B. Tech Project Report

On

Crop Disease Detection using Deep Learning CNN

Submitted in the partial fulfilment

of award of

B.Tech Degree in Electrical Engineering

by

Ravina

Roll no. 12114163

Under the Mentorship of

Dr. Shelza Dua

Under the Supervision of

Dr. Lillie Dewan

Department of Electrical Engineering

National Institute of Technology, Kurukshetra

Haryana-136119, India

Jan-May 2024

DECLARATION

I hereby declare that the work which is being presented in the Project **"Crop Disease Detection using DEEP CCN"**, in partial fulfilment of the requirements for Internship project is an authentic record of my own work carried for the evaluation of the Sixth semester of **Bachelor of Technology in Electrical Engineering**, in National Institute of Technology, under the mentorship of Professor**, Dr Lillie Dewan** and Research Associate, **Dr Shelza Dua** in Electrical Engineering Department. The matter presented in this internship report has not been submitted for the award of any other degree elsewhere.

Signature of candidate

Ravina

12114163

Date: 05.04.2024

It is to certify that Ravina roll no. 12114163 has done project under our guidance during the period of January, 2024 to May, 2024. The statement made above is true to best of our knowledge.

Dr. Shelza Dua

Lab Mentor

Dr. Lillie Dewan

Faculty Member

Table of Contents

4. Result

Chapter 1

INTRODUCTION

1.1 OVERVIEW

Crop diseases are a significant problem for agriculture, leading to major crop losses and reducing food security. It is very important to identify crop diseases rapidly and accurately to manage and prevent them effectively. Recently, technologies like computer vision and deep learning have appeared as promising tools for detecting crop diseases. Deep learning models, in particular, have shown great potential in accurately identifying diseases in crop images. However, given the wide variety of crop diseases and the complexity of crop images, it's important to choose which deep learning model is most effective for the task.

The next section provides an overview using deep learning models for crop disease detection. The methodology section details the approach, model structure, and training process. The results and discussion section shows the performance of each model. Finally, the conclusion summarizes the key values of the research and discusses potential future research directions in this field.

1.2 MOTIVATION

Latest technologies made human society able enough to produce enough food to meet the demand of more than we need. However, food security remains terrorised by a lot of factors like climate change. Crop diseases are dangerous to food security at the global level. So various efforts have been developed to prevent crop loss due to diseases. Crop disease detection is necessary for safeguarding food production, minimizing economic losses, and promoting a sustainable agriculture.

Timely identification helps farmers mitigate crop damage, optimize resource use, and protect the environment by reducing the need for harmful chemical interventions. It supports global food security efforts by ensuring stable yields and good quality produce. Additionally, research in disease detection advances agricultural science, leading to innovations in crop breeding and management practices. Ultimately,

the motivation for crop disease detection lies in its critical role in ensuring food security, preserving agricultural livelihoods, and fostering environmental sustainability for present and also for the future generations.

1.3 APPLICATIONS

The applications of crop disease detection are wide-range and impactful:

- **Early Disease Detection**: Fast and accurate detection of crop diseases allows for timely intervention, preventing the spread of pathogens and minimizing crop damage.
- **Remote Sensing**: Remote sensing techniques, which includes satellite imagery and drones, can be utilized for large-scale monitoring of crop health, facilitating early detection of diseases over extensive agricultural areas.
- **Precision Agriculture**: Disease detection technologies enable farmers to monitor the health of their crops with precision, allowing for targeted treatments and resource allocation.
- **Disease Management**: Information gathered from disease detection systems guides farmers in implementing effective disease management strategies, such as crop rotation, resistant variety selection, and judicious use of pesticides.
- **Smart Farming Systems**: Integration of disease detection sensors and systems into smart farming platforms enables real-time monitoring and data-driven decision-making, optimizing crop management practices.
- **Research and Development**: Disease detection technologies serve as invaluable tools for researchers studying crop-pathogen interactions, leading to the development of novel disease control methods and disease-resistant crop varieties.

Chapter 2

TECHNOLOGIES EXPLANATION

2.1 Software Requirement

2.1.1 JUPYTER NOTEBOOK

Jupyter Notebook Application is a server client-based application which allows editing and running code using web browser. This App integrates code and its output into a single document. It can be installed on a remote server and can be used using internet or it can be executed on a local desktop without any internet connection. It is an interactive environment in which you can combine mathematics and its various equations, plots and also multimedia content. In other words, Jupyter Notebook is a web application that can be used to create and share documents that contains codes, charts, mathematical equations, visualizations and text. Jupyter Notebook supports various languages like as Julia, Python and R. Jupyter ships with Python Kernels, which allows us to write your programs in Python. It is separately installed on our computer. First version of notebook for IPython was first released in 2011. Jupyter Notebook was offshoot project from IPython. Visual Studio or VS code also supports jupyter notebook locally on your computer.

Advantages of Jupyter Notebook

1.User-friendly: Jupyter Notebook is easy to use interface that makes it user friendly to write code, execute and modify code. It supports many languages. It is versatile for various data analysis and scientific computing tasks.

2. Integration with other tools: Jupyter Notebook allows you to work with different libraries like NumPy, matplotlib which are helpful in machine learning, data analysis and training models.

3. Flexibility of Notebook: Jupyter Notebook allows you to divide code into small division which are modular and reusable cells. This helps to run code into independently without affecting entire document. Thid helps to find error in specific part of code. You can also modify specific part of code independently.

4. Large community and Ecosystem: Jupyter Notebook supports has a vast ecosystem of developers which helps new developers who are starting their journey. It helps them if they find difficulty while using it.

5. Immersive media: Jupyter Notebook allows you to output rich media, videos, picture, charts and graphs using other libraries.

2.1.2 GOOGLE COLABORATORY

Google Colaboratory or Google Colab is a cloud based integrated development environment (IDE) provided by google. It allows users to write, execute and share python colaboratory using web browser. It is a product from google research. It is a jupyter notebook environment which works completely in the cloud. In other words, collab allows anybody to write python code and run it.

It is free to use for everyone. One of the most important features of its that it does not require any setup and your team member an edit the notebooks created by you. It supports all the machine learning libraries. Most of the libraries used are already installed. Another important service provided by google colaboratory is that the use of GPUs and TPUs. It allows developers the use of GPU freely, which is helpful for

students because machine with GPU is expensive. This can also be linked with google drive. Google Collaboratory offers an accessible and useful platform for coding and experimentation, especially to analyse data, machine learning and research related projects. Its cloud-based nature and collaborative features make it a popular choice between students, researchers and developers looking for a free coding environment.

Advantages of using Google Colaboratory:

1. Free Accessibility: Google colab offers free access to a GPU and high-performance computing environment. It is useful for tasks that require high computational power like training deep learning models and processing very large datasets.

2. Integrated with google drive: This can be seamlessly integrating with google drive, allowing you to easily save and load notebooks and datasets from google drive. This ensures easy access to your files.

3.Pre installed libraries: Google colab contains many pre-installed popular libraries of python such as TensorFlow, NumPy, pandas, matplotlib, pytorch which are required for data analysis and machine learning. This is a time saver and also save effort in setting up environment and installing dependencies.

2.1.3 BROWSER

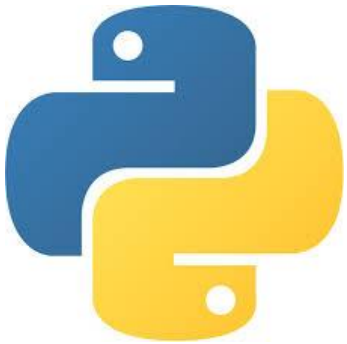
A Browser (also called a web browser) is a software usefulness for gaining access to facts on the arena huge net. Every individual internet web page, photo and video is recognize by using a wonderful uniform useful resource locator (URL), permitting browsers to retrieve those sources from an internet server and display them at consumer's device. A browser is not a identical thing as search engine, though the two are regularly burdened.

For users, search engine is only a website, consisting of google.com, that stores searchable facts approximately all different websites. However, for connecting with a website's server and show its internet pages, a user desires to have an internet browser set up on their tool. There are some most popular browsers such as Chrome, Safari, Firefox , Explorer and edge.

2.2 Language And Library Requirements

2.2.1 PYTHON

Python is a high-level, interpreted programming language known for its simplicity and readability. Created by Guido van Rossum in 1991, it has become one of the most popular languages for various applications. It is easy to learn and use. It has extensive standard library and third-party packages Cross-platform compatibility Dynamic typing and also an automatic memory management Support for multiple programming models.



It mainly focuses attention on code readability and its code is generally of small size or very less lines comparatively to the other programming languages. It is designed to increase readability. It uses English keywords mostly while other languages use punctuation. It also has fewer syntactical construction than other languages. There are many uses of Python in programming field and also in data science and analysis of data. Some of the many uses of Python are application development, fully constructed

programming library, implementation of automation testing process, database system accessibility, machine learning, data analysis, scientific computing, artificial intelligence. Python supports multiple programming models, functional programming, including object oriented, allows developers to choose the most appropriate according to their respective projects.

Advantages of Python:

Python has many **advantages** which have contributed to its widespread popularity between developers. Some of the key advantages are given below:

1.Easy to use and read: Python's simplicity and readability makes it a magnificent language for beginners. Its easy syntax allows new programmers to grasp fast and start writing useful code.

2. Cross-Platform Compatibility: Python is available on many platforms like Windows, macOS, Linux and many more. This feature allows programmers to write code on any system and run it on any other.

3. Huge Libraries and Ecosystem: Python has a huge community that has contributed to a loaded ecosystem of packages and libraries. Python package allows access to many pre built modules and libraries.

4. A Strong community: Python has a very strong and helpful community of developers and creators, which allows easy access to information and resources. Python's popularity has been growing steadily due to its features such as easy of use, compatibility with different operating systems, extensive resources. It's versatility and simplicity makes it an perfect choice for both beginners learning to code and experienced developers working on complex applications.

2.2.2 NUMPY

NumPy stands for "Numeric Python" and is a Python package used for working with arrays, both single-dimensional (like lists) and multi-dimensional (like matrices). It's an extension of Python that is mostly written in C, which makes it very fast for performing calculations.

NumPy stands for "Numeric Python" and is a Python package used for working with arrays, both single-dimensional (like lists) and multi-dimensional (like matrices). It's an extension of Python that is mostly written in C, which makes it very fast for performing calculations.

NumPy offers powerful data structures for handling arrays and matrices, making it ideal for tasks that involve a lot of number crunching. It's especially useful for efficiently managing large amounts of data and performing operations like matrix multiplication and reshaping data. Because of its speed, NumPy is a great choice when working with large datasets.

Advantages of NumPy:

1. It efficiently implements the multidimensional arrays.

2. NumPy performs array-oriented computing.

3. It performs scientific computations.

4. It is capable of performing Fourier Transforms and reforming data stored in muti-dimensional arrays.

5. It provides the inherent functions for linear algebra and random number generation.

2.2.3 OPEN CV

Computer vision shortly called as OpenCV is a rapidly advancing field that aims to enable computers and digital systems to derive meaningful information from visual inputs, such as images and videos. This interdisciplinary field combines expertise from areas like image processing, pattern recognition, machine learning, and artificial intelligence. At its core, its tasks involve the automated analysis and understanding of digital visual data. It involves a huge range of applications, from simple image classification to complex scene understanding and object detection. Computer vision algorithms can be trained to recognize specific faces, objects, text, or even entire environments, allowing for applications in fields like security, surveillance, autonomous vehicles, medical imaging, and robotics.

One of the key drivers behind the progress in computer vision has been the rise of deep learning, a subset of machine learning that utilizes artificial neural networks. Deep learning models, such as convolutional neural networks (CNNs), have demonstrated remarkable accuracy in tasks like image classification, object recognition, and semantic segmentation. These models can learn hierarchical visual representations from large datasets, enabling them to tackle increasingly challenging computer vision problems.

As the field continues to develop, researchers and developers are exploring new limits, such as 3D reconstruction, real-time video analysis, and multimodal perception. The ongoing advancements in hardware, algorithms, and available data are poised to drive further breakthroughs in computer vision, transforming the way we interact with and understand the visual world around us.

Various uses of computer vision are: –

- Write and Read Image
- Perform Feature detection
- Detect specific Images
- Capture and Save videos
- Process Images

2.2.4 Hardware Requirements

1. Laptop/Desktop.
2. 1.3 GHz or better recommended processor.
3. Hard disk space: From range 800MB to 210GB of available space.
4. 8 GB of RAM.
5. A Good Internet connection.

Chapter 3

METHODOLOGY

3.1 Propose Model

.

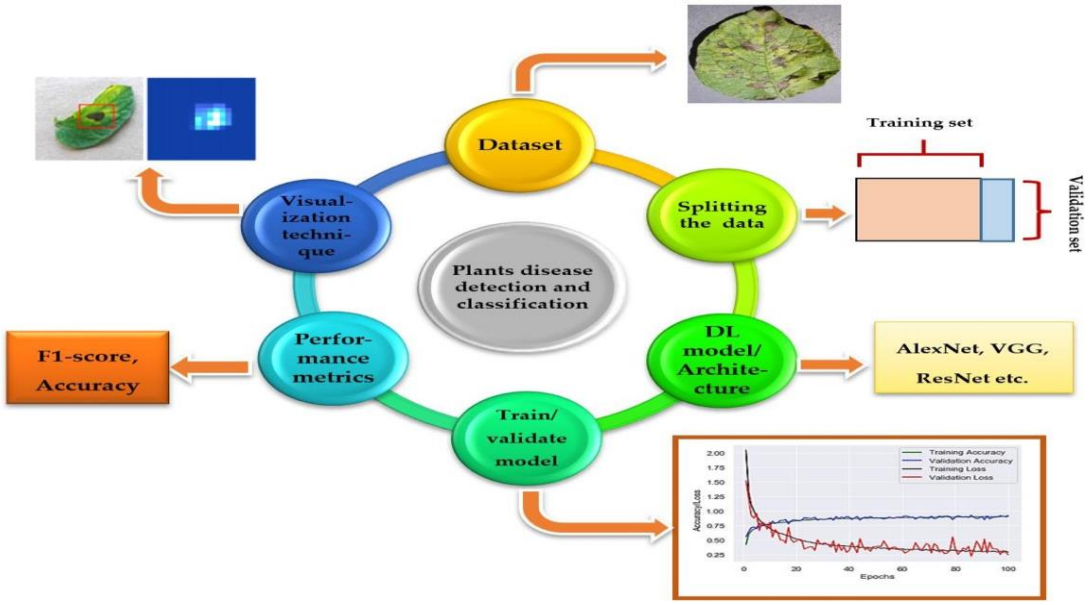


Fig. 3.1 Model Architecture

3.2 Model explanation

1.Data Collection

Dataset is the centre of deep learning-based researches. For boosting higher accuracy and precision, there should be a enough amount of data. Here we Gathered a variegated dataset of 8000 crop leaf images, which includes both healthy leaves and leaves affected by various diseases which are divided into 8 classes.

2. Splitting the Dataset

Divided the dataset into three categories named as training, validation, and testing sets. The training set is used to teach the model how to make predictions. The validation set helps fine-tune the model and check how well it's doing during training. Finally, the testing set is used to see how well the model performs after it has been fully trained.

3. Model Selection

- Choose the VGG19 model as the base architecture for disease detection.

- VGG19 is a deep convolutional neural network (CNN) architecture which contains 19 layers.

4.Transfer Learning

Fine-tune the pretrained VGG-19 model on the crop leaf dataset. This fine-tuningn involves freezing the weights of the early layers (convolutional layers) to retain the learned features from ImageNet, while training the later layers (fully connected layers) to adapt to the specific task of crop disease detection.

5.Training

In this step we train the model using training dataset and monitor its performance on the validation set. Adjust hyperparameters such as batch size, learning rate, and regularization techniques to improve performance and prevent overfitting.

6.Evaluation

Evaluating the model after training on the testing set to estimate its performance in detecting crop diseases. Metrics such as accuracy, precision, recall and F1-score can be used to quantify the model's performance.

7.Fine-tuning and Optimization

Fine-tune the model further by experimenting with different hyperparameters, architectures, and optimization techniques to attain the best possible performance.

8.Deployment

Once fulfilled with the model's performance, then deploy it for real-world crop disease detection applications. This may include integrating the model into a mobile or web application, or deploying it on edge devices for in-field detection. Ensure that the deployment process is scalable, efficient, and user-friendly.

3.3 Model Architecture

VGG 19 Model as a Feature extractor and classifier

The VGG19 (Visual Geometry Group 19) model is a deep convolutional neural network architecture developed by researchers at the University of Oxford's Visual Geometry Group. Introduced in 2014, VGG19 has become a widely adopted and influential model in the field of computer vision.

The VGG19 model is an extension of the original VGG16 architecture, with a total of 19 deep layers, including 16 convolutional layers, 5 max-pooling layers, and 3 fully connected layers. The convolutional layers are responsible for extracting hierarchical visual features from the input images, while the max-pooling layers gradually reduce the spatial dimensions of the feature maps, allowing the model to capture more abstract and high-level representations.

One of the key strengths of the VGG19 model is its simplicity and consistency in design. The network employs 3x3 convolution filters and 2x2 max-pooling operations throughout its architecture, which contributes to its performance and ease of implementation. Additionally, the model utilizes ReLU (Rectified Linear Unit) activation functions, which introduce non-linearity and help the network learn more complex representations.

The VGG19 model has demonstrated excellent performance on a wide range of computer vision tasks, such as image classification, object detection, and semantic segmentation. Its deep and well-structured architecture allows it to capture intricate visual patterns and achieve state-of-the-art results on benchmark datasets like ImageNet.

Despite its depth, the VGG19 model is relatively efficient in terms of computation and memory usage compared to some other deep learning architectures. This, along with its pre-trained weights available for transfer learning, has made VGG19 a popular choice for various computer vision applications, especially when computational resources are limited.

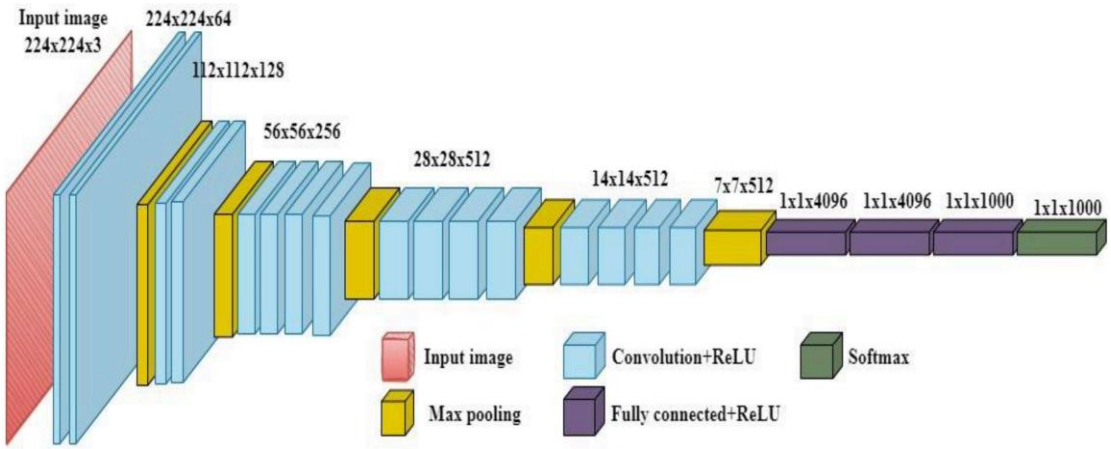


Fig. 3.3 Architecture of VGG 19 Model

3.4 PSEUDO CODE

Pseudo code for setting up environment for image classification using TensorFlow and Keras in Google Colab.

Step by step procedure:

Step 1. Importing necessary libraries

```
import matplotlib.pyplot as plt

import numpy as np

import os

import PIL
```

import tensorflow as tf

from tensorflow import keras

import pandas as pd

import seaborn as sns

from PIL import Image

import cv2

Step 2. Mount Google Drive

Mount Google Drive in Colab to access and store data:

- from google.colab import drive

- drive.mount('/content/drive')

Step 3. Load traning and testing datasets

Define paths for training and testing data

Load images from directories into TensorFlow datasets using

image_dataset_from_directory function:

- Specify batch size, image size, shuffle, labels, and label mode.

- Example:

train_path = "/content/drive/MyDrive/dataset (1)/tomato&grape/training"

test_path = "/content/drive/MyDrive/dataset (1)/tomato&grape/testing"

test_dataset = image_dataset_from_directory(test_path, batch_size=32, image_size=(256,256), shuffle=True)

Step 4. Get class names

Retrieve class names from the training dataset to understand the labels.

Example: class_names = train.class_names print(class_names)

Step 5. Load pre-teained VGG16 base model

Load the VGG19 model from Keras applications, a pre-trained convolutional neural

network:

- Define the input shape expected by the model.

- Initialize the VGG19 model with pre-trained weights from the ImageNet dataset.

base_model = VGG19(weights='imagenet', include_top=False, input_shape=(256, 256, 3))

Step 6. Freeze pre-trained layers in the base model

This means that during training, the parameters of these layers will not be updated, and only the additional layers added on top of the base model will be trained.

for layer in base_model.layers: layer.trainable = False

Step 7. Define the model architecture

Define a custom model architecture using the Sequential API, combining the pre-trained VGG19 model with additional layers:

- Add the base model (VGG16) as the first layer

- Flatten the output of the base model

- Add a fully connected dense layer with 512 units and ReLU activation

- Add the output layer with 8 units for classification and softmax activation

Example:

from keras.models import Sequential from keras.layers import Flatten, Dense, Dropout

folders = ['grape_black measles', 'grape_black rot', 'grape_healthy', 'grape_isoproisis spot','tomato_bacterial', 'tomato_healthy', 'tomato_late blight', 'tomato_leaf curl']

model = Sequential([base_model, Flatten(),Dense(512, activation='relu'), Dense(8, activation='softmax')])

Step 8.Compile the model

Compile the model with specified optimizer, loss function, and evaluation metrics:

- Adam optimizer

- Sparse categorical crossentropy loss

- Accuracy metric

Step 9. Train the model

Train the compiled model using the fit method:

- Provide training dataset, number of epochs, and validation dataset for evaluation.

Step 10: Plot training and validation accuracy

Plot the training and validation accuracy over epochs using Matplotlib:

- Visualize how the model performance improves over training it.

Chapter 4

RESULT AND ANALYSIS

We train the model using the training images and for more understanding we divide the training dataset into train, validate and test dataset. We use the train and validate data set for the training and the test dataset for the prediction of the outcome. The experimental results demonstrate the effectiveness of the VGG19 model in accurately detecting crop diseases.

The below figure shows the average training and validation accuracy and loss calculations.

```
Epoch 1/10
88/88 [==============================] - 1122s 13s/step - loss: 12.2391 - accuracy: 0.8500 - val_loss: 2.0591 - val_accuracy: 0.9329
Epoch 2/10
88/88 [==============================] - 34s 369ms/step - loss: 0.5451 - accuracy: 0.9739 - val_loss: 0.8695 - val_accuracy: 0.9668
Epoch 3/10
88/88 [==============================] - 34s 372ms/step - loss: 0.1947 - accuracy: 0.9914 - val_loss: 0.8038 - val_accuracy: 0.9704
Epoch 4/10
88/88 [==============================] - 34s 378ms/step - loss: 0.2332 - accuracy: 0.9882 - val_loss: 0.8114 - val_accuracy: 0.9700
Epoch 5/10
88/88 [==============================] - 38s 424ms/step - loss: 0.0266 - accuracy: 0.9964 - val_loss: 0.7662 - val_accuracy: 0.9764
Epoch 6/10
88/88 [==============================] - 39s 426ms/step - loss: 0.1502 - accuracy: 0.9929 - val_loss: 0.7582 - val_accuracy: 0.9725
Epoch 7/10
88/88 [==============================] - 38s 425ms/step - loss: 0.2039 - accuracy: 0.9886 - val_loss: 6.6685 - val_accuracy: 0.8629
Epoch 8/10
88/88 [==============================] - 39s 428ms/step - loss: 0.3325 - accuracy: 0.9893 - val_loss: 1.0396 - val_accuracy: 0.9707
Epoch 9/10
88/88 [==============================] - 38s 427ms/step - loss: 0.1042 - accuracy: 0.9950 - val_loss: 1.1343 - val_accuracy: 0.9725
Epoch 10/10
88/88 [==============================] - 39s 428ms/step - loss: 0.1476 - accuracy: 0.9921 - val_loss: 1.4197 - val_accuracy: 0.9557
```

4.1 MODEL ACCURACY

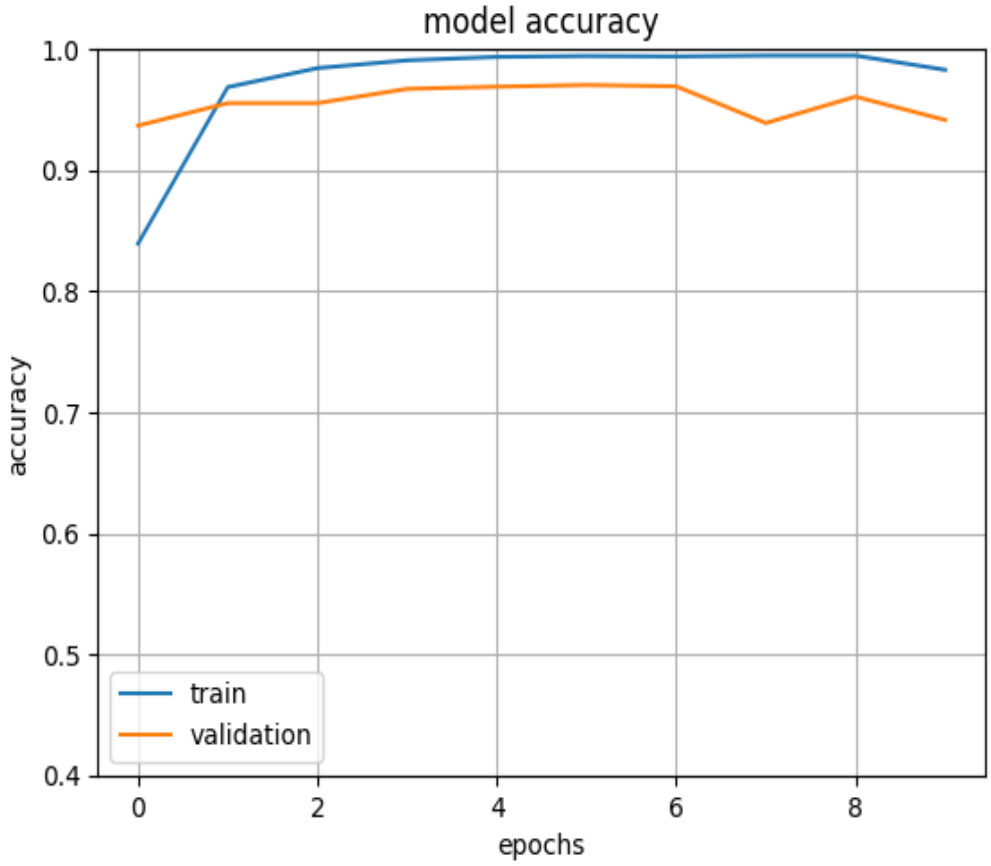


Fig. 4.1 Model Accuracy

4.2 MODEL LOSS

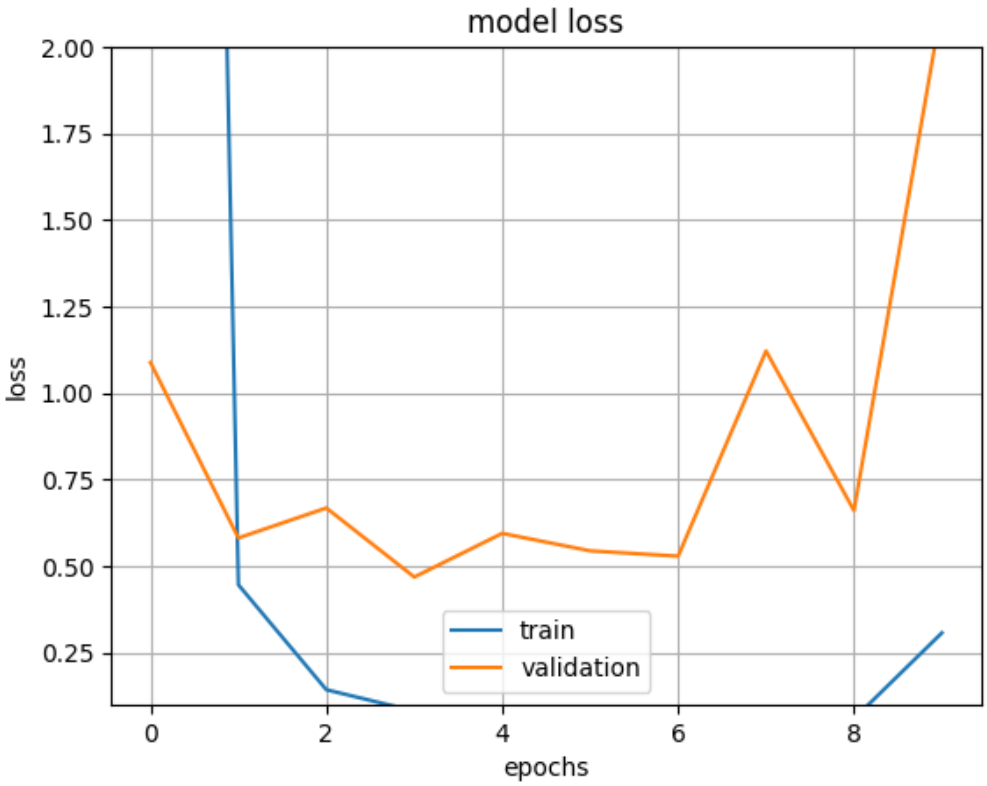


Fig. 4.2 Model loss

4.3 PREDICTED DATASET

The given result is the showcase of how it is detected on the images which are collected from the real field.

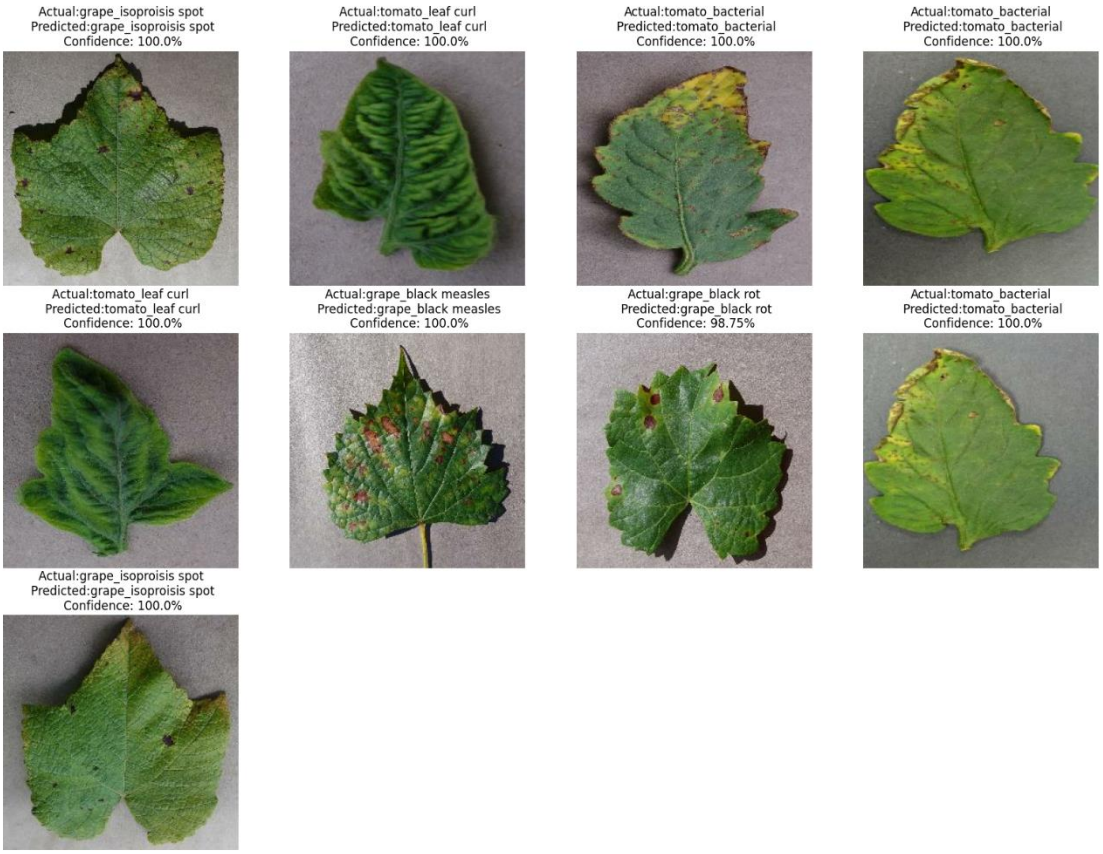


Fig. 4.3 Predicted Dataset

Chapter 5

CONCLUSION

The evolution of crop disease detection models has gone through significant transformation, transitioning from rule-based systems to machine learning algorithms, particularly convolutional neural networks (CNNs). These advancements have facilitated the development of robust models capable of learning intricate disease patterns directly from data, which leads to improved accuracy and adaptability. Crop disease detection models be of service to various stakeholders in agriculture, providing them early disease warnings, facilitating timely interventions, and also offering real-time insights to agribusinesses and agricultural extension services. Despite their advantages, crop disease detection models face some challenges such as the availability of high-quality training data, generalization across different crops and environments, and interpretability of model decisions.

Furthermore, crop disease detection models play a crucial role in research and academia, facilitating the study of disease epidemiology, pathogen dynamics, and host-pathogen interactions. Researchers can use these models for analysis of datasets on a large-scale, identify emerging disease trends, and develop predictive models for future disease outbreaks. Despite their numerous advantages, crop disease detection models face several challenges and limitations that delay their widespread adoption and effectiveness. One of the primary challenges is the availability of high-quality training data. Building robust models requires large, diverse, and accurately annotated datasets, which can be difficult and expensive to acquire, especially for rare or emerging diseases. Models trained on specific crops or geographical regions may struggle to perform accurately when applied to new datasets with varying characteristics. The future of crop disease detection models holds promise with opportunities for innovation and advancement. Integration of multimodal data sources and decentralized computing solutions can enhance model robustness and extend their reach to resource-constrained environments. Collaboration among interdisciplinary teams is essential for developing

holistic solutions that address the complex challenges associated with agricultural innovation.

In conclusion, crop disease detection models play an important role in growth of agricultural technology, providing transformative solutions for disease management and crop protection. By using machine learning and image processing techniques, these empowering models enable rapid and accurate identification of crop diseases, agribusinesses, farmers, and researchers to make informed decisions and take timely actions.

References

1. Krish naik, Deep Learning Paylist

2. Alatawi, A. A., Alomani, S. M., Alhawiti, N. I., & Ayaz, M. (2022). Plant disease detection using AI based VGG-16 model. International Journal of Advanced Computer Science and Applications, 13(4).

3. Ahmed, Imtiaz, and Pramod Kumar Yadav. "Plant disease detection using machine learning approaches." Expert Systems 40.5 (2023): e13136.

4. Roy, Arunabha M., and Jayabrata Bhaduri. "A deep learning enabled multi-class plant disease detection model based on computer vision." Ai 2.3 (2021): 413-428.