A MINI-PROJECT REPORT ON

PLANT DISEASE DETECTION USING MACHINE LEARNING

Submitted in partial fulfilment of requirementsfor the award of the degree of

MASTER OF COMPUTER APPLICATIONS

Submitted by:

M.GURU MOUNIKA (23091F0009)

Under the Guidance of

Mr. B. RAMESWARA REDDY

Assistant Professor, Dept. of MCA

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

RAJEEV GANDHI MEMORIAL COLLEGE OF ENGINEERING & TECHNOLOGY(AUTONOMOUS)

AFFILIATED TO J.N.T UNIVERSITY ANANTAPUR. ACCREDITED BY NBA (TIER-1) & NAAC OF UGC. NEW DELHI, WITH A+ GRADE
RECOGNIZED UGC-DDU KAUSHAL KENDRANANDYAL-518501, (Estd-1995)

YEAR: 2024-2025

Rajeev Gandhi Memorial College of Engineering &Technology

(AUTONOMOUS)

AFFILIATED TO J.N.T UNIVERSITY ANANTAPUR. ACCREDITED BY NBA (TIER-1) & NAAC OF UGC. NEW DELHI, WITH A+ GRADE
RECOGNIZED UGC-DDU KAUSHAL KENDRA
NANDYAL-518501, (Estd-1995)

(ESTD – 1995)

(ESTD-1995)

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that **M. GURU MOUNIKA** (23091F0009), of MCA III- semester, has carried out the mini-project work entitled "**PLANT DISEASE DETECTION USING MACHINE LEARNING**" under the supervision and guidance of Mr. B. RAMESWARA REDDY in partial fulfilment of the requirements for the award of Degree of **Master of Computer Applications** from **Rajeev Gandhi Memorial College of Engineering & Technology (Autonomous)**, Nandyal is a bonafied record of the work done by him during 2024-2025.

Project Guide

Mr. B. Rameswara Reddy
Assistant Professor, Dept. of MCA

Head of the Department

Dr. K. Subaa Reddy MTech, Ph.D.
Professor, Dept. of CSE

Place: Nandyal
Date:

External Examiner

ACKNOWLEDGEMENT

I manifest my heartier thankfulness pertaining to your contentment over my project guide **Mr. B. Rameswara Reddy** in Department of CSE with whose adroit concomitance the excellence has beenexemplified in bringing out this project to work with artistry.

I express my gratitude to **Dr. K. Subba Reddy garu,** Head of the Department of Computer Science Engineering & MCA departments, all teaching and non-teaching staff of the Computer Science Engineering department of Rajeev Gandhi memorial College of Engineering and Technology for providing continuous encouragement and cooperation at various steps of my project.

Involuntarily, I am perspicuous to divulge my sincere gratefulness to our Principal, **Dr. T. Jaya Chandra Prasad garu,** who has been observed posing virtue in abundance towards my individuality to acknowledge my project work tangentially.

At the outset I thank to honorable **Chairman Dr. M. SanthiRamudu garu,** for providing us with exceptional faculty and moral support throughout the course.

Finally, I extend my sincere thanks to all the **Staff Members** of MCA & CSE Departments who have co-operated and encouraged us in making my project successful.

Whatever one does, whatever one achieves, the first credit goes to the **Parents** be it not for their love and affection, nothing would have been responsible. I have seen every good that happens to us their love and blessings.

By:

M. GURU MOUNIKA

(23091F0009)

CONTENTS

LIST OF FIGURES

ABSTRACT

In nations like India, where agriculture is the primary sector for economic growth, it is essential for meeting the world's food demands. On the other hand, unusual weather patterns and a number of climatic changes might encourage serious fungal, viral, and bacterial infections in plants. The food supply may be seriously threatened by certain plant diseases; thus, it is crucial to spot them early and protect plants from them. Traditional methods took a long time and were relied on subject-matter specialists. diverse disease identification methodologies employing diverse domains have been developed in light of the fact that technology is always evolving and offers many benefits in the field of plant leaf disease detection.

Extreme weather and various climatic changes, however, can encourage serious fungal, viral, and bacterial infections in plants. It is crucial to recognise and protect plants from these diseases at the earliest stages because they pose a serious risk to the supply of food. The traditional methods took a lot of time since they required subject-matter specialists. In order to identify and treat plant diseases that affect plant leaves, a variety of disease identification methodologies employing different domains have been presented in the literature. This is because technology is advancing daily and has many benefits in the field of plant leaf disease detection. Despite the fact that many of the current methods have produced improved outcomes, there are still problems.

Keywords: Plant leaf disease detection, Machine learning, Deep learning,
CNN (Convolutional Neural Networks), Plant leaf disease classification, Transfer Learning.

CHAPTER 1

INTRODUCTION

1.1 Overview

Plant disease detection using machine learning and deep learning techniques has gained significant attention in recent years due to its potential in improving crop management and reducing yield losses. In particular, deep learning models, such as Convolutional Neural Networks (CNNs), have demonstrated promising results in accurately identifying and classifying plant diseases.

Transfer learning is a technique in deep learning where a pre-trained model, trained on a large dataset, is used as a starting point for a new task. This approach has several advantages, including reducing the need for large amounts of labeled data and leveraging the knowledge learned from the pre-trained model. Some commonly used pre-trained models for plant disease detection are VGG 16, Inception V3, and MobileNetV2.

1.2. Problem Definition

The problem of plant disease detection refers to the task of identifying and classifying diseases or abnormalities in plants based on visual symptoms exhibited by the plant's leaves. The objective is to develop a reliable and accurate system that can assist farmers, agricultural experts, and researchers in detecting and diagnosing plant diseases in a timely manner.

1.3. Objectives

The objective of plant disease detection is to promptly and reliably diagnose plant diseases in order to stop their spread and lessen crop damage. Plant disease identification is crucial for a number of reasons, including:

- High Accuracy: The objective is to achieve a high level of accuracy in identifying and classifying plant diseases.

1.4. Methodology

An alternative methodology is proposed, predicated on the concept of transfer learning to overcome existing limitations and achieve minimal pre-processing, compute time, and low loss function, resulting in enhanced plant identification accuracy. This will use the MobileNetV2 Architecture to apply the transfer learning idea, which has been pretrained on image dataset and outperforms the other architectures.

1.5. Features

- Object Recognition: Detects and identifies objects based on characteristics like shape, color, and size.
- Localization: Determines the position and dimensions of objects with in an image or video.
- Classification: Categorizes objects into different types based on their visual attributes.
- Real-time Processing: Processes images or videos quickly for immediate decision-making.
- Accuracy: Achieves high accuracy in recognizing and classifying objects using sophisticated algorithms.

1.6. Advantages

- Early Detection: Allows early identification of plant diseases for quick intervention and minimal crop loss.
- Non-destructive: Offers a non-destructive way to monitor plants continuously without harm.
- High Throughput: Quickly analyzes numerous plant samples, ideal for large- scale agricultural applications.
- Cost-effective**:** Reduces costs compared to manual inspections and lab tests, making it accessible for all farmers.

1.7. Applications

- Early Disease Detection: Timely identification of plant diseases top revent extensive crop damage and loss.

- Precision Agriculture: Reduces chemical use by applying pesticides and fungicides only where needed based on disease detection.
    - Crop Management: Helps farmers optimize irrigation and fertilization by monitoring plant health across fields.

    •Disease Monitoring and Surveillance: Tracks plant disease outbreaks and trends for proactive.

CHAPTER 2

LITERATURE SURVEY

2.1 RELATED WORK

RELATED WORK-1

Efficient Disease Detection of Paddy Crop using CNN

Authors: Dr. P.A. Harsha Vardhini, S. Asritha, Y. Susmitha Dev

This research, led by Dr. P.A. Harsha Vardhini S. Asritha, and Y. Susmitha Devi, explores the application of Convolutional Neural Networks (CNNs) and Artificial Intelligence to detect diseases in paddy crops effectively. The system stands out for its cost-effectiveness and user- friendly design, making it highly accessible to many users, especially in agricultural sectors where resources are often limited. However, its specific focus on paddy crops limits its more comprehensive application across other crop types. Despite its few drawbacks, such as dependence on high-quality data and potential technical challenges, the future scope of this project is promising. The team aims to integrate more advanced technologies to enhance its robustness and applicability, potentially making it a cornerstone tool in agricultural disease management.

Pros:

- Cost Efficiency: The model is designed to be economically feasible for widespread use, reducing the financial barrier for small and large-scale farmers.
- Accuracy in Disease Detection: Utilizes the robust capabilities of CNNs to detect diseases, accurately enhancing paddy crop health and productivity.
- Quick Diagnosis: Allows rapid disease detection, enabling timely interventions to prevent widespread crop damage.

Cons:

- Limited Crop Scope: Specifically designed for paddy crops, which restricts its applicability to other types of crops.
- Dependence on Data Quality: The accuracy of disease detection depends on the quality
  and variety of training data provided.

RELATED WORK-2:

Plant Disease Identification using a Novel Convolution Neural Network

Authors: SK Mahmudul Hassan, Arnab Kumar Maji

In their research, SK Mahmudul Hassan and Arnab Kumar Maji introduce a cutting-edge approach to plant disease identification using a novel Convolutional Neural Network (CNN) that leverages inception and residual connections. This innovative model excels in effective image representation, significantly enhancing its ability to detect and analyze plant diseases. Despite its advanced capabilities, the model faces challenges due to its training on an imbalanced dataset, which may affect overall accuracy. Furthermore, the complexity of the model increases the computational demands and sensitivity to the data quality. The authors plan to incorporate clustering-based techniques to refine the disease identification process further, aiming to improve the model's robustness and applicability to a broader range of plant health issues. This development holds promising potential for agricultural technology, especially in enhancing crop management and disease prevention strategies.

Pros:

- Effective Image Representation: The novel CNN model enhances feature representation with inception and residual connections.
- Improved Detection Capabilities: The advanced architecture allows for better extraction and learning of complex patterns in plant images.

- Versatility in Application: Adapts well to different plant disease scenarios, improving usability across various plant types.

Cons:

- The Imbalanced Dataset Usage: The model was trained on an imbalanced dataset, which can lead to lower accuracy compared to training with a balanced dataset.
- Potential Overfitting: Risk of overfitting due to complex CNN model, especially with limited or skewed data.

RELATED WORK-3:

Soil Sensors-Based Prediction System for Plant Diseases Using Exploratory Data Analysis and Machine Learning.

Authors: Manish Kumar, Ahlad Kumar, Vinay S. Palaparth

The authors have developed a sophisticated soil sensors-based prediction system that utilizes exploratory data analysis and machine learning to identify plant diseases early and accurately. This innovative system leverages Artificial Neural Networks (ANN) for pattern recognition and classification, providing significant benefits in early disease detection and accurate predictions. These capabilities allow for more timely and effective interventions, ultimately leading to healthier crops and reduced losses. However, the complexity of the machine learning models and the dependence on high-quality data present challenges in system implementation and operation. Despite these hurdles, the future scope of this project includes incorporating more sensors and improving predictive models to enhance the system's robustness and applicability across different agricultural environments. This ongoing development promises to revolutionize plant disease management through advanced technology, making it a valuable tool for farmers and agronomists.

Pros:

Early Disease Detection: The system enables early identification of potential plant diseases, allowing for prompt intervention.

- Cost-Effective Monitoring: Reduces the need for extensive physical field checks, lowering labor costs and resources.

Cons:

- Data Dependency: The accuracy of predictions heavily depends on the quality and volume of data collected.
- Limited by Sensor Capabilities: The prediction system's performance is limited by the capabilities and accuracy of the sensors used.

RELATED WORK-4:

Internet of Things (IoT) and Machine Learning Model of Plant Disease Prediction—Blister Blight for Tea Plant

Authors: Liu, Rab Nawaz Bashir, Muhammad Tausif, Qasim Umer, Salman Iqbal, Malik Muhammad Ali Shahid

In the paper authored by Zhiyan Liu, Rab Nawaz Bashir, Muhammad Tausif, Qasim Umer, Salman Iqbal, and Malik Muhammad Ali Shahid, an innovative Internet of Things (IoT) and Machine Learning model is presented for predicting plant diseases, specifically Blister Blight in tea plants. This system leverages supervised learning algorithms like decision trees to enable real-time monitoring and provide data-driven insights, significantly enhancing the management and health of tea crops. The IoT technology allows for early detection of diseases, scalable monitoring solutions, and improved resource management, leading to better decision-making processes for farmers. Challenges such as high costs, technical complexity, and data security concerns must be addressed despite the advantages. Future developments aim to leverage advanced IoT technologies, overcoming current limitations and offering more robust solutions for sustainable agriculture.

Pros:

- Real-time Monitoring: Enables continuous observation and tracking ofplant health conditions.
- Improved Detection Capabilities: Improved Resource Management: Optimizes the use of water, Fertilizers and pesticides through precise application based on plant needs.

Cons:

- Cost: A high initial investment is required for setting up IoT devices and infrastructure.
- Technical Complexity: Requires specialized knowledge to install, maintain, and effectively use the IoT system.

RELATED WORK-5:

A Survey on Automated Disease Diagnosis and Classification of Herbal Plants Using Digital Image Processing.

Authors: Ankit Khandelwal, Aashish Shukla, Mangal Sain

In their study, Ankit Khandelwal, Aasheesh Shukla, and Mangal Sain explore the application of digital image processing in the automated disease diagnosis and classification of herbal plants. Utilizing advanced computer vision algorithms, this approach significantly enhances the efficiency and accuracy of plant disease identification. The technology offers the potential for scalability and accessibility, making it a cost-effective solution that can be deployed across various locations. Moreover, the non-invasive nature of digital image processing ensures that plant health is not compromised during diagnosis. However, the system's reliance on high-quality images and its limited ability to generalize across different plant species are notable challenges. With a future scope focused on advancing segmentation techniques, the authors aim to refine the technology, thereby broadening its applicability and improving its robustness in herbal plant disease management.

Pros:

- Efficiency and Accuracy: Enables quick processing and precise disease identification in herbal plants.
- Accuracy: High precision in identifying and classifying diseases in herbal plants using advanced algorithms.
- Cost-Effective: Reduces the need for manual labor and lowers the costs associated with traditional diagnosis methods.

Cons:

- Data Dependency on Image Quality: The algorithms' effectiveness heavily relies on the quality of the input images.
- Limited Generalization: Algorithms might not perform well across different types of herbal plants without specific tuning.

2.2 Challenges

- Dependency on High-Quality Data: High-quality data is essential for training ML models, diagnosing plant diseases, and ensuring sensor accuracy, requiring reliable data acquisition systems.
- Technical Complexity and Resource Requirements: Implementing advanced technologies like CNNs, IoT, and machine learning demands substantial computational resources and technical expertise.

2.3 Problem Statement

In modern agriculture, the challenge is optimizing plant disease detection methods. This study addresses the need to combine the strengths of deep learning and classical machine learning for robust, accurate- time disease detection, contributing to early diagnosis and sustainable farming practices.

2.4 Summary

The reviewed research highlights advancements in plant disease detection using technologies like CNNs and IoT, each providing valuable insights for

agricultural improvements. Despite their potential, challenges such as the need for high-quality data, technical complexity, and limited generalization underscore areas for further enhancement. Future efforts should focus on refining data analysis, strengthening model robustness, and improving user interfaces to broaden these technologies' applicability and effectiveness in global agricultural practices. This research is essential for advancing sustainable and technologically integrated crop management and food security approaches.

2.5. Existing System

A. Traditional CNN-based approach

In the project [2] Pavan Kumar V, E Gurumohan Rao, G Anitha, G Kiran Kumar implemented an Image Classification model with Deep Learning and classic/traditional Convolutional Neural Networks (fully-connected from scratch) using Tensorflow. This model supported standard metrics and gave decent accuracy score on the tested datasets but it held few drawbacks like involving more computational resources and high processing time, as the whole model had to be trained on all data points, absence of transfer learning is main drawback.

B. Inception v3-based approach

In the project [1], Nikhitha M, Roopa Sri S, Uma Maheswari B used TensorFlow to implement an inception v3 based model approach for classifying plant diseases. This model is based on the concept of transfer learning, which involves building a base model on top of a pre-trained CNN architecture (Inception v3), resulting in faster pre-processing, low computational resources, processing time, and training time, resulting in an efficient model.

2.6. Limitation of Existing System

- Computational Requirements
- Training Data Size

- Overfitting
- Limited Contextual Information
- Generalization to New Diseases
- No scope for better performance on imagenet weights (due to Inceptionv3)
- Older version of TensorFlow

2.7. Proposed System

In order to find and record the plant diseases detection for scientific research and biological advancements. The proposed system suggests an approach based on the concept of transfer learning to overcome existing limits and achieve minimal pre-processing, compute time, resulting in improved disease identification accuracy. The **MobileNetV2 Architecture**, which has been pre-trained on imagenet weights and outperforms the other architectures on image net weights, will be used to implement the transfer learning principle.

In our work, we have used model known as **MobileNetV2**.It is a 53 layers deep CNN (Convolutional Neural Network) and takes input images of size 128x128 pixels. TensorFlow hub library was used to implement this model.

CHAPTER 3
METHODOLOGY

3.1. Dataset

The dataset contains leaves that are both diseased and disease-free. We have collected the dataset from Kaggle. The total number of images is 11000 which have been affected by different biotic stresses. This dataset was split into 2 parts that are- the train set and the val set. The training set is the portion of the dataset that is used to train or teach the machine learning model. The training set contains of 10000 images which is 75% of the dataset taken.The Validation set is the dataset that is used to evaluate the performance of the trained model. The Validation set contains of 1000 images which is 25% of the dataset taken.

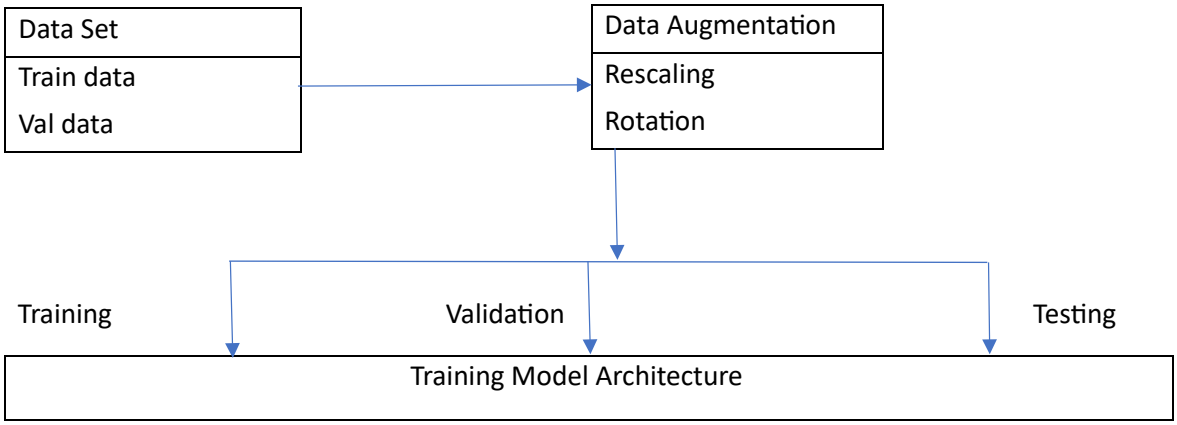
3.2 System Architecture

The technologies that have been implemented in this model include different Python Libraries, using Google Colaboratory, running over GPU/TPU runtimes. Our primary Libraries are going to be TensorFlow, Keras, Matplotlib, pandas, NumPy, SciKit Learn, os, Files, and io. The MobileNetV2 Architecture is considered our base model because it outperforms other architectures like inception v3 and VGG 16 and other architectures over the images. MobileNetV2 is a transfer learning architecture designed for mobile and embedded devices with limited computational resources. It is a lightweight convolutional neural network (CNN) that achieves a good balance between accuracy and efficiency.

MobileNetV2 consists of depth wise separable convolutions, which decompose the standard convolution operation into two separate layers: depth wise convolution and pointwise convolution. The depth wise convolution applies a single filter to each input channel independently, reducing the computational cost. The pointwise convolution then performs a 1x1 convolution to combine the output channels from the depth wise convolution.

The architecture also incorporates residual connections, similar to the ones used in Reset, to enable information flow across different layers without vanishing gradients. These connections help in training deeper networks without sacrificing performance.

MobileNetV2 introduces a new component called "inverted residuals" that further improves efficiency. Inverted residuals utilize bottleneck blocks with a bottleneck layer that reduces the number of input channels before applying depth wise separable convolutions. This reduces the computational burden while preserving important features. To adapt MobileNetV2 for transfer learning, the architecture can be used as a feature extractor. The pretrained MobileNetV2 model, trained on a large dataset like ImageNet, can be fine-tuned on a target dataset with relatively few labeled samples. By freezing the initial layers and training only the last few layers, the model can be specialized for a specific task, such as image classification or object detection, while leveraging the learned features from the pre-trained model. The system architecture of the leaf image classification in plant disease project is built around a convolutional neural network (CNN) model. The architecture consists of several layers: input layer, convolutional layers, pooling layers, fully connected layers, and an output layer. Each layer is designed progressively derive higher-level features from the raw input images. The model receives preprocessed image data, which passes through each layer to result in a classification output.



Transfer Learning MobileNetV2	Flatten Layer	Dense Layer with Relu Activation	Dense Layer with Relu Activation	Dropout Layer with Dropout 0.3	Dense Layer with SoftMax Activation
Test Accuracy					

Fig.3.1. MobileNetV2 Model Architecture

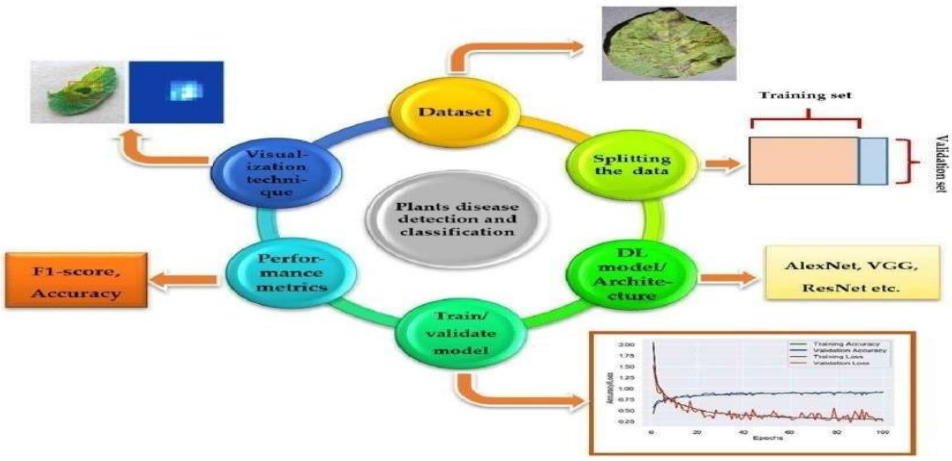


Fig.3.2. CNN Architecture

3.3 UML Diagrams

UML is Object oriented software engineering. Managed by the Object Management Group, it aims to be a common language for creating software models. UML consists of two major components: a Meta-model and a notation. It serves as a standard for specifying, visualizing, constructing, and documenting software artifacts, as well as for business modeling.

Goals

The Primary goals in the design of the UML are as follows:

- Offer users a readily accessible, expressive visual modeling language to facilitate the development and exchange of meaningful models.
- Provide mechanisms for extendibility and specialization to enhance the core concepts.

3.3.1. Use Case Diagram:

A use case diagram in the Unified Modeling Language (UML) represents a behavioral diagram derived from a Use-case analysis. The primary goal of a use case diagram is to illustrate the system functions performed for each actor, showcasing the roles of actors within the system.

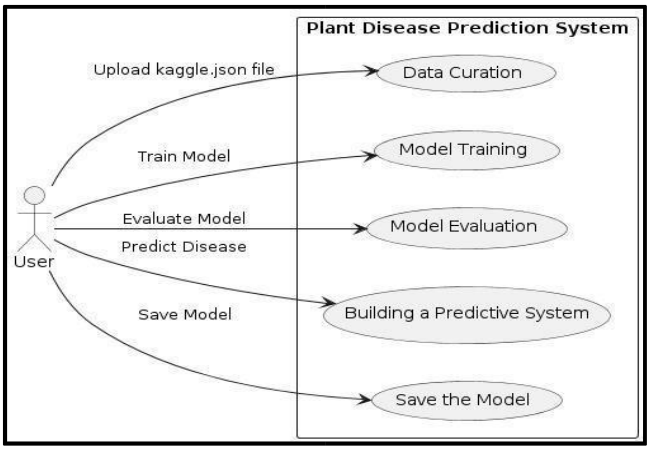


Fig:3.3 Use Case Diagram

3.3.2. Class Diagram:

In the realm of software engineering, a class diagram in the Unified Modelling Language (UML) serves as a static structure diagram depicting the structure of a system.

It presents the system's classes, along with their attributes, operations.

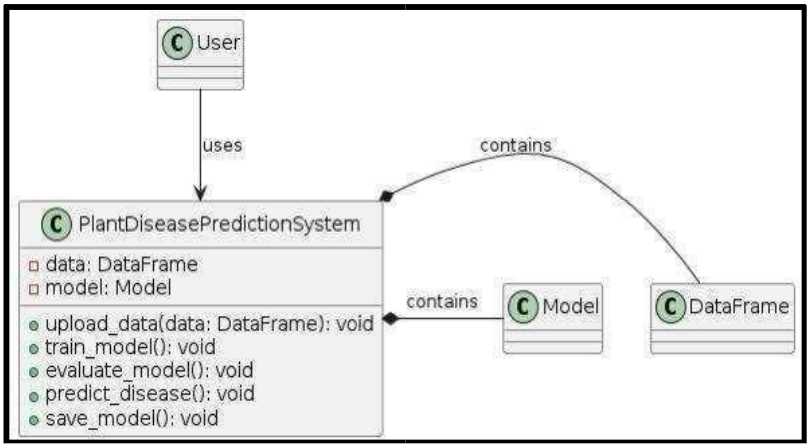


Fig:3.4 Class diagram

3.3.3. Sequence Diagram:

A sequence diagram within the Unified Modeling Language (UML) is an interaction diagram delineating how processes interact with each other and in what sequence.

It serves as a visual representation of a Message Sequence.

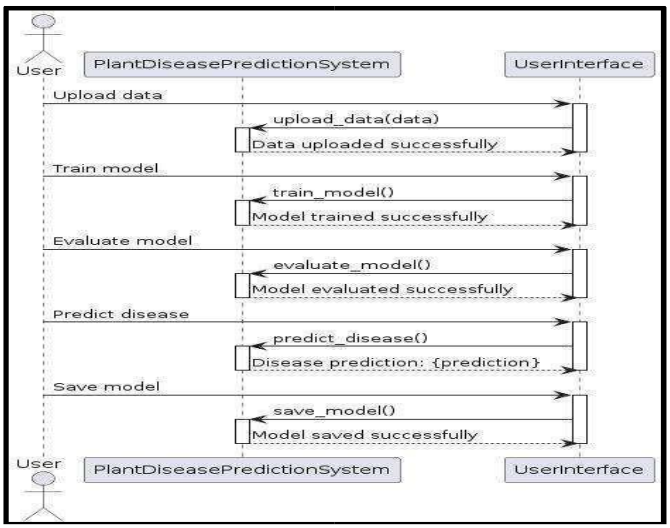


Fig:3.5 Sequence diagram

3.3.4. Activity Diagram:

Activity diagrams offer graphical representations of stepwise work flows, portraying actions and activities with support for choice, iteration, and concurrency.

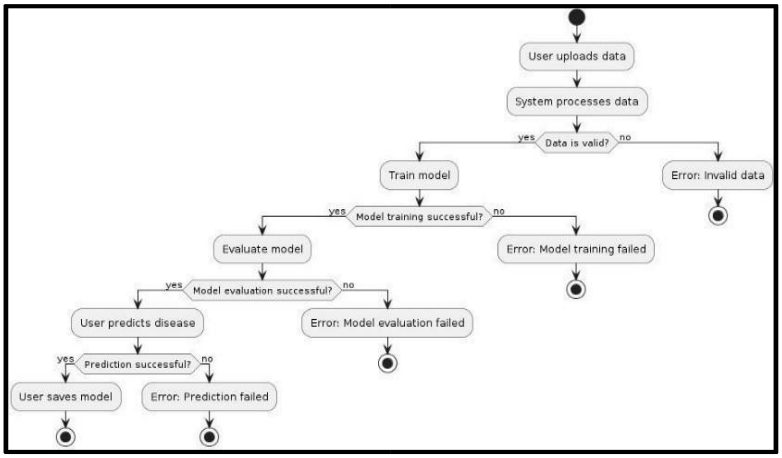


Fig:3.6 Activity diagram

3.4 Working Principles:

3.4.1 Modules:

- **Image Processing:**

  It involves various tasks such as resizing, normalization, noise removal, and enhancing image quality to prepare them for further analysis. Techniques like edge detection, image segmentation, and colour space conversion may also be applied to extract relevant information from the images. This module ensures that the input images are appropriately pre- processed to improve the accuracy of disease detection algorithms.

- **Feature Extraction:**

  Feature extraction module focuses on identifying extracting meaningful features from pre-processed images. It involves analyzing the texture, shape, and other visual characteristics of the images to represent them in a more compact and informative manner. The objective of this module is to convert unprocessed image data into a feature vector suitable for utilization in machine learning algorithms.

- **Classification:**

The classification module is responsible for categorizing the pre-processed and feature-extracted images into different disease classes. It utilizes machine learning algorithms such as support vector machines (SVM), k-nearest neighbours (k-NN), or deep learning models like CNNs for accurate classification. The module trains the classifier using labeled data from the dataset and then predicts the disease class of unseen images based on their extracted features.

- **User Interface:**

The user interface module provides an interactive platform for users to interact with the system. It includes functionalities for uploading input images, selecting disease detection options, and displaying the results of classification. Graphical components like buttons, text boxes, and image displays are utilized to enrich user interaction and simplify navigation. The System Requirements Specification (SRS), also referred to as Software Requirements Specification, comprises a document or collection of documents outlining the functionalities and operations of a system or software application.

CHAPTER 4

TOOL DESCRIPTION

This section gives a detailed description about the hardware tools and software tools involved in developing this system.

4.1. Hardware Requirements

- Intel core i5 10th Gen Processor.
- Supported GPU/CPU/TPU Resources provision.
- 16 GB RAM.
- Windows 11 OS.
- 1 TB Storage.
- Intel(R) Core (TM) i5-10210U CPU @ 1.60GHz, 2101 Mhz, 4 Core(s), 8 Logical Processor(s).

4.2. Software Requirements

- Python version - 3.7.13.
- Jupyter Notebook / Google Colaboratory / Python IDE / Code Editors.
- TensorFlow version – 2.8.2
- Keras version 2.8.0
- Python Libraries used – TensorFlow, Keras, Matplotlib, pandas, numpy, SciKit Learn, skimage, os, Files, io.
- Convolutional Neural Networks, Xception Architecture (Imagenet weights).

CHAPTER 5

IMPLEMENTATION

5.1. ALGORITHM STEPS:

- **Importing all necessary Libraries:**

  TensorFlow is the main Prerequisite, followed by other libraries like os for operating system integration, Keras for implementation of deep neural networks, Matplotlib for data visualization, pandas and numpy for data pre-processing and analysis, SciKit Learn for ML/DL validation, skimage for image pre-processing, Files for accessing filesystem, io for dealing with various types of I/O.

- **Data Collection/Data Labelling and Pre-processing:**

  Here, data is primarily acquired from verified public repositories and is pre-processed using multiple libraries, after these images have been properly labelled. Pre-processing involves resizing/rescaling and other methods.

- **Data Generator (Load input):**

  Data is loaded into the Colaboratory from local file system/public repository using the keras/TensorFlow.

- **Data Augmentation:**

  The Image DataGenerator method has been to generate/augment more data samples from the existing data points, by augmenting them. (Rotating, zooming, and so on)

- **Call backs:**

  The tensorflow, keras. callbacks module is used in order to implement early stopping and checkpoint methods, to avoid overfitting of the model.

- **Implementing the Base Model (MobileNetV2):**

  The MobileNetV2 on keras library is used, by giving the imagenet dataset as weights and specifying size and dimensions of the image along with their batch size.

- **Build Head Model (CNN):**

Addition of extra head layers is performed, Average pooling, flattening, dense and dropout are the multiple head layers added to the model. Here, we use the relu and softmax activation functions.

- **Compile Model:**

  Here, the entire model is compiled by integrating the base and head model layers using model. compile method and specify the loss function and accuracy metrics.

- **Running model cycles (First and Second):**

  The model is run in 2 cycles, by varying the number of epochs in each run for improvising accuracy score and reducing loss function value, by checking for the best parameter.

- **Plotting Accuracy and Loss curves:**

  For each run, the accuracy and loss curves of each epoch have been plotted for both training and testing.

  This is to compare and check which is the best run case.

- **Loading Best Model Weights:**

  After both the runs, the optimal model's weights are chosen and use them as the final output parameters.

- **Model prediction on Test set:**

  The test data is used from the Test Data Generator method, to validate the model performance on new images.

- **Process and store Results file:**

  The final output results, in the form of probabilities for each class, are processed and stored in a proper tabular format along with the best model weights.

5.2. Sample Code:

Plant_Disease_Detection.pynb

#Seeding for reproducibility import random random. seed (0)
import numpy as np np. random. seed (0) import TensorFlow
as tf tf.random.set_seed (0)

##Importing the dependencies

imprt os imprt json
from zipfile import ZipFilefrom PIL

import matplotlib. pyplot as plt import matplotlib.image as mpimg

from tensor flow. keras. preprocessing. Image import

ImageDataGeneratorfromtensorflow.keras import layers, models

##Data Curation

##Upload the kaggle.json file

! pip install Kaggle

kaggle_credentails = Json. load (open ("Kaggle. Json")) # setup
Kaggle API key as environment variables os.
environ['KAGGLE_USERNAME'] =kaggle_credentails["username]
os. environ['KAGGLE_KEY'] = kaggle_credentails["key"]

! Kaggle datasets download -d abdallahalidev/plantvillage-dataset! ls

 # Image Data Generators

data_gen = ImageDataGenerator
(rescale=1./255, validation split=0.2 #
Use 20% of data for validation)

 # Train Generator

train_generator = data_gen. flow_from_directory(base_dir,

 # Training the Model

history=model.fit(train_ge
nerator,

steps_per_epoch=train_generator.samples // batch_size,
 # Number of steps perepoch epochs=5,
 # Number of epochs
validation_data=validation_generator,

validation_steps=validation_generator.samples // batch_size #
Validation steps)

 # Function to Predict the Class of an Image

def predict_image_class(model, image_path, class_indices):
preprocessed_img = load_and_preprocess_image(image_path)
predictions

model.predict(preprocessed_img)

predicted_class_index = np.argmax(predictions, axis=1)[0] predicted_class_name
= class_indices[predicted_class_index] return predicted_class_name

 # Create a mapping from class indices to class names

class_indices = {v: k for k, v in
train_generator.class_indices.items()}

 # saving the class names as json file

```
json.dump(class_indices, open('class_indices.json',
'w'))# Example Usage imagea_path
'/content/1.JPG'imageb_path  = '/content/2.jpg'
imagec_path='/content/3.JPG'
imaged_path='/content/4.JPG'
imagef_path='/content/5.JPG'

predicta_class_name = predict_image_class(model,imagea_path,class_indices)

predict_image_class(model,imageb_path,class_indices)
predict_image_class(model,imagec_path,class_indices)
```

 # Output the result

```
print("Predicted Class Name:", predicta_class_name) print("Predicted Class

Name:", predictb_class_name)print("Predicted Class
     Name:",predict_class_name)

print("Predicted Class Name:", predicted_class_name)
```

5.3. Main Code

```
import os

import json

from PIL import Image

import numpy as np

import tensorflow as tf

import streamlit as st

working_dir = os.path.dirname(os.path.abspath(_file_))

model_path = f"{working_dir}/trained_model/plant_disease_prediction_model.h5"

# Load the pre-trained model

model = tf.keras.models.load_model(model_path)
```

```
 # loading the class names
 class_indices = json.load(open(f"{working_dir}/class_indices.json"))


# Function to Load and Preprocess the Image using Pillow
def load_and_preprocess_image(image_path, target_size=(224, 224)):
# Load the image
img = Image.open(image_path)
 # Resize the image
 img = img.resize(target_size)
 # Convert the image to a numpy array
 img_array = np.array(img)
  # Add batch dimension
img_array = np.expand_dims(img_array, axis=0)
 # Scale the image values to [0, 1]
img_array = img_array.astype('float32') / 255.
 return img_array


# Function to Predict the Class of an Image
def predict_image_class(model, image_path, class_indices):
preprocessed_img = load_and_preprocess_image(image_path)
predictions = model.predict(preprocessed_img)
predicted_class_index = np.argmax(predictions, axis=1)[0]
predicted_class_name = class_indices[str(predicted_class_index)]
return predicted_class_name
# Streamlit App
st.title('Plant Disease Classifier')
```

```
uploaded_image = st.file_uploader("Upload an image...", type=["jpg", "jpeg", "png"])

if uploaded_image is not None:
image = Image.open(uploaded_image)
col1, col2 = st.columns(2)

with col1:
 resized_img = image.resize((150, 150))
st.image(resized_img)

with col2:
if st.button('Classify'):
# Preprocess the uploaded image and predict the class
prediction = predict_image_class(model, uploaded_image, class_indices)
st.success(f'Prediction: {str(prediction)}')
```

CHAPTER 6

TESTING

6.1 Need of Testing

Testing aims to uncover errors by exercising software to ensure it meets requirements and user expectations. It involves checking the functionality of components, assemblies, and finished products to prevent unacceptable failures. Different test types address specific testing needs.

6.2 Types of Testing

6.2.1 Unit Testing:

Unit testing involves designing test cases to validate internal program logic and ensure that inputs produce valid outputs. It's done at the individual unit level after completion and before integration, focusing on specific business processes or system configurations. These tests validate each unique path of a process against documented specifications, ensuring accuracy and precise expected results.

6.2.2 Integration Testing:

Integration tests validate the functionality of integrated software components, ensuring they operate collectively as a single program. They focus on event-driven testing and the overall outcome of screens or fields. Integration tests verify that while individual components passed unit testing, their combination is correct and consistent, uncovering issues that may arise from component interactions.

6.2.3 Functional Testing:

Functional tests systematically demonstrate that functions tested meet specified business and technical requirements, system documentation, and user manuals.

They focus on valid and invalid input classes, exercising identified functions, and exercising classes of application outputs. Functional testing also involves invoking interfacing systems or procedures. The organization and preparation of functional tests prioritize.

6.3. Test Strategy And Approach:

Field testing will be executed manually, while functional tests will be meticulously scripted.

6.4. Test objectives:

- Ensure the functionality of all field entries.
- Validate the activation of pages from designated links.
- Verify timely response of entry screens and messages.

6.5. Features To Be Tested:

- Validate the format correctness of entries.
- Prohibit duplicate entries.
- Confirm all links redirect users to the appropriate pages.

6.6. Screenshots:

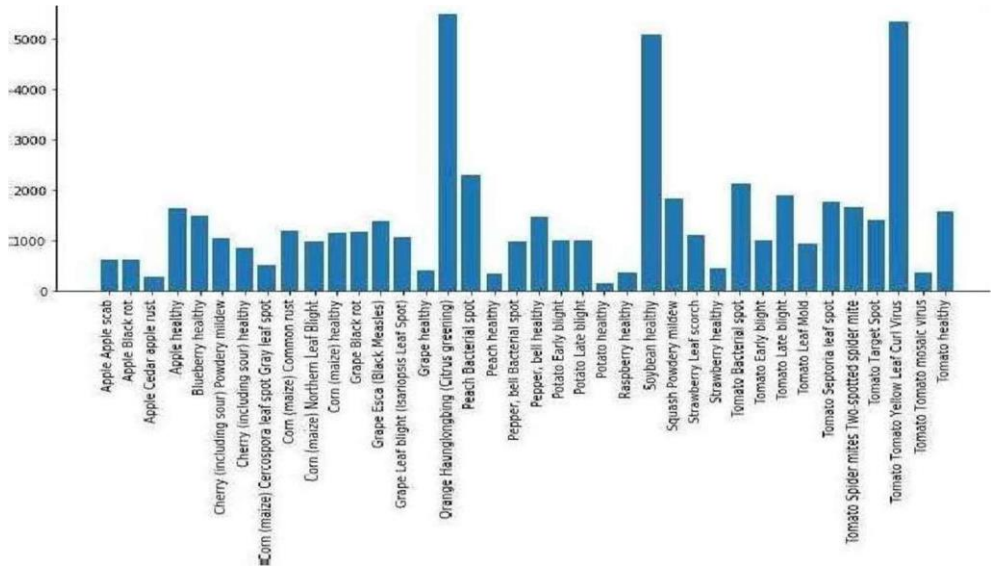
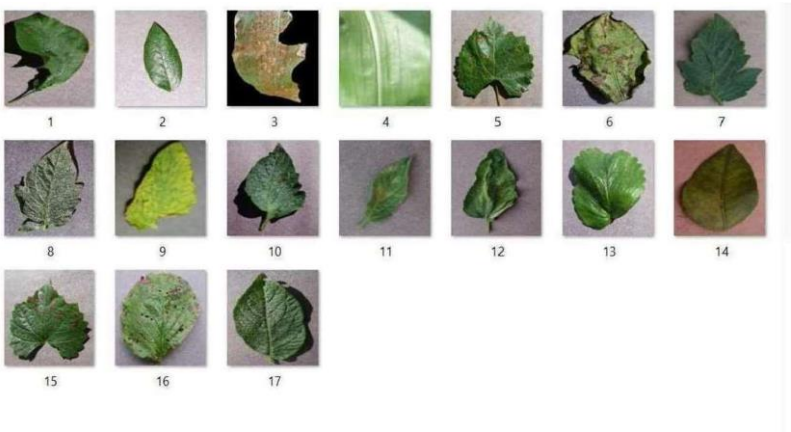
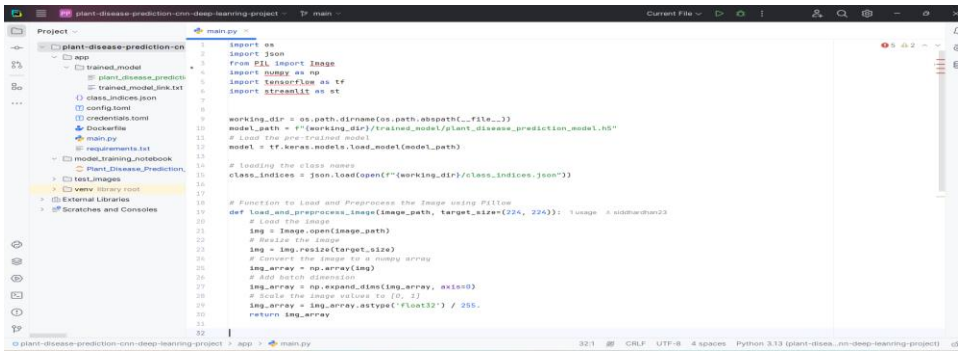


Fig 6.3.1 Sample input of data classes

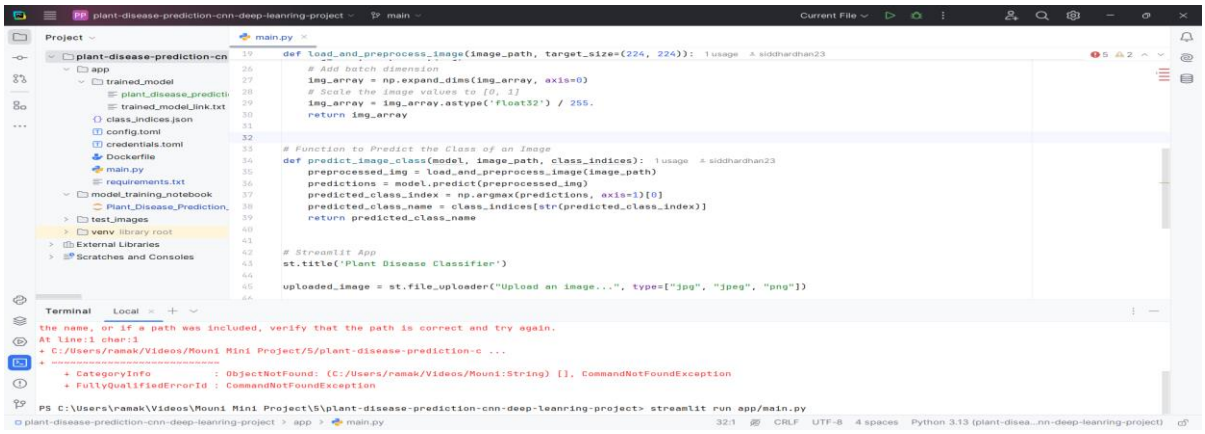


6.3.2 Loading images for testing



```
import os
import json
from PIL import Image
import numpy as np
import tensorflow as tf
import streamlit as st


working_dir = os.path.dirname(os.path.abspath(__file__))
model_path = f"{working_dir}/trained_model/plant_disease_prediction_model.h5"
# Load the pre-trained model
model = tf.keras.models.load_model(model_path)

# loading the class names
class_indices = json.load(open(f"{working_dir}/class_indices.json"))


# Function to Load and Preprocess the Image using Pillow
def load_and_preprocess_image(image_path, target_size=(224, 224)):
    # Load the image
    img = Image.open(image_path)
    # Resize the image
    img = img.resize(target_size)
    # Convert the image to a numpy array
    img_array = np.array(img)
    # Add batch dimension
    img_array = np.expand_dims(img_array, axis=0)
    # Scale the image values to [0, 1]
    img_array = img_array.astype('float32') / 255.
    return img_array

```

Fig6.3.3 Code in PyCharm



```
def load_and_preprocess_image(image_path, target_size=(224, 224)):
    # Add batch dimension
    img_array = np.expand_dims(img_array, axis=0)
    # Scale the image values to [0, 1]
    img_array = img_array.astype('float32') / 255.
    return img_array

# Function to Predict the Class of an Image
def predict_image_class(model, image_path, class_indices):
    preprocessed_img = load_and_preprocess_image(image_path)
    predictions = model.predict(preprocessed_img)
    predicted_class_index = np.argmax(predictions, axis=1)[0]
    predicted_class_name = class_indices[str(predicted_class_index)]
    return predicted_class_name


# Streamlit App
st.title('Plant Disease Classifier')

uploaded_image = st.file_uploader("Upload an image...", type=["jpg", "jpeg", "png"])
```

```
the name, or if a path was included, verify that the path is correct and try again.
At line:1 char:1
+ C:/Users/ramak/Videos/Moun1 Mini Project/5/plant-disease-prediction-c ...
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (C:/Users/ramak/Videos/Moun1:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\ramak\Videos\Moun1 Mini Project\5\plant-disease-prediction-cnn-deep-leanring-project> streamlit run app/main.py
```

Fig6.3.4 Running code in PyCharm

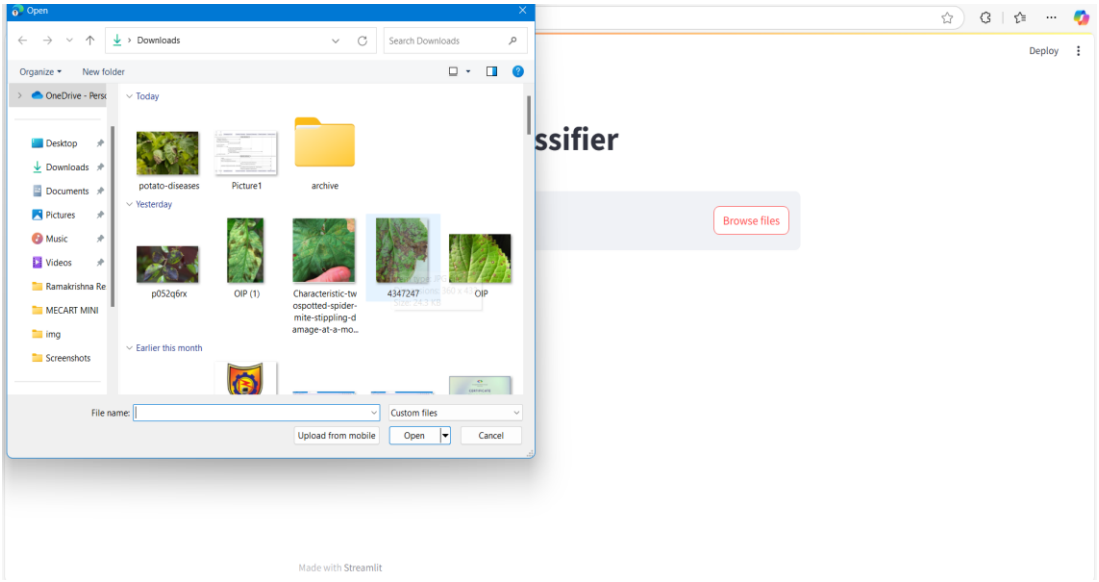


Fig6.3.5 Browsing Plant Leaf Images

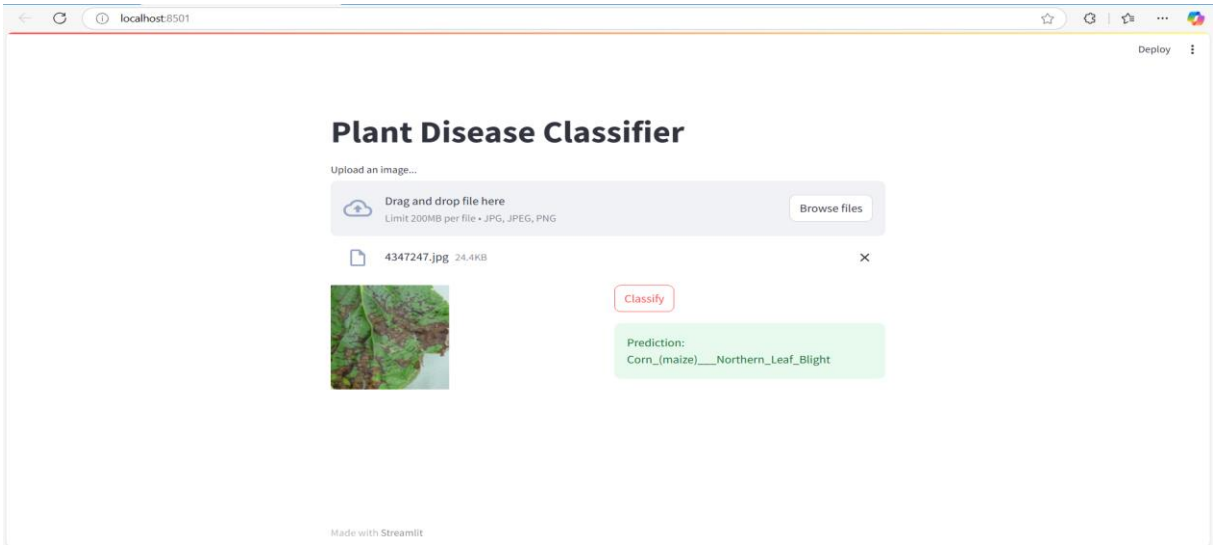


Fig6.3.6 Prediction of the Plant Leaf

CHAPTER 7

RESULTS AND ANALYSIS

This section is containing a description of the main findings of this proposed system with the figures, a detailed explanation by comparing with previous studies and analysis.

7.1. Result Discussion

Upon running 5 epochs in the first run and 10 epochs in the second run, the obtained loss:

0.1387 - accuracy: 0.9527 as our optimal loss function and accuracy values.

In the final Results file, the obtained classification results which indicate classification probabilities of each image, for all the tomato (classes), the final probability of each case sums up to one. Here, the high computational speeds achieved is 8s per step which is efficient in comparison to previous models.

7.2. Comparison with Previous Studies

In the inception model discussed in previous studies, the loss function curves between training - testing validation sets and epochs are not satisfactory, on the other hand the MobileNetV2 architecture which has been implemented holds higher satisfactory curve integration between the training – validation set and the number of epochs run.

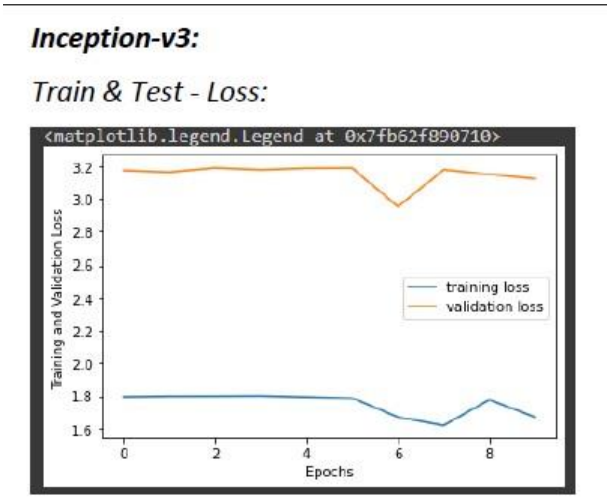


Fig.7.2.1. comparison plot of training and validation loss vs epochs using line plot (Inception-v3)

Here, the integration between the training loss and validation loss is very low making the model less trust worthy.

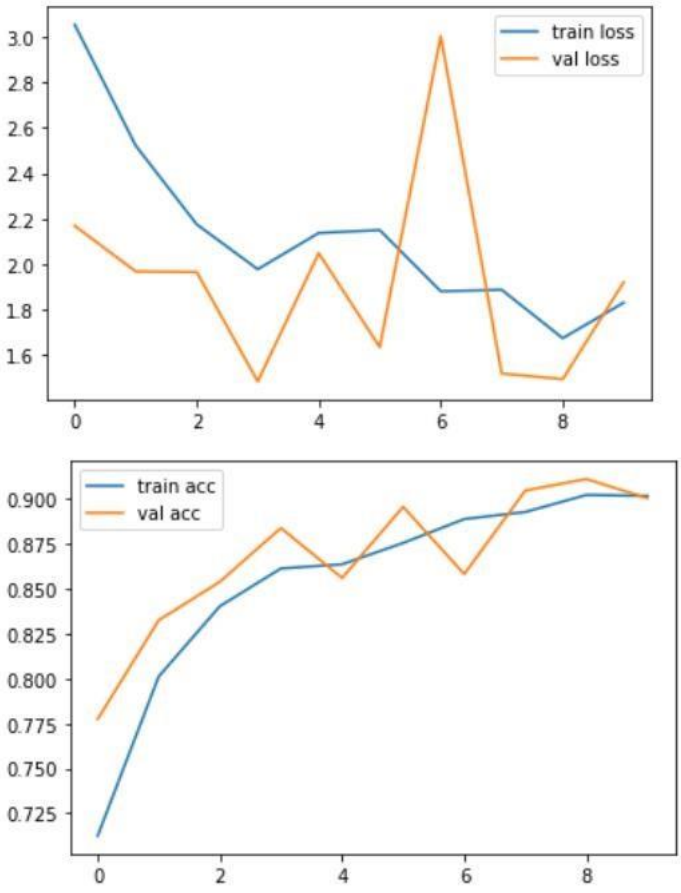


Fig.7.2.2. comparison plot of training and validation loss vs epochs using line graph (Inception V3)

In above graph, the curves are observed to be integrated, meaning the training and validation loss are in same stream as the number of epochs keep varying.

7.3. Analysis

The MobileNetV2 architecture model avoids overfitting by using check point call-back method and also decreases computational time and pre-processing by applying transfer learning concept. Here the lower loss function and higher accuracy in comparison to other architectures like inception v3 and VGG 16, are observed. This model is highly preferred for these reasons for the purpose of disease identification.

CHAPTER 8

CONCLUSIONS AND FUTURE SCOPE

In conclusion, the use of MobileNetV2 for plant disease detection offers several advantages. Its lightweight architecture and efficient computational performance make it suitable for deployment on mobile devices, enabling real-time disease detection in the field.

MobileNetV2 achieves a balance between accuracy and efficiency, making it well-suited for resource-constrained environments. By leveraging transfer learning, the model can be trained on limited annotated data, reducing the need for extensive labeled datasets. Additionally, MobileNetV2 can generalize well to various plant species, making it versatile for detecting diseases across different crops. Overall, the integration of MobileNetV2 in plant disease detection systems holds great promise for early detection, accurate diagnosis, and effective management of plant diseases, ultimately leading to improved crop yields and agricultural sustainability.

REFERENCES

[1] Sunil S. Harakannanavara, Jayashri M. Rudagi, Veena I. Puranikmath, Ayesha Siddiqua, R. Pramodhini, Plant leaf disease detection using computer vision and machine learning algorithms" *Global Transitions Proceedings* 3 (1) (2022) 305–310.

[2] R. Sujatha, J.M. Chatterjee, N. Jhanjhi, S.N. Brohi, Performance of Deep learning vs machine learning in plant leaf disease detection, Microprocess. Microsystem. 80 (2021), 103615

[3] C.K. Sunil, C.D. Jaidhar, N. Patil, Cardamom plant disease detection approach using EfficientNetV2, in: IEEE Access, vol. 10, 2021, pp. 789–804, https://doi.org/10.1109/ACCESS.2021.3138920, 2021.

[4] M. Bhagat, D. Kumar, I. Haque, H. S. Munda and R. Bhagat, "Plant Leaf Disease Classification Using Grid Search Based SVM," 2nd International Conference on Data, Engineering and Applications (IDEA), Bhopal, India, 2020, pp. 1-6, doi: 10.1109/IDEA49133.2020.9170725.

[5] Ghaiwat, Savita N., and Parul Arora. "Detection and classification of plant leaf diseases using image processing techniques: a review." *International Journal of Recent Advances in Engineering & Technology* 2.3 (2014): 1-7.

[6] Kulkarni, Anand H., and Ashwin Patil. "Applying image processing technique to detect plant diseases." *International Journal of Modern Engineering Research* 2.5 (2012): 3661-3664.

[7] Mohanty, Sharada P., David P. Hughes, and Marcel Salathé published the article titled "Using deep learning for image-based plant disease detection" in Frontiers in Plant Science, volume

[8] Fujita, E., et al. "A practical plant diagnosis system for field leaf images and feature visualization." International Journal of Engineering & Technology 7.4.11 (2018): 49-54.

[9] Haralick, Robert M., Karthikeyan Shanmugam, and Its' Hak Dinstein published the paper titled "Textural features for image classification" in the IEEE Transactions on Systems, Man, and Cybernetics, volume 6, in 1973, spanning pages 610-621.

[10] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297. C. Dicle, M. Sznaier, and O. Camps.