Report On

Plant Disease Detection using CNN

Submitted in partial fulfillment of the requirements of the Course project in Semester VII of Fourth Year Computer Engineering

by

Vipul Bhoir (Roll No. 07) Mrudul Chaudhari (Roll No. 12) Abhinav Desai (Roll No. 14)

Mentor

Dr. Megha Trivedi





# Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

**CERTIFICATE**

This is to certify that the Course Project entitled **“ Plant disease Detection using CNN ”** is a bonafide work of **Vipul Bhoir (Roll No. 07), Mrudul Chaudhari (Roll No. 12), Abhinav Desai (Roll No. 14)** submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the degree of **“Bachelor of Engineering”** in Semester VII of Fourth Year **“Computer Engineering” .**

Dr. Megha Trivedi

Mentor

Dr. Megha Trivedi Head of Department

Dr. H.V. Vankudre Principal

# Contents

### Abstract i

[List of Abbreviations iv](#_TOC_250002)

[List of Figures v](#_TOC_250001)

[List of Tables vi](#_TOC_250000)

List of Symbols vii

1. Introduction 1
   1. Introduction
   2. Problem Statement
   3. Objectives
   4. Scope

### Literature Survey 3

* 1. Survey of Existing System
  2. Limitation Existing system or Research gap

### Proposed System 6

* 1. Architecture/ Framework/Block diagram
  2. Algorithm and Process Design
  3. Details of Hardware & Software
  4. Experiment and Results for Validation and Verification
  5. Analysis
  6. Conclusion and Future work.

### References 22

1. Annexure
   1. **Published Paper /Camera Ready Paper/ Business pitch/proof of concept**

# ABSTRACT

Plant diseases cause low agricultural productivity. Plant diseases are challenging to control and identify by the majority of farmers. In order to reduce future losses, early disease diagnosis is necessary. This study presents a deep learning approach for detecting tomato leaf diseases using Convolutional Neural Networks (CNNs). The proposed method involves preprocessing the tomato leaf images, followed by training the CNN model to classify them into one of ten categories: healthy, yellow leaf curl virus (YLCV), bacterial spot (BS), early blight (EB), leaf mold (LM), Septoria leaf spot (SLS) target spot (TS), two spotted spider mite spot(TSSMS), mosaic virus(MV) and late blight (LB). The model was trained using a dataset of 16021 tomato leaf images. The training was conducted for 2 epochs, 5 epochs and the accuracy achieved was 64%, 89% respectively. These results demonstrate the effectiveness of the proposed approach in detecting tomato leaf diseases, and the performance improves with increasing epochs. The automated approach can aid in the early detection and prevention of tomato diseases, which can ultimately help in improving the yield and quality of tomato crops.

# List of Abbreviations

|  |  |
| --- | --- |
| CNN | Convolutional Neural Networks |
| MPL | Max Pooling layer |
| YLCV | Yellow Leaf Curl Virus |
| BS | Bacterial Spot |
| EB | Early Blight |
| LM | Leaf Mold |
| SLS | Septoria Leaf Spot |
| TS | Target Spot |
| TSSMS | Two spotted Spider Mite Spot |
| MV | Mosaic Virus |
| LB | Late Blight |

# List of Figures

Figure 1 Methodology flow chart Figure 2 CNN model architecture Figure 3 Convolution layer Figure 4 Max pooling layer Figure 5 Fully connected layer

Figure 6 visualization of tomato leaf images Figure 7 Loss and Accuracy at 2-5 epochs Figure 8 Disease Detection for Testing Images Figure 9 Disease Detection for New Image

# List of Tables

Table 1 Class distribution of the dataset.

Table 2 Dataset lengths for train, valid and test.

Table 3 Comparison of Loss and Accuracy for train and valid at different epochs.

# CHAPTER – 1

**INTRODUCTION**

## Introduction

Tomatoes are one of the most widely cultivated and consumed crops globally, making it a significant part of the agriculture industry. Unfortunately, tomato plants are susceptible to various diseases, which can lead to significant economic losses due to reduced yield and quality. One of the most common diseases that affect tomato plants is gray leaf spot, which damages and kills the leaves, ultimately hindering the plant's ability to produce fruit. The infection caused by the pathogen responsible for gray leaf spot in tomato plants progresses through four phases: contact, invasion, latency, and onset. Detecting diseases early can help prevent large-scale pandemics and enable appropriate management practices.

Traditional methods of detecting diseases are time-consuming and expensive, especially when the farm is extensive, making it challenging to monitor each plant. Thus, there is a need for a more efficient and cost-effective solution. Image processing techniques can automate the detection of diseases in leaves, thereby saving time, money, and effort. The dataset used in this study consists of 16021 images of ten types of diseased tomato leaf images. All images are resized to 256 x 256 and divided into three parts, namely, training, testing, and validation datasets. By analyzing the features of diseased leaves, image processing technology can accurately diagnose illnesses quickly. Deep learning techniques, specifically convolutional neural networks (CNNs), have proven to be effective in image classification tasks, including plant disease detection.

## Problem Statement

Manual plant disease detection methods are time-consuming and inefficient, particularly for large-scale farms. Traditional disease detection techniques, such as visual inspection, are susceptible to errors and often require a team of experts. Moreover, early disease detection is essential to control and prevent plant diseases, which traditional techniques cannot guarantee. Therefore, there is a need for an accurate, efficient, and automated disease detection approach for tomato plants that can provide early detection and effective prevention.

## Objectives

The main objective of this study is to develop an automated system for detecting and classifying tomato leaf diseases using CNN. Specifically, this study aims to:

1. Develop a CNN model that can accurately detect and classify common tomato leaf diseases, such as early blight, late blight mold, bacteria spot, leaf mold, target spot, yellow leaf curl virus, two spotted spider mite, mosaic virus and septoria leaf spot.
2. Compare the performance of the developed CNN model with at different epochs.
3. Contribute to sustainable agriculture by providing a cost-effective, automated solution to identify tomato leaf diseases at an early stage, thereby enabling farmers to take preventive measures and reduce crop losses.

## Scope

Identifying and recognizing tomato leaves disease is the solution for saving the reduction of large farms in crop disease detection and profit in productivity, it is beneficial in agricultural institute, Biological research.

# CHAPTER – 2 LITERATURE REVIEW

## Survey of Existing System

literature In this chapter, presents a literature survey of traditional plant disease detection approaches based on computer vision technologies that are commonly utilized to extract the texture, shape, color, and other features of disease spots. This method has a low identification efficiency because it relies on an extensive expert understanding of agricultural illnesses. Many academics have conducted significant research based on deep learning technology to increase the accuracy of plant disease detection in recent years, thanks to the fast growth of artificial intelligence technology. The majority of existing techniques to plant disease analysis are based on disease classification.

## Limitation Existing system or Research gap

### A Survey on Supervised Convolutional Neural Network and Its Major Applications; D. T. Mane and U. V. Kulkarni

With the advent of deep learning, the world has proceeded into the new era of machine learning. With the main intention of getting closer to the original goal of machine learning, that is, Artificial Intelligence, deep learning has opened up new avenues to explore. Artificial Neural Networks (ANNs) are biologically motivated machine learning algorithms applied to solve problems, where conventional approaches fail, such as computer vision. It takes in the input, let it be an image or an audio signal, extracts features which describe the input and generalizes these features so that the results obtained can be replicated for other examples of the input. This paper gives an overview of a particular type of ANN, known as supervised Convolutional Neural Network (CNN) and gives information of its development and results in various fields. Initially, we see the history of CNN followed by its architecture and results of its applications. The references of the few used papers have been mentioned here.

### Tomato crop disease classification Using A Pre-Trained Deep Learning Algorithm; Aravind KR, Raja P, Anirudh R.

A study on the classification of three major tomato crop diseases - Early Blight, Late Blight, and Leaf Mold - using a pre-trained deep learning algorithm called VGG16. The authors describe the dataset used for the study, which consisted of images of tomato leaves infected with the three diseases and healthy leaves. The VGG16 algorithm was fine-tuned using transfer learning to classify the images into the four categories. The authors report that the VGG16 algorithm achieved an accuracy of 98.67% in classifying the images, outperforming other algorithms such as Random Forest and K- Nearest Neighbours. The paper also discusses the limitations of the study and potential areas for future research, such as the use of more diverse datasets and the development of a mobile application for farmers to identify crop diseases.

### Attention Embedded Residual CNN for Disease Detection in Tomato Leaves; Karthik R, Hariharan M, Anand Sundar, Mathikshara Priyanka, Johnson Annie, Menaka R.

A dataset consisting of images of tomato leaves affected by five different diseases - Early Blight, Late Blight, Leaf Mold, Septoria Leaf Spot, and Spider Mites - and healthy leaves. The proposed CNN architecture consists of residual blocks, which enable the network to learn the mapping between the input and output more efficiently, and attention modules, which help the network to focus on the most important features in the images. The authors report that the proposed approach achieved an accuracy of 98.3% in detecting tomato crop diseases, outperforming other state-of-the-art approaches such as VGG16 and Inception-v3. The paper also provides a detailed analysis of the performance of the proposed approach on different disease classes and provides visualizations of the attention maps generated by the attention modules.

### Research on deep learning in apple leaf disease recognition. Comput Electron Agric; Zhong Yong, Zhao Ming.

The article presents a study on the use of deep learning algorithms for the recognition of apple leaf diseases. The authors developed a deep learning framework that uses a convolutional neural network (CNN) to automatically identify and classify different apple leaf diseases based on images. The authors trained their model on a large dataset of apple leaf images and achieved high accuracy in disease recognition across multiple apple cultivars. They also demonstrated the potential for their model to be used in real-world scenarios, such as in orchards and nurseries. The findings of this study may have practical applications in the agricultural industry by providing a tool for early detection and diagnosis of apple leaf diseases. This could ultimately lead to improved crop yields and reduced economic losses for apple farmers.

Overall, this article demonstrates the potential for deep learning algorithms to revolutionize the field of crop disease detection and management, with practical applications in a range of crops and settings.

### AI-powered banana diseases and pest detection. Plant Methods. 2019; 15:92; Selvaraj MG, Vergara A, Ruiz H, et al.

A dataset of banana plant images and show that it can accurately detect the presence of diseases and pests with high accuracy. They also demonstrate that the method can be applied in real-world settings using a smartphone app that allows farmers to easily capture and upload images of their plants for analysis. Overall, the study shows the potential of machine learning techniques for plant disease and pest detection and highlights the importance of developing practical and accessible tools to support farmers in monitoring and managing their crops. The authors suggest that their approach could be extended to other crops and regions, contributing to the development of more sustainable and efficient agricultural practices.

# CHAPTER – 3

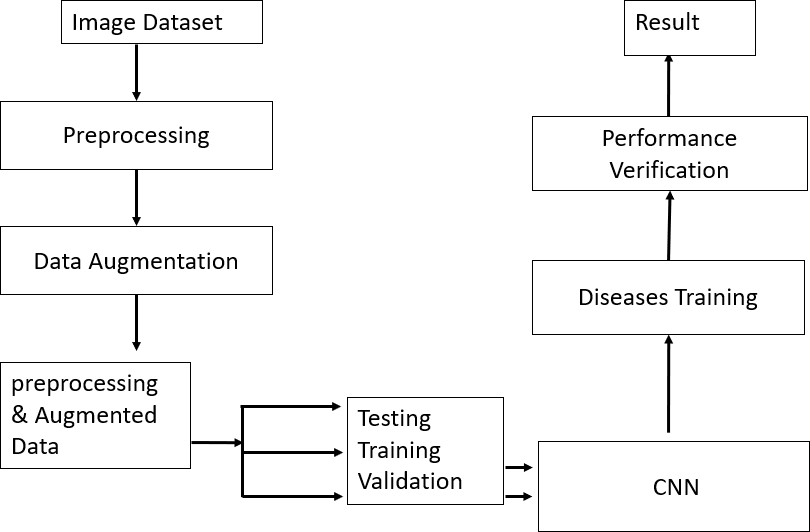
**Proposed System**

This chapter presents a detailed description of the dataset used in this study on tomato leaf disease detection using CNN, including dataset collection, preprocessing, dataset statistics and dataset split for train, valid and test datasets.

## Architecture/ Framework/Block diagram/Methodology

### Methodology Flow chart

The following figure shows the methodology flow chart, it describes the way of approach to detect the tomato leaf diseases.



**Figure 1 Methodology flow chart**

### Data Collection

The dataset used for this project was obtained from Kaggle. The dataset consists of 16,021 tomato leaf images with ten classes: Tomato\_Bacterial\_spot, Tomato\_Early\_blight, Tomato\_Late\_blight, Tomato\_Leaf\_Mold, Tomato\_Septoria\_leaf\_spot, Tomato\_Spider\_mites\_Two\_spotted\_spider\_mite

Tomato Target\_Spot, Tomato Tomato\_YellowLeaf Curl\_Virus, Tomato\_mosaic\_virus and Tomato\_healthy. The images were captured from different locations, seasons, and under different lighting conditions.

### Dataset Selection

The dataset consists of 16,021 images with ten different classes representing different types of tomato leaf diseases and a healthy class from them we have selected only 06 classes consisting of a total 8969 image files. Table 1 presents the class distribution of the dataset.

### Table 1 Class distribution of the dataset.

|  |  |  |
| --- | --- | --- |
| **S.No** | **Type of tomato leaf disease** | **No. of images** |
| 1 | Tomato\_Bacterial\_spot | 2,127 |
| 2 | Tomato\_Septoria\_leaf\_spot | 1,771 |
| 3 | Tomato\_Spider\_mites\_Two\_spotted\_spider\_mite | 1,676 |
| 4 | Tomato Tomato\_YellowLeaf Curl\_Virus | 3,209 |
| 5 | Tomato Tomato\_mosaic\_virus | 373 |
| 6 | Tomato\_healthy | 1,591 |

* + 1. **Preprocessing**

Before training the CNN model, the dataset was preprocessed using several techniques such as data augmentation, normalization, and resizing. **Data Augmentation** is a very popular technique in image processing, especially computer vision to increase the diversity and amount of training data by applying random (but realistic) transformations. Data augmentation techniques such as rotation, flipping (horizontal and vertical), and random cropping were applied to increase the size of the dataset and introduce more variability in the data.

**Normalization** was applied to scale the pixel values between 0 and 255 to improve the convergence of the model during training.

**Resizing** was also applied to standardize the image size to 256x256 pixels to reduce the computational cost of training the mode and also send the images as batches as the batch size is 16.

### Dataset Split

To evaluate the performance of the CNN model, the dataset was split into three subsets, namely the training dataset, validation dataset, and test dataset. The training dataset was used to train the model, the validation dataset was used to tune the hyperparameters, and the test dataset was used to evaluate the performance of the model on unseen data.

The dataset was split randomly into the three subsets, with the training dataset containing 80% of the images, the validation dataset containing 10% of the images, and the test dataset containing 10% of the images. Table 2 shows the dataset lengths for the train, validation, and test datasets.

### Table 2 Dataset lengths for train, valid and test.

|  |  |
| --- | --- |
| Train dataset length | 448 |
| Valid dataset length | 57 |
| Test dataset length | 56 |
| Total dataset length | (448+57+56)=561 |

The dataset split ensures that the model is trained on a sufficiently large dataset while also allowing for a fair evaluation of its performance on unseen data.

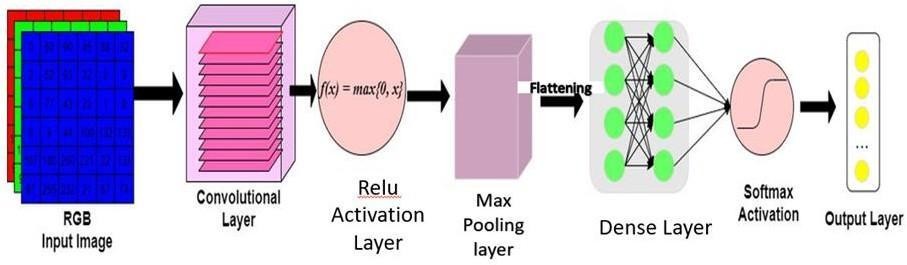
This chapter describes the methodology of the system. In the next chapter, Chapter 4, presents the CNN model architecture and the training process. The dataset split presented in this chapter is used to train and evaluate the performance of the model,

### CNN Model Architecture

This chapter describes the Convolutional Neural Network (CNN) model architecture used for our image classification task and the process of training the model. The architecture includes multiple convolutional and pooling layers, followed by fully connected layers, and ends with a softmax output layer. The training process involves initializing the model parameters, defining the loss function, selecting an optimization algorithm, and iteratively updating the model parameters using backpropagation and gradient descent.

A Convolutional Neural Network (CNN) is a type of artificial neural network commonly used for image and video analysis, recognition, and processing. It is designed to automatically extract meaningful features from raw pixel data of an image, enabling it to recognize objects, faces, shapes, and patterns.

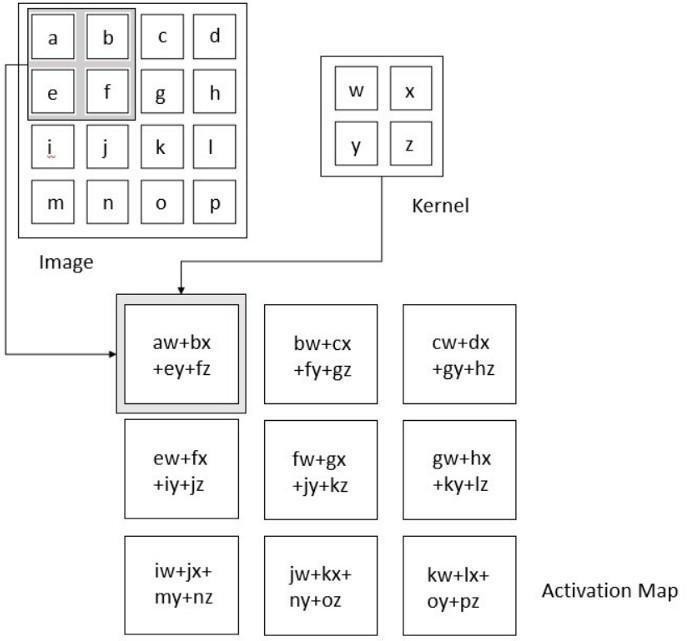
CNNs are inspired by the structure and function of the visual cortex in the brain. The network is made up of a series of interconnected layers, each consisting of several neurons that perform simple computations on the input data. The layers are typically arranged in a specific order, including convolutional layers, pooling layers, and fully connected layers. The following fig 2 shows the CNN model architecture with properly connected layers.



**Figure 2 CNN model architecture**

### Convolution Layer

Convolutional layers are the core building blocks of a CNN. They apply filters or kernels to the input image, sliding over the entire image and performing a dot product between the filter and the input pixels. This process generates a feature map, highlighting the regions of the input image that are most important for recognizing a particular pattern or object. Fig 3 shows the mathematical operation kernel filter with input image.



### Figure 3 Convolution layer

Fig 3 illustrates the mathematical operation of the convolution layer, where a 2x2 kernel filter is convolved with an input image. The resulting feature map highlights the edges and corners of the object in the image.

### Relu Activation Function

The Rectified Linear Unit (ReLU) activation function is a widely used activation function in CNNs. It introduces nonlinearity into the network and improves its ability to model complex relationships between the input and output data. The ReLU function only allows positive values to pass through the neuron. When the input to the neuron is positive, the output is equal to the input value, and when the input is negative, the output is equal to zero.

The ReLU function is defined as f(x) = max{0,x}.

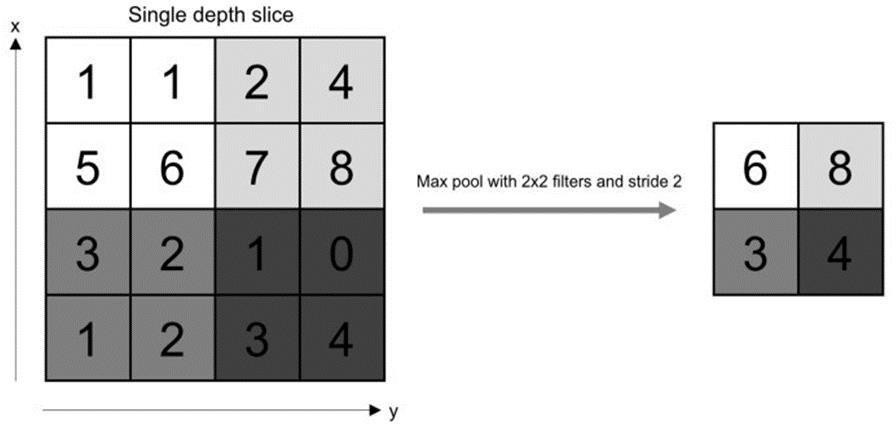
### Pooling Layers

Pooling layers are an essential component of Convolutional Neural Networks (CNNs) used in computer vision applications. They are used to reduce the spatial dimensionality (width and height) of the input data while retaining its essential features.

Max pooling is a commonly used type of pooling layer in which the maximum value within a defined region of the input feature map is selected and then passed on to the next layer. The size of this defined region (often referred to as the pooling window or kernel size) is typically set by the user. Fig 4 shows the operation of max pooling.

For example, let's say we have an input feature map with a size of 6x6 and a pooling window size of 2x2. The max pooling operation would take place as follows:

1. The input feature map is divided into non-overlapping regions of size 2x2.
2. The maximum value within each region is identified.
3. A new feature map is created with a size of 3x3, consisting of the maximum values from each region.



### Figure 4 Max pooling layer

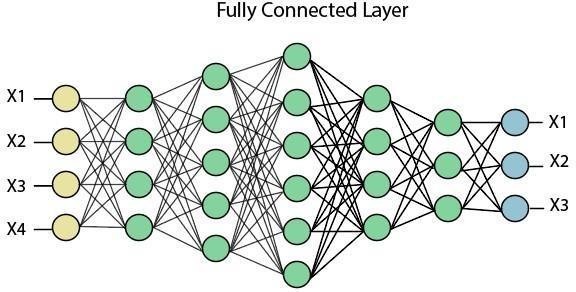
* + - * **Fully Connected Layer**

**Fully connected input layer** – The preceding layers' output is "flattened" and turned into a single vector which is used as an input for the next stage.

**Fully connected middle layer** – adds weights to the inputs from the feature analysis to anticipate the proper label.

**Fully connected output layer** – offers the probability for each label in the end.

Fig 5 shows the internal working of fully connected layer



**Figure 5 Fully connected layer**

## Training Process

The training process involves initializing the model parameters, defining the loss function, selecting an optimization algorithm, and iteratively updating the model parameters using backpropagation and gradient descent. It discuss step-by-step process involved in training a neural network:

1. The first step in the training process is loading and preprocessing the training data. This involves normalizing the data, splitting it into batches, and converting it into the appropriate format for the model. This step is discussed in Chapter 3, section 3.3 and 3.5.
2. The second step in the training process is defining the model architecture. This step involves specifying the neural network's architecture, including the number and type of layers, activation functions, optimizer, and loss function. The architecture of the CNN model is discussed in section 4.1.
3. The third step in the training process is compiling the model. This step involves configuring the model for training by specifying the optimizer, loss function, and any additional metrics to track during training. The Adam optimizer and Sparse Categorical Cross entropy loss function are discussed in current chapter sub section

4.1.4 and sub section 4.1.5, respectively.

1. The final step in the training process is training the model. This step involves feeding the training data into the model, computing the output, and adjusting the model parameters using the Adam optimizer algorithm to minimize the loss function. The number of training epochs determines how many times the entire training dataset is used to train the model. The trained model is used for testing on the test set in the next chapter – 6 Results under the section 6.2 Training Results.

This chapter discussed CNN model architecture and the step-by-step process involved in training a neural network model, including loading and preprocessing the training data, defining the model architecture, compiling the model, and training the model. In the next chapter, Chapter 5, presents the algorithm process, implementation code for the automatic detection for tomato leaf disease detection.

## Algorithm and Process Design

* + 1. **Algorithm Process**

**Step. 1 -** Import the necessary libraries.

**Step. 2 -** Set the input parameters, such as image size, batch size, and number of classes**. Step. 3 –** Load the dataset and preprocess the images

**Step. 4** -Define the CNN model architecture.

**Step. 5** - To train the CNN model at different epochs.

**Step. 6** – Evaluate the performance of the model and save the model with .h5 format

**Step. 7** - Reload the model and predict the tomato leaf images.

## Code

**main.py**

import os os.environ['TF\_ENABLE\_ONEDNN\_OPTS'] = "0" from fastapi import FastAPI, File, UploadFile

from fastapi.middleware.cors import CORSMiddleware import uvicorn

import numpy as np from io import BytesIO from PIL import Image import tensorflow as tf

app = FastAPI() origins = [

"[http://localhost](http://localhost/)",

"http://localhost:3000",

]

app.add\_middleware( CORSMiddleware, allow\_origins=origins, allow\_credentials=True, allow\_methods=["\*"], allow\_headers=["\*"],

)

model = tf.keras.models.load\_model(r'models\tomatoE5.h5',compile=False) tomato\_classes = ['Tomato\_Bacterial\_spot',

'Tomato\_Septoria\_leaf\_spot', 'Tomato\_Spider\_mites\_Two\_spotted\_spider\_mite', 'Tomato Tomato\_YellowLeaf Curl\_Virus', 'Tomato Tomato\_mosaic\_virus', 'Tomato\_healthy']

def image\_into\_np\_array(data) -> np.ndarray: image = np.array(Image.open(BytesIO(data))) return image

# plant\_name: Annotated[str, Form()], => for different models @app.post("/predict")

async def predict( file: UploadFile = File(...) ): image = image\_into\_np\_array( await file.read()) img = np.expand\_dims(image, 0)

predictions = model.predict(img)

predicted\_label = tomato\_classes[np.argmax(predictions[0])] score = round(np.max(predictions[0]),2)

return {

'label' : predicted\_label, 'score' : float(score)

}

if name == ' main ':

uvicorn.run(app, host = 'localhost', port = 8000)

## Hardware & Software Requirements

### Hardware Requirements:

* + - 1. Windows/ Linux / MacOS Operating System.
      2. Processor : Standard Processor with a speed of 1.6 GHz.
      3. RAM : 256 MB RAM or more.
      4. Hard Disk : 20 GB or more sufficient storage space.
      5. Monitor : Standard Color Monitor.
      6. Internet Connection: Good internet connection with at least 4 mbps speed.

### Software Requirements:

* + - 1. tensorflow = 2.16.2
      2. Python 3.10.\*
      3. numpy
      4. pandas
      5. pillow
      6. jupyter/colab
      7. web browser (eg. Google Chrome)
      8. React JS

## RESULTS

In this chapter, it presents the visualization of the training results and predictive analysis obtained from the proposed CNN-based approach.

## Visualization

The aim of this study is to detect the different types of tomato leaf diseases using convolutional neural networks (CNNs). The dataset used for this study is Plant Village, taken from the Kaggle website, which contains ten types of tomato leaf images with labels indicating healthy and unhealthy leaves. Figure 8 shows the different types of tomato leaf images with their corresponding labels.



**Figure 6 visualization of tomato leaf images**

## Training result

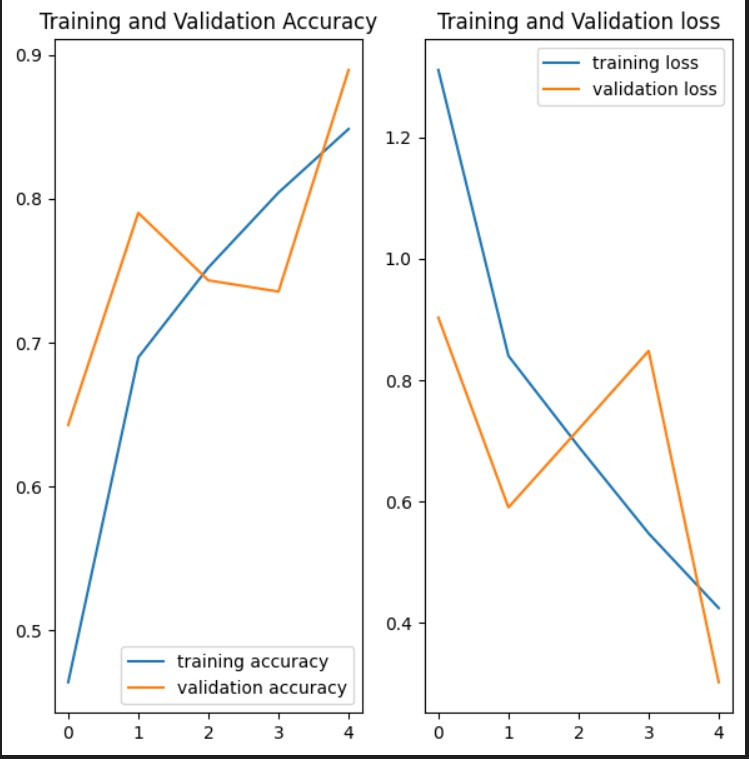
The training results for a CNN typically include the loss and accuracy metrics during the training process. The loss metric is a measure of how well the model is performing on the training data and is typically calculated as the difference between the predicted output and the actual output. The goal of the training process is to minimize the loss function. As discussed chapter 4, section 4.1 the concept of CNN model architecture and section 4.2 the training process. CNN model was trained at 2 epochs, 5 epochs. The total dataset length is 561. As discussed chapter 3 Table 2 shows the dataset lengths for train, valid and test.

From the results shown in Table 3, it is observed that the accuracy increases and the loss decreases with a modification in the number of epochs.

### Table 3 Comparison of Loss and Accuracy for train and valid at different epochs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.**  **No** | **No. of**  **epochs** | **Train**  **Accuracy** | **Train**  **Loss** | **Valid**  **Accuracy** | **Valid**  **Loss** |
| 1 | 2 | 0.64 | 0.34 | 0.61 | 0.45 |
| 2 | 5 | 0.84 | 0.42 | 0.89 | 0.30 |

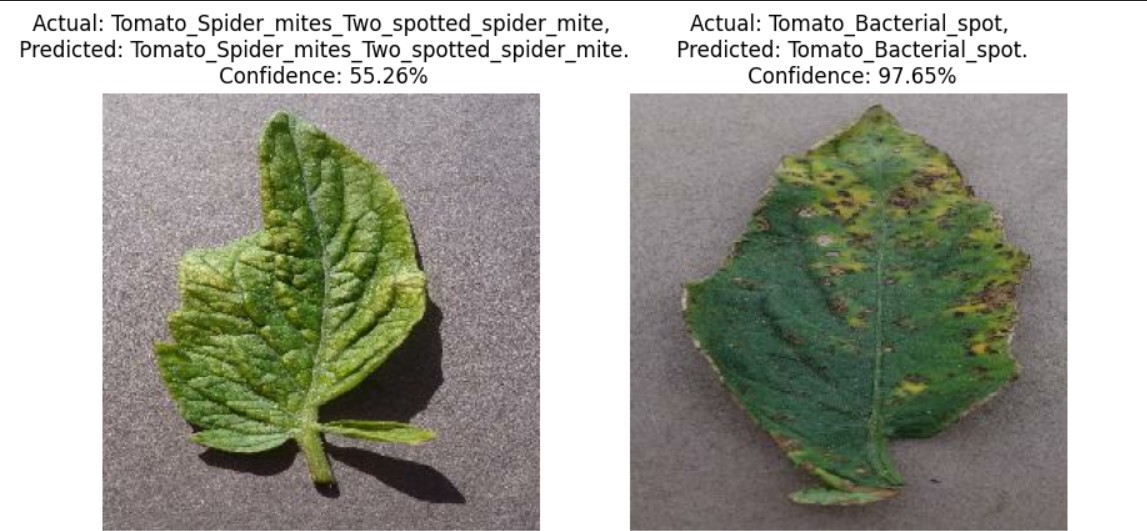
The below figure shows the variations of accuracy and loss for both train and validation.



**Figure 7 variations of accuracy and loss for both train and validation**

## Correctly Predicted Leaf Diseases

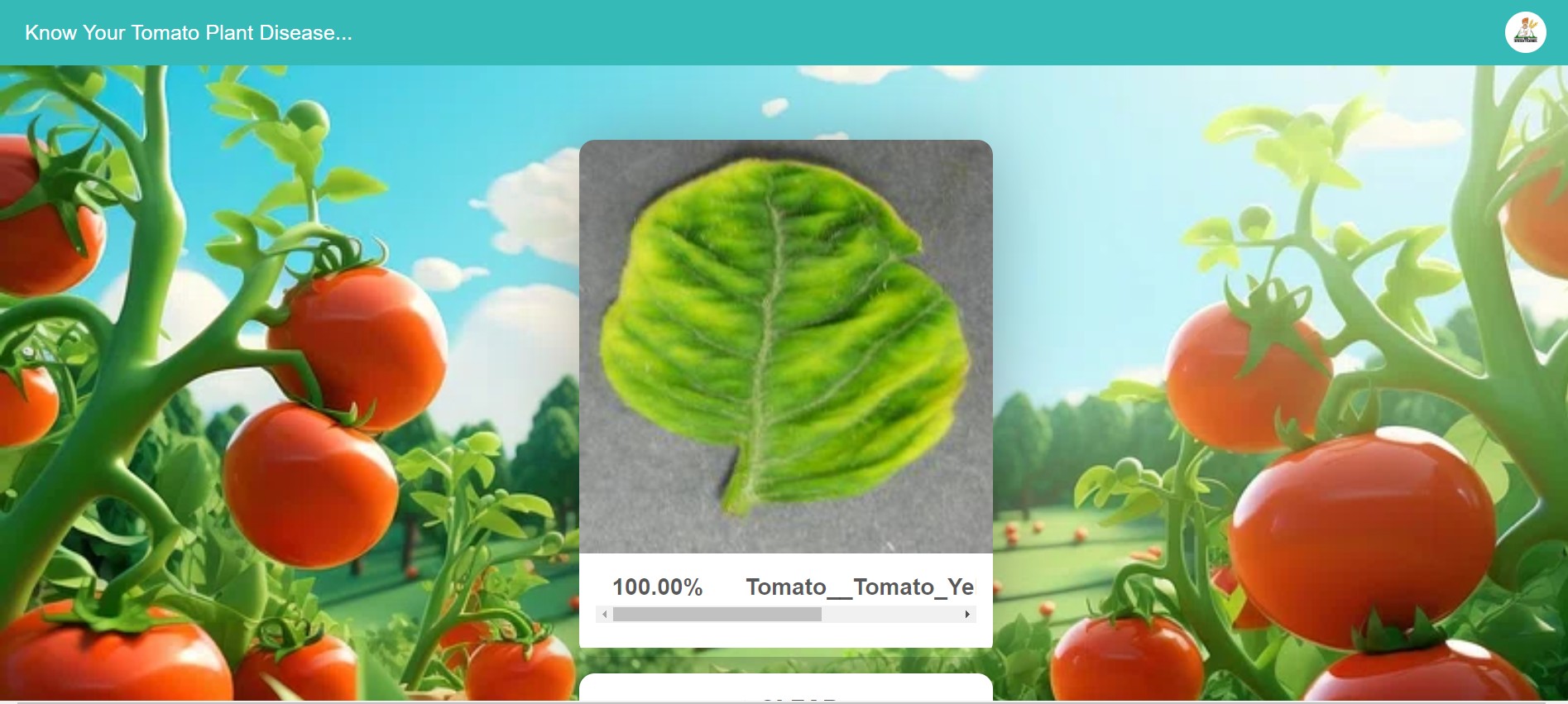
Actually, the trained CNN model is saved with saved\_model.h5, at the time of predicting an image, reloading it and doing predictions.



**8 Disease Detection for Testing Images**

## Website and Tomato Plant Disease Detection

To predict the disease of a tomato plant you just have to drag and drop an image of the leaf of the tomato plant. below fig. accurately predicts disease with 100% score.



**Figure 9 Disease Detection for New Image**

## CONCLUSION AND FUTURE SCOPE

* + 1. **Conclusion**

In this project, we have presented an approach for tomato leaf disease detection using convolutional neural networks (CNN). We trained a deep learning model using a dataset of tomato leaf images, which was collected from various sources. The trained model was able to accurately detect the presence of ten common tomato leaf diseases, namely, bacterial spot, early blight, late blight, leaf mold, Septoria leaf spot, spider mites two spotted spider mite, target spot, yellow leaf curl virus, mosaic virus and healthy. The proposed system is designed to provide an easy-to-use and efficient solution for detecting tomato leaf diseases. It uses a web interface page that allows end-users to upload images of tomato leaves and get real-time predictions on the presence of diseases. The system is capable of processing a large number of images quickly, making it ideal for use in agricultural applications.

## Future Scope

Tomato leaf disease detection using CNN has great potential for future applications. Here are some possible future scopes for this technology:

* + - * Real-time disease detection: The current project used pre-captured images of tomato leaves for disease detection. In the future, the system can be designed to detect diseases in real-time using a camera attached to a robotic arm that moves around the tomato plants. This would enable early detection and treatment of diseases, thus improving crop yields and reducing losses.
      * Transfer learning: The current project used a CNN model. In the future, transfer learning can be used to improve the accuracy of the model for large datasets. This would involve using pre-trained CNN models that have been trained on a large dataset and fine-tuning them on the tomato leaf disease dataset.
      * Deployment on mobile devices: The current project was implemented on a desktop computer and working as a website. In the future, the system can be optimized for deployment on mobile devices such as smartphones and tablets. This would enable farmers to use the system in the field for real-time disease detection and treatment

# CHAPTER – 4 REFERENCES

1. D. T. Mane and U. V. Kulkarni, “A survey on supervised convolutional neural network and its major applications,” International Journal of Rough Sets and Data Analysis, vol. 4, no. 3, pp. 71–82, 2017.
2. Aravind KR, Raja P, Anirudh R. Tomato crop disease classification Using A Pre-Trained Deep Learning Algorithm, Procedia Comput Sci. 2018; 133:1040–7.
3. Karthik R, Hariharan M, Anand Sundar, Mathikshara Priyanka, Johnson Annie, Menaka R. Attention embedded residual CNN for disease detection in tomato leaves. Applied Soft Comput. 2020.
4. Zhong Yong, Zhao Ming. Research on deep learning in apple leaf disease recognition. Comput Electron Agric. 2020; 168:105146.
5. Selvaraj MG, Vergara A, Ruiz H, et al. AI-powered banana diseases and pest detection. Plant Methods. 2019; 15:92.
6. Girshick, R., J. Donahue, T. Darrell, and J. Malik. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation." CVPR '14 Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition. Pages 580-587. 2014.
7. Fuentes A, Yoon S, Kim SC, Park DS. A robust deep-learning-based detector for real-time tomato plant diseases and pest’s recognition. Sensors. 2022; 2017:17.
8. Picon A, Alvarez-Gila A, Seitz M, Ortiz-Barredo A, Echazarra J, Johannes

A. Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild. Comput Electron Agric. 2019;1(161):280