feature map that contained the most prominent features of the prior feature map.

FLATTENING LAYER:

The flattening phase is a relatively straightforward stage in the construction of a convolutional neural network. It entails converting the pooled feature map obtained during the pooling process into a one-dimensional vector. We convert the pooled feature map into a one-dimensional vector since it will now be fed into an artificial neural network. To put it another way, this vector will now serve as the input layer of an artificial neural network, which will be chained onto the convolutional neural network we've been working on throughout this course.

FULL CONNECTION:

The final stage is to link an artificial neural network to our current convolutional neural network. The full connection stage is so named because the artificial neural network's hidden layer is replaced by a special sort of hidden layer known as a fully connected layer. A completely linked layer's neurons are, as the name indicates, connected to all of the neurons in the following layer. In a neural network, fully linked layers are those in which all of the inputs from previous layer are connected to succeeding layer.

2.3 SYSTEM DETAILS

This section describes the software details of the system:

2.3.1 SOFTWARE DETAILS

The project is developed using Python. The libraries include TensorFlow, NumPy, Pandas, Matplotlib, Scikit-learn. Flask framework is used to provide us with libraries, tools that allow us to build a web application.

- **Operating System:** Windows
- **Coding Language**: Python 3.7
- **Libraries used**:
    - ➢ TensorFlow

- NumPy
- Pandas
- Matplotlib
- Scikit-learn
- SciPy

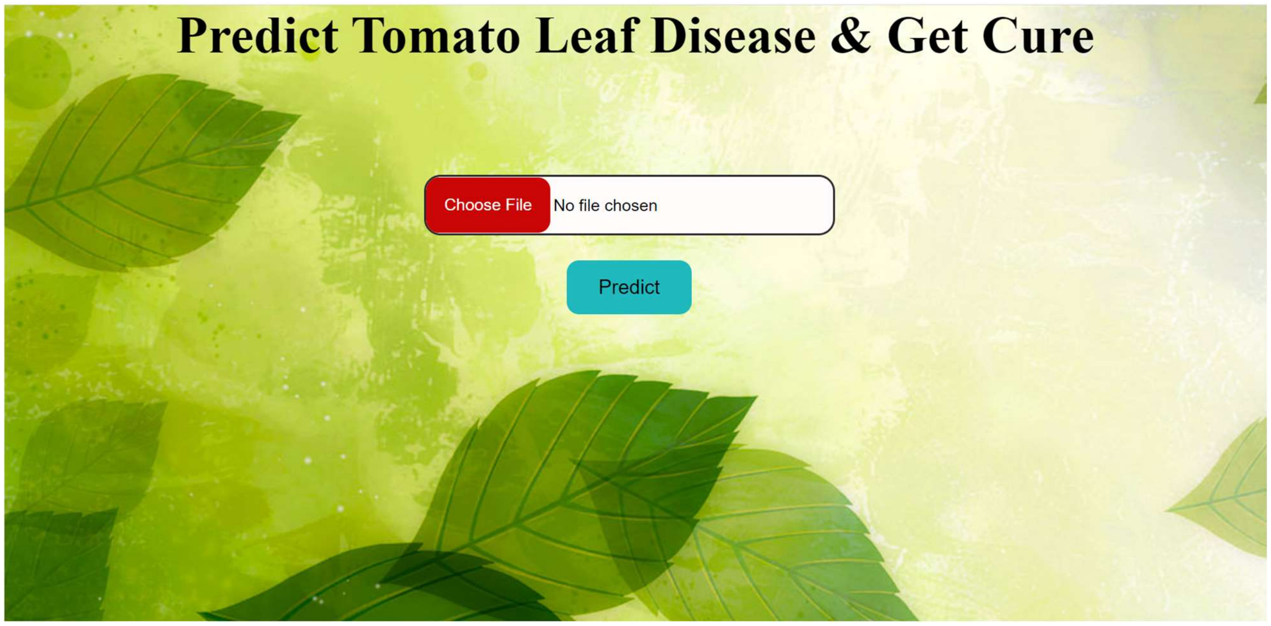
- **Framework**: Flask
- **Templating Engine**: jinja2

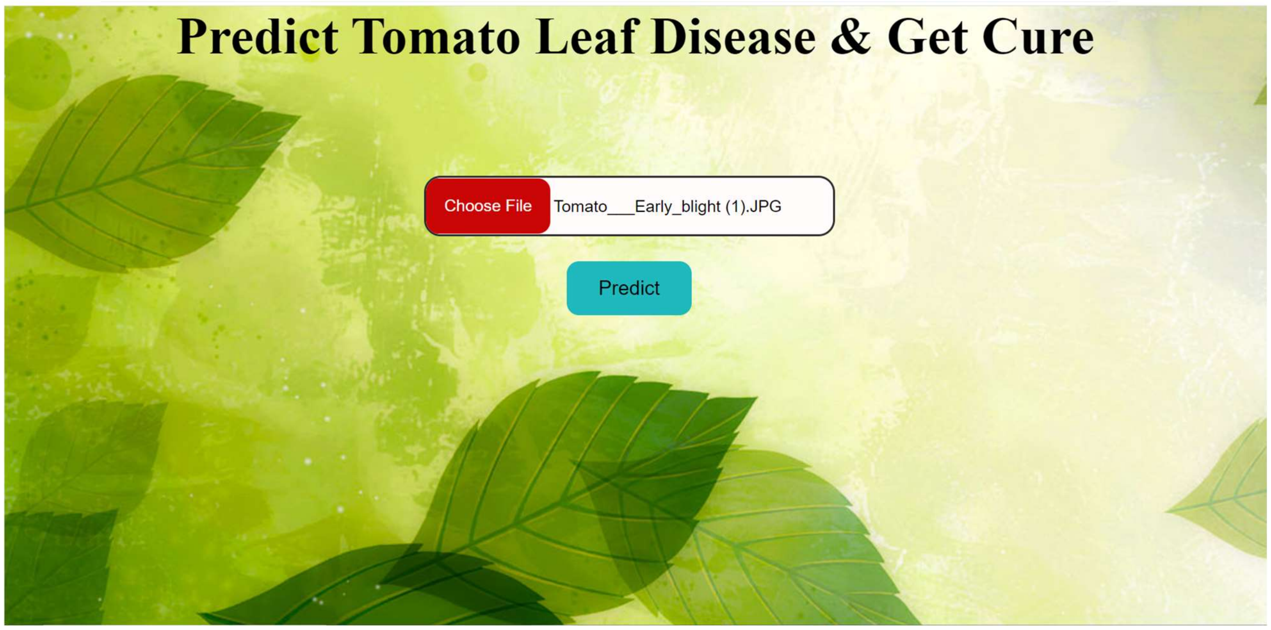
CHAPTER 3

RESULTS AND DISCUSSIONS

a. IMAGE UPLOAD

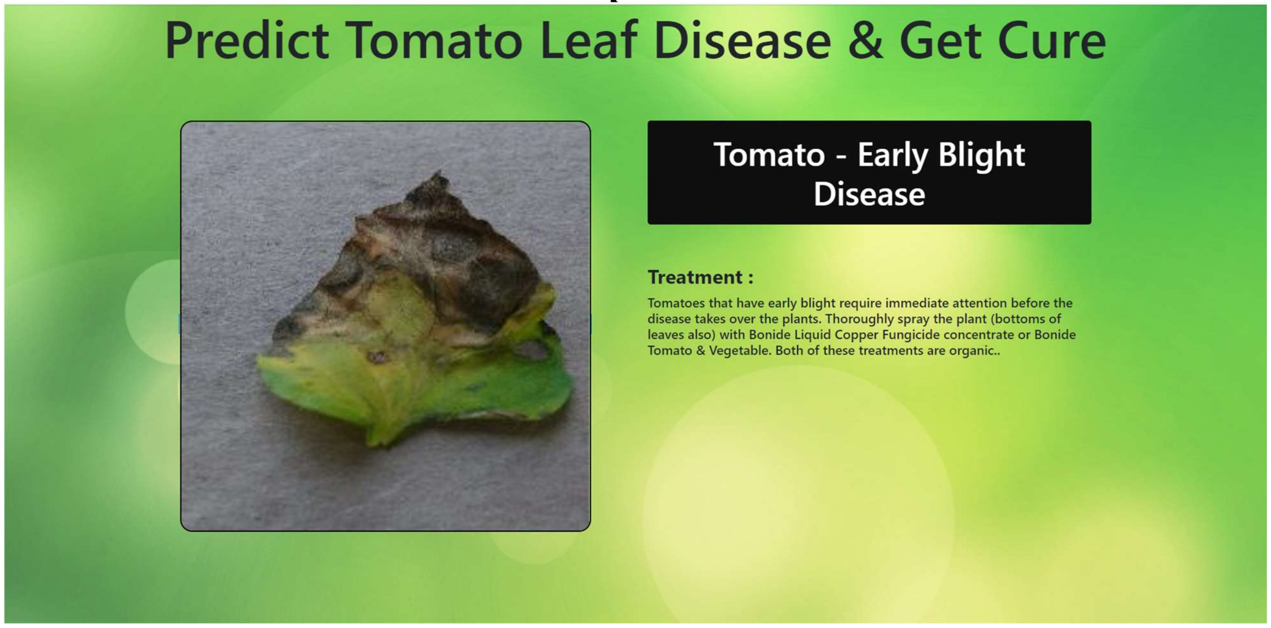
Here user can upload the leaf to find out if there is any disease.




Predict Tomato Leaf Disease & Get Cure
Choose File
Tomato___Early_blight (1).JPG
Predict

b. IDENTIFY TOMATO LEAF DISEASE

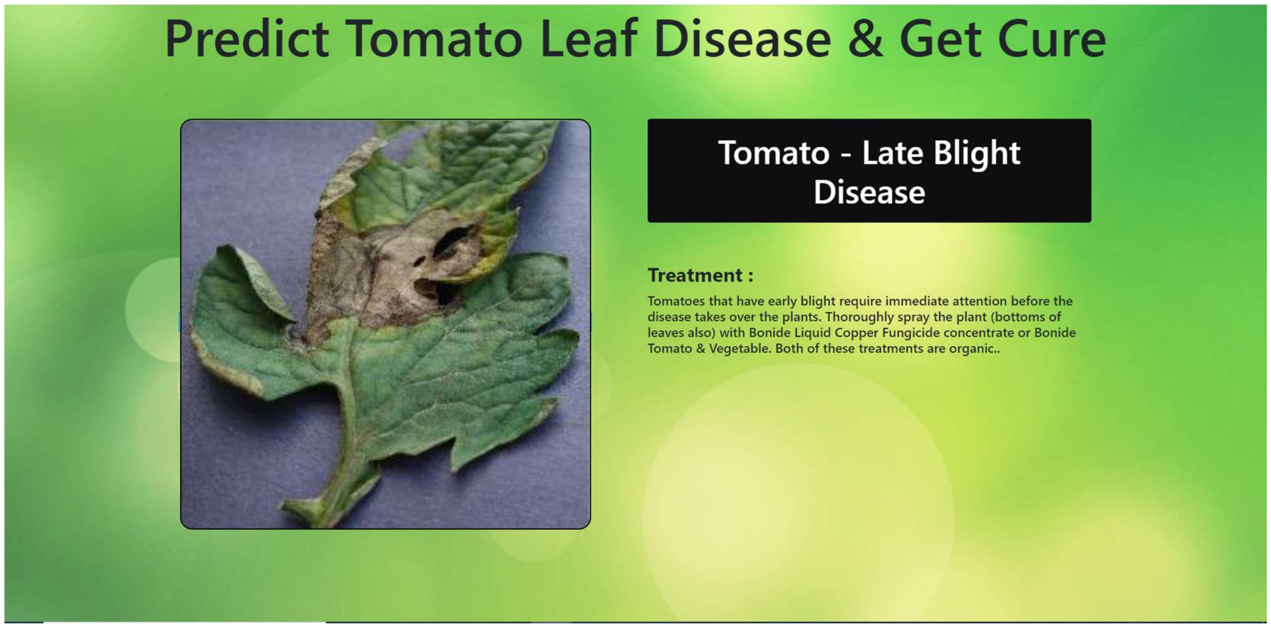
The CNN model identifies the disease present in the tomato leaf.



Tomato - Early Blight Disease

Treatment :

Tomatoes that have early blight require immediate attention before the disease takes over the plants. Thoroughly spray the plant (bottoms of leaves also) with Bonide Liquid Copper Fungicide concentrate or Bonide Tomato & Vegetable. Both of these treatments are organic..

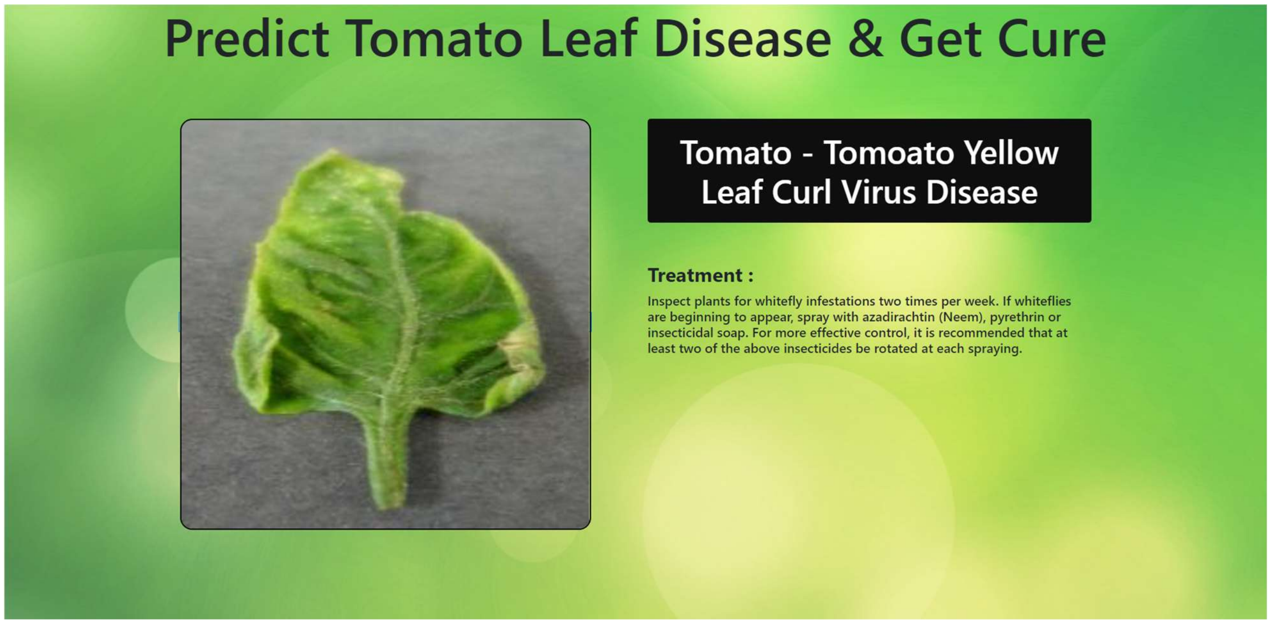


Predict Tomato Leaf Disease & Get Cure

Tomato - Late Blight Disease

Treatment :

Tomatoes that have early blight require immediate attention before the disease takes over the plants. Thoroughly spray the plant (bottoms of leaves also) with Bonide Liquid Copper Fungicide concentrate or Bonide Tomato & Vegetable. Both of these treatments are organic..



Predict Tomato Leaf Disease & Get Cure

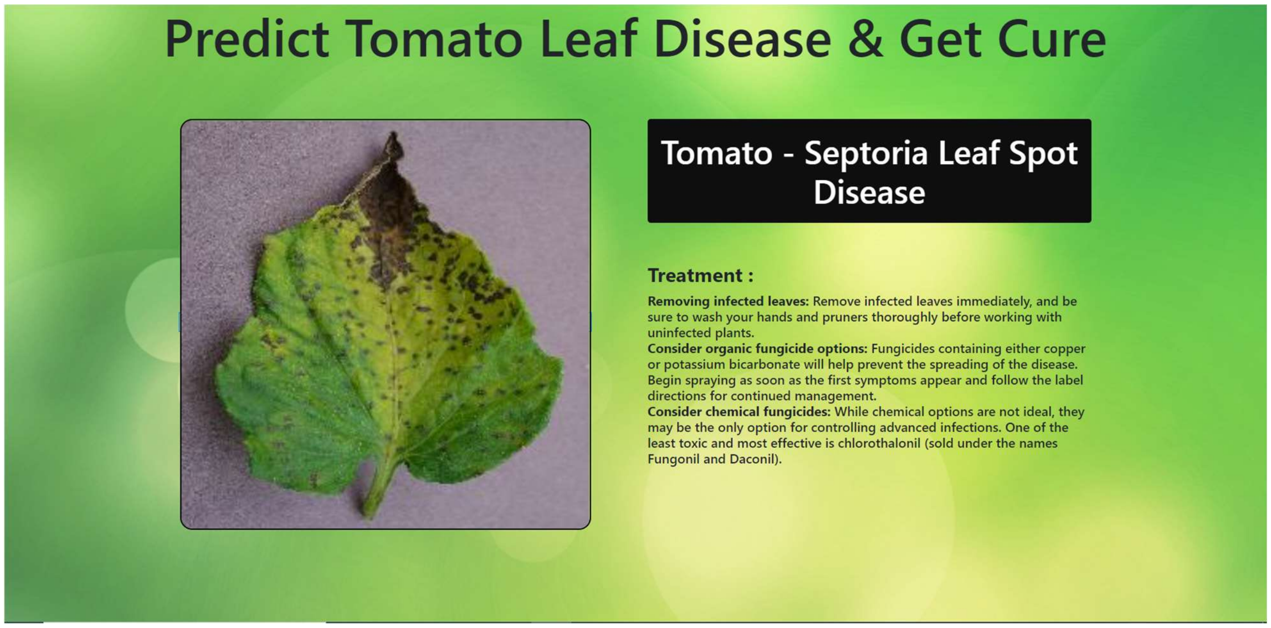
Tomato - Tomoato Yellow Leaf Curl Virus Disease

Treatment :

Inspect plants for whitefly infestations two times per week. If whiteflies are beginning to appear, spray with azadirachtin (Neem), pyrethrin or insecticidal soap. For more effective control, it is recommended that at least two of the above insecticides be rotated at each spraying.

c. TREATMENT

The treatment for the existing leaf disease will be suggested.

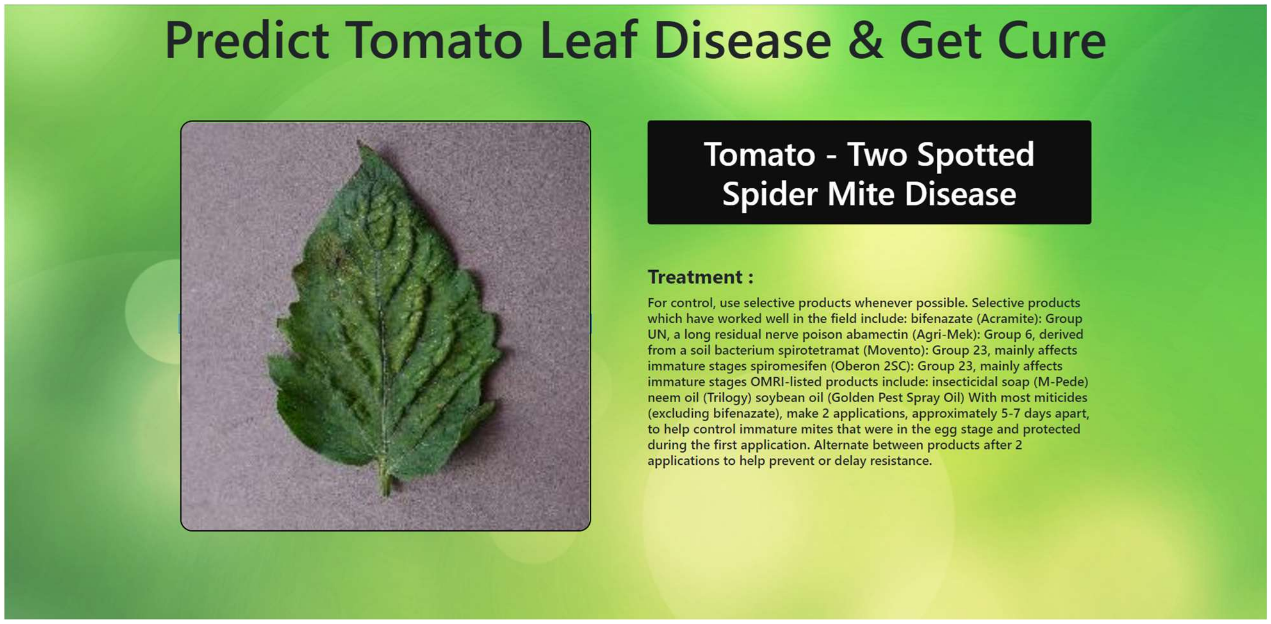


Predict Tomato Leaf Disease & Get Cure

Tomato - Septoria Leaf Spot Disease

Treatment :

Removing infected leaves: Remove infected leaves immediately, and be sure to wash your hands and pruners thoroughly before working with uninfected plants.
Consider organic fungicide options: Fungicides containing either copper or potassium bicarbonate will help prevent the spreading of the disease. Begin spraying as soon as the first symptoms appear and follow the label directions for continued management.
Consider chemical fungicides: While chemical options are not ideal, they may be the only option for controlling advanced infections. One of the least toxic and most effective is chlorothalonil (sold under the names Fungonil and Daconil).

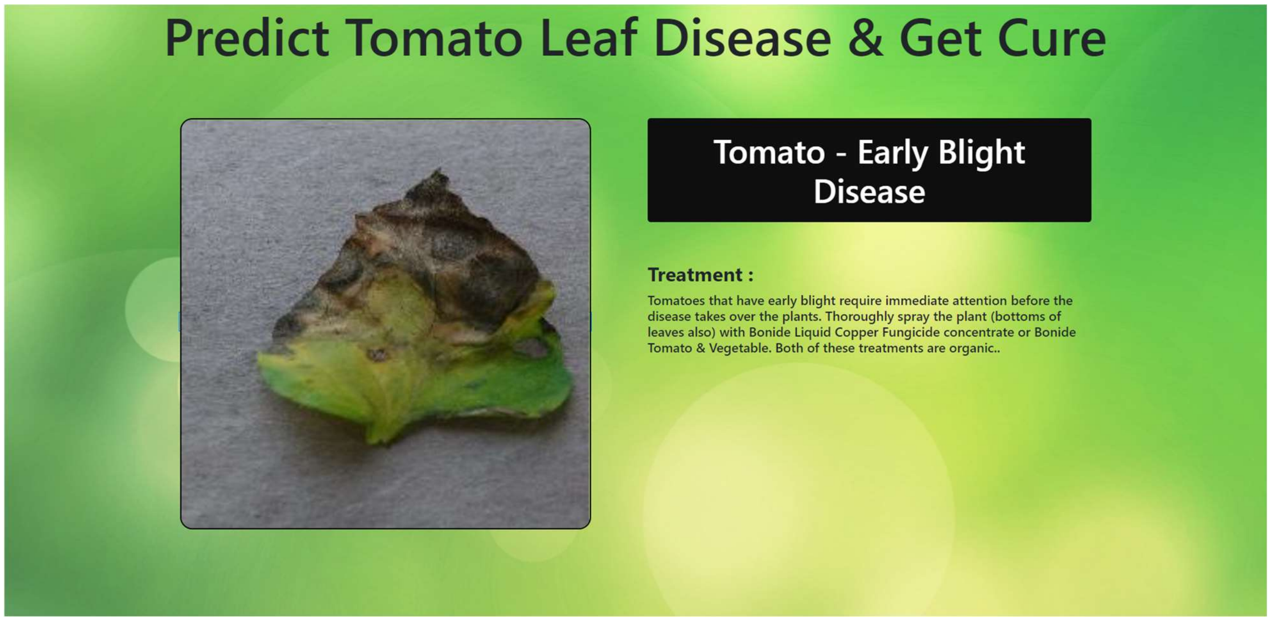


Predict Tomato Leaf Disease & Get Cure

Tomato - Two Spotted Spider Mite Disease

Treatment :

For control, use selective products whenever possible. Selective products which have worked well in the field include: bifenazate (Acramite): Group UN, a long residual nerve poison abamectin (Agri-Mek): Group 6, derived from a soil bacterium spirotetramat (Movento): Group 23, mainly affects immature stages spiromesifen (Oberon 2SC): Group 23, mainly affects immature stages OMRI-listed products include: insecticidal soap (M-Pede) neem oil (Trilogy) soybean oil (Golden Pest Spray Oil) With most miticides (excluding bifenazate), make 2 applications, approximately 5-7 days apart, to help control immature mites that were in the egg stage and protected during the first application. Alternate between products after 2 applications to help prevent or delay resistance.


Predict Tomato Leaf Disease & Get Cure

Tomato - Early Blight Disease

Treatment :

Tomatoes that have early blight require immediate attention before the disease takes over the plants. Thoroughly spray the plant (bottoms of leaves also) with Bonide Liquid Copper Fungicide concentrate or Bonide Tomato & Vegetable. Both of these treatments are organic..

CHAPTER 4

CONCLUSION AND FUTURE WORK

This project proposes an automated, low-cost, and simple-to-use end-to-end solution to one of the most difficult challenges in the agricultural domain for farmers: precise, instant, and early diagnosis of crop diseases and knowledge of disease outbreaks, which would aid in quick decision making for disease control measures. The high-performance deep CNN model "Inception" offers real-time illness categorization in the platform through a web interface. Overall, the results of our project show that the proposal has significant practical deployment potential due to multiple dimensions: the infrastructure is highly scalable, and the underlying algorithm works accurately even with a large number of disease categories, performs better with high fidelity real-life training data, improves accuracy with increase in the training dataset, is capable of detecting early symptoms of diseases, and can succumb to failure.

Future studies will entail improving the model to incorporate more parameters that might improve the disease association. To increase our model accuracy and allow disease predictions, we may supplement the picture database with supporting inputs from the farmer on soil, prior fertiliser and pesticide application, as well as publically accessible climatic indicators like as temperature, humidity, and rainfall. A simple approach of establishing the threshold based on the mean of all classification scores may be utilised for automated admission of user-uploaded photos into the Training Database for greater classification accuracy and as little human interaction as feasible. This might be used to provide automated time-based monitoring of disease density maps, which can be used to track illness progression and trigger alerts. Predictive analytics may be used to notify consumers of potential illness outbreaks near their location.

CHAPTER 5

APPENDIX

CODES

Leaf.python code

```
#Import necessary libraries
from flask import Flask, render_template, request

import numpy as np
import os

from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model

filepath = 'C:\\Users\\Lenovo\\Desktop\\Plant-Leaf-Disease-Prediction-main\\model.h5'
model = load_model(filepath)
print(model)

print("Model Loaded Successfully")

def pred_tomato_dieas(tomato_plant):
  test_image = load_img(tomato_plant, target_size = (128, 128)) # load image
  print("@@ Got Image for prediction")

  test_image = img_to_array(test_image)/255 # convert image to np array and normalize
  test_image = np.expand_dims(test_image, axis = 0) # change dimention 3D to 4D

  result = model.predict(test_image) # predict diseased palnt or not
  print('@@ Raw result = ', result)

  pred = np.argmax(result, axis=1)
  print(pred)
  if pred==0:
      return "Tomato - Bacteria Spot Disease", 'Tomato-Bacteria Spot.html'
  elif pred==1:
     return "Tomato - Early Blight Disease", 'Tomato-Early_Blight.html'

  elif pred==2:
     return "Tomato - Healthy and Fresh", 'Tomato-Healthy.html'

  elif pred==3:
     return "Tomato - Late Blight Disease", 'Tomato - Late_blight.html'

  elif pred==4:
     return "Tomato - Leaf Mold Disease", 'Tomato - Leaf_Mold.html'

  elif pred==5:
     return "Tomato - Septoria Leaf Spot Disease", 'Tomato - Septoria_leaf_spot.html'

  elif pred==6:
     return "Tomato - Target Spot Disease", 'Tomato - Target_Spot.html'

  elif pred==7:
     return "Tomato - Tomoato Yellow Leaf Curl Virus Disease", 'Tomato - 
Tomato_Yellow_Leaf_Curl_Virus.html'
  elif pred==8:
     return "Tomato - Tomato Mosaic Virus Disease", 'Tomato - Tomato_mosaic_virus.html'

  elif pred==9:
     return "Tomato - Two Spotted Spider Mite Disease", 'Tomato - Two-
spotted_spider_mite.html'



# Create flask instance
app = Flask(__name__)

# render index.html page
@app.route("/", methods=['GET', 'POST'])
def home():
        return render_template('index.html')


# get input image from client then predict class and render respective .html page for solution
@app.route("/predict", methods = ['GET','POST'])
def predict():
     if request.method == 'POST':
        file = request.files['image'] # fet input
        filename = file.filename
        print("@@ Input posted = ", filename)
        file_path = os.path.join('C:\\Users\\Lenovo\\Desktop\\Plant-Leaf-Disease-Prediction-
main\\static\\upload\\', filename)
        file.save(file_path)

        print("@@ Predicting class......")
        pred, output_page = pred_tomato_dieas(tomato_plant=file_path)

        return render_template(output_page, pred_output = pred, user_image = file_path)

# For local system & cloud
if __name__ == "__main__":
    app.run(threaded=False,port=8081)
```

Training.python code

```
from tensorflow.compat.v1 import ConfigProto

from tensorflow.compat.v1 import InteractiveSession

config = ConfigProto()

config.gpu_options.allow_growth = True

session = InteractiveSession(config=config)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import tensorflow as tf

tf.compat.v1.disable_eager_execution()

import matplotlib.pyplot as plt

import numpy as np

import os


#basic cnn

# Initialising the CNN
```

```
classifier = Sequential()

# Step 1 - Convolution
classifier.add(Conv2D(32, (3, 3), input_shape = (128, 128, 3), activation = 'relu'))

# Step 2 - Pooling
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Adding a second convolutional layer
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Step 3 - Flattening
classifier.add(Flatten())

# Step 4 - Full connection
classifier.add(Dense(units = 128, activation = 'relu'))
classifier.add(Dense(units = 10, activation = 'sigmoid'))

# Compiling the CNN
classifier.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = 0.2,
horizontal_flip = True)
test_datagen = ImageDataGenerator(rescale = 1./25++5)
training_set = train_datagen.flow_from_directory('C:\\Users\\Lenovo\\Desktop\\Plant-Leaf-
Disease-Prediction-main\\Dataset\\train', # relative path from working directoy
                                    target_size = (128, 128),
                                    batch_size = 6, class_mode = 'categorical')
valid_set = test_datagen.flow_from_directory('C:\\Users\\Lenovo\\Desktop\\Plant-Leaf-Disease-
Prediction-main\\Dataset\\val', # relative path from working directoy
                                 target_size = (128, 128),
                          batch_size = 3, class_mode = 'categorical')

labels = (training_set.class_indices)
print(labels)
classifier.fit_generator(training_set,
            steps_per_epoch = 20,
            epochs = 50,
            validation_data=valid_set

            )

classifier_json=classifier.to_json()
with open("model1.json", "w") as json_file:
  json_file.write(classifier_json)
# serialize weights to HDF5
  classifier.save_weights("my_model_weights.h5")
  classifier.save("model.h5")
  print("Saved model to disk")

'''
import cv2
from matplotlib import pyplot as plt
import os
import numpy as np
img=cv2.imread('C:/Users/Madhuri/AppData/Local/Programs/Python/Python38/Leaf_disease/dat
a/d (7)_iaip.jpg')
img_resize = cv2.resize(img, (128,128))
CV2 reads an image in BGR format. We need to convert it to RGB
b,g,r = cv2.split(img_resize)       # get b,g,r
rgb_img = cv2.merge([r,g,b])     # switch it to rgb
```

```
plt.imshow(rgb_img)
label_map = (training_set.class_indices)
print(label_map)
img_rank4 = np.expand_dims(rgb_img/255, axis=0)
classifier.predict(img_rank4)
h = list(label_map.keys())[classifier.predict_classes(img_rank4)[0]]
font = cv2.FONT_HERSHEY_DUPLEX
cv2.putText(img, h, (10, 30), font, 1.0, (0, 0, 255), 1)
cv2.imshow(h,img)

print(h)
'''
```

Example.python code

```
import cv2
from matplotlib import pyplot as plt
import os
import numpy as np
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model

filepath = 'C:\\Users\\Lenovo\\Desktop\\Plant-Leaf-Disease-Prediction-main\\model.h5'
model = load_model(filepath)
print(model)

print("Model Loaded Successfully")
```

```
tomato_plant = cv2.imread('C:\\Users\\Lenovo\\Desktop\\Plant-Leaf-Disease-Prediction-
main\\Dataset\\test\\Tomato___Bacterial_spot (1).JPG')

test_image = cv2.resize(tomato_plant, (128,128)) # load image


test_image = img_to_array(test_image)/255 # convert image to np array and normalize

test_image = np.expand_dims(test_image, axis = 0) # change dimention 3D to 4D


result = model.predict(test_image) # predict diseased palnt or not


pred = np.argmax(result, axis=1)

print(pred)

if pred==0:

   print( "Tomato - Bacteria Spot Disease")


elif pred==1:

   print("Tomato - Early Blight Disease")


elif pred==2:

   print("Tomato - Healthy and Fresh")


elif pred==3:

   print("Tomato - Late Blight Disease")


elif pred==4:

   print("Tomato - Leaf Mold Disease")


elif pred==5:

   print("Tomato - Septoria Leaf Spot Disease")


elif pred==6:
    print("Tomato - Target Spot Disease")

elif pred==7:
      print("Tomato - Tomoato Yellow Leaf Curl Virus Disease")
elif pred==8:
      print("Tomato - Tomato Mosaic Virus Disease")

elif pred==9:
      print("Tomato - Two Spotted Spider Mite Disease")
```

index.html code

```
<html>
<head>
<style>
body  {
  background-image: {{url_for('static',filename = 'images/Background.jpg')}};;
  background-repeat: no-repeat;
  background-size: auto;
}
* {
            margin: 0px;
            padding: 0px;
            box-sizing: border-box;
        }
        form {
            display: flex;
            height: 85vh;
            justify-content: center;
            align-items: center;
```

```
    margin-top: -150px;
    width: 60%;
    text-align: center;
                        margin-left:300px;

}
.details h2 {
    position: relative;
    top: 100px;
    margin: auto;
    color: rgb(18, 231, 231);
    font-size: 3rem;
}

label:hover {
    transform: scale(1.03);
}

.details h2 {
    /* margin-bottom: 300px; */
    position: relative;
    top: 100px;
    margin: auto;
    color: rgb(18, 231, 231);
    font-size: 3rem;

}
.details h1 {
    color: white;
    padding: 20px;
    border-radius: 15px;
    background-color: rgb(8, 8, 8);
}
.upload {
    font-size: 20px;
    background-color: rgb(255, 252, 252);
    border-radius: 20px;
    outline: none;
    width: 500px;
    color: rgb(0, 0, 0);
    border: 3px solid rgb(45, 47, 49);
}
::-webkit-file-upload-button {
    color: rgb(255, 252, 252);
    padding: 20px;
    border: 2px solid rgb(201, 6, 6);
    background-color: rgb(201, 6, 6);
    border-radius: 15px;


}
::-webkit-file-upload-button:hover {
    border-radius: 20px;
    border: 2px solid rgb(177, 174, 174);
}
input[type="submit"] {
    position: absolute;
    margin-top: 200px;
    padding: 15px 35px;
    background-color: rgb(31, 185, 190);
    border-radius: 15px;
```

```
            color: black;

            font-size: 1.5rem;

            border: 4px solid rgb(31, 185, 190);

        }

</style>

</head>

<body background="{{url_for('static',filename = 'images/Background.jpg')}}">

            <h1 style="text-align:center;font-size:4rem">Predict Tomato Leaf Disease & Get
Cure</h1>

<section>

            <form action="/predict" method="post" enctype="multipart/form-data"
onsubmit="showloading()">

                <input type="file" name="image" class="upload">

                                        <br>

                                        <br>

                <input type="submit" value="Predict">


            </form>

        </div>

    </section>


</body>

</html>
```

Tomato-late blight html code

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.1/css/bootstrap.min.css"
```

```
        integrity="sha384-
VCmXjywReHh4PwowAiWNagnWcLhlEJLA5buUprzK8rxFgeH0kww/aWY76TfkUoSX"
crossorigin="anonymous">

    <title>TOMATO LEAF DISEASE PREDICTION</title>

    <style>
        * {
            margin: 0px;
            padding: 0px;
            box-sizing: border-box;
        }

        .border img {
            border-radius: 15px;
            border: 2px solid black;
        }

    </style>
</head>

<body background="{{url_for('static',filename = 'images/Back.jpg')}}">

    <div>
        <h1 style="text-align:center;font-size:4rem">Predict Tomato Leaf Disease & Get Cure</h1>
    </div>

   <br>
   <br>


    <div class="container my-2">
        <div class="row mb-5">

            <div class="col-sm" style="margin-bottom: 23px;">
                <span class="border border-primary">
                    <img src="{{url_for('static',filename = 'images/Tomato___Late_blight.JPG' )}}"
alt="User Image" style="width:500px;height:500px;">
                </span>
            </div>

            <div class="col-sm">

                <div>
                    <h1 style="padding: 15px; background-color: rgb(15, 15, 15); color: white;"
                        class="text-center mb-5 content-h1 rounded">
                        {{pred_output}} </h1>
                </div>
                <div class="details">
```

```
                                        <h4><b> Treatment : </b> </h4>
                    <h6>

                        Tomatoes that have early blight require immediate attention before the disease
takes over the plants. Thoroughly spray the plant (bottoms of leaves also) with Bonide Liquid
Copper Fungicide concentrate or Bonide Tomato & Vegetable. Both of these treatments are
organic..</br></br>


                    </h6>
                </div>
            </div>
        </div>


    </div>
</body>

</html>
```

Main.css

```
.img-preview {
    width: 256px;
    height: 256px;
    position: relative;
    border: 5px solid #F8F8F8;
    box-shadow: 0px 2px 4px 0px rgba(0, 0, 0, 0.1);
    margin-top: 1em;
    margin-bottom: 1em;
}

.img-preview>div {
    width: 100%;
    height: 100%;
    background-size: 256px 256px;
    background-repeat: no-repeat;
    background-position: center;
}

input[type="file"] {
    display: none;
}

.upload-label{
    display: inline-block;
    padding: 12px 30px;
    background: #39D2B4;
    color: #fff;
    font-size: 1em;
    transition: all .4s;
    cursor: pointer;
}

.upload-label:hover{
    background: #34495E;
    color: #39D2B4;
}

.loader {
    border: 8px solid #f3f3f3; /* Light grey */
    border-top: 8px solid #3498db; /* Blue */
    border-radius: 50%;
    width: 50px;
    height: 50px;
    animation: spin 1s linear infinite;
}

@keyframes spin {
    0% { transform: rotate(0deg); }
    100% { transform: rotate(360deg); }
}
```

REFERENCES

[1] L. Saxena and L. Armstrong, "A survey of image processing techniques for agriculture," in Proceedings of Asian Federation for Information Technology in Agriculture, 2014, pp. 401-413.

[2] https://www.frontiersin.org/articles/10.3389/fpls.2016.01419/full

[3] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems, 2012.

[4] TensorFlow.[Online].Available: https://www.tensorflow.org/

[5] https://ieeexplore.ieee.org/document/9395751

[6] S. Raza, G. Prince, J. P. Clarkson and N. M. Rajpoot, "Automatic detection of diseased tomato plants using thermal and stereo visible light images," in PLoS ONE, 2015.

[7] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9492343/

[8]https://thesai.org/Publications/ViewPaper?Volume=13&Issue=1&Code=IJACSA&SerialNo=38

[9] C. Szegedy,"Rethinking the inception architecture for computer vision," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818-2826.

[10]https://www.slideshare.net/javaidahmad23/plant-disease-detection-and-classification-using-deep-learning-240805631

[11]https://www.analyticsvidhya.com/blog/2021/08/beginners-guide-to-convolutional-neural-network-with-implementation-in-python/

BIODATA

Name	: Roopa Chowdary Cherukuri
Mobile Number	: +91 9848632999
E-mail	: roopa.19bce7049@vitap.ac.in
Permanaent Address	: D no: 25-102, Near Pothureddipalli Road, R R Peta, Nuzvid-521021, AP

Name	: Allu Meghana Chowdary
Mobile Number	: +91 8790194246
E-mail	: meghana.19bcd7105@vitap.ac.in
Permanaent Address	: Flot no :405, Manjunadha Towers, Prakash nagar, Narasaraopet-522601, AP