

# CROP DISEASE DETECTION USING DEEP LEARNING

*Minor project-II report submitted  
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology  
in  
Information Technology**

**By**

<b>B. LEELA KRISHNA MOHAN</b>	<b>(21UTIT0009)</b>	<b>(VTU19580)</b>
<b>M. MANASWINI</b>	<b>(21UTIT0034)</b>	<b>(VTU20353)</b>
<b>U. VARSHITHA</b>	<b>(21UTIT0057)</b>	<b>(VTU19181)</b>

*Under the guidance of  
Dr. N. KATHIRVEL, M.E., Ph.D.,  
ASSISTANT PROFESSOR(SG)*



**DEPARTMENT OF INFORMATION TECHNOLOGY  
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF  
SCIENCE & TECHNOLOGY**

**(Deemed to be University Estd u/s 3 of UGC Act, 1956)**

**Accredited by NAAC with A++ Grade  
CHENNAI 600 062, TAMILNADU, INDIA**

**May, 2024**

# **CROP DISEASE DETECTION USING DEEP LEARNING**

*Minor project-II report submitted  
in partial fulfillment of the requirement for award of the degree of*

**Bachelor of Technology  
in  
Information Technology**

**By**

<b>B. LEELA KRISHNA MOHAN</b>	<b>(21UTIT0009)</b>	<b>(VTU19580)</b>
<b>M. MANASWINI</b>	<b>(21UTIT0034)</b>	<b>(VTU20353)</b>
<b>U. VARSHITHA</b>	<b>(21UTIT0057)</b>	<b>(VTU19181)</b>

*Under the guidance of  
Dr. N. KATHIRVEL, M.E., Ph.D.,  
ASSISTANT PROFESSOR(SG)*



**DEPARTMENT OF INFORMATION TECHNOLOGY  
SCHOOL OF COMPUTING**

**VEL TECH RANGARAJAN DR. SAGUNTHALA R&D INSTITUTE OF  
SCIENCE & TECHNOLOGY**

**(Deemed to be University Estd u/s 3 of UGC Act, 1956)**

**Accredited by NAAC with A++ Grade  
CHENNAI 600 062, TAMILNADU, INDIA**

**May, 2024**

# CERTIFICATE

It is certified that the work contained in the project report titled "CROP DISEASE DETECTION USING DEEP LEARNING" by "B. LEELA KRISHNA MOHAN (21UTIT0009), M. MANASWINI (21UTIT00034), U. VARSHITHA (21UTIT0057)" has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

**Signature of Supervisor**

**Information Technology**

**School of Computing**

**Vel Tech Rangarajan Dr. Sagunthala R&D**

**Institute of Science & Technology**

**May, 2024**

**Signature of HOD**

**Information Technology**

**School of Computing**

**Vel Tech Rangarajan Dr. Sagunthala R&D**

**Institute of Science & Technology**

**May, 2024**

# DECLARATION

We declare that this written submission represents my ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

(B.LEELA KRISHNA MOHAN)

Date:        /        /

(Signature)

(M.MANASWINI)

Date:        /        /

(Signature)

(U.VARSHITHA)

Date:        /        /

# APPROVAL SHEET

This project report entitled (CROP DISEASE DETECTION USING DEEP LEARNING) by (B.LEELA KRISHNA MOHAN (21UTIT0009)), (M.MANASWINI (21UTIT0034)), (U.VARSHITHA (21UTIT0057)) is approved for the degree of B.Tech in Information Technology.

**Examiners**

**Supervisor**

Dr. N. Kathirvel, M.E.,Ph.D.,

**Date:**        /        /

**Place:**

# ACKNOWLEDGEMENT

We express our deepest gratitude to our respected **Founder Chancellor and President Col. Prof. Dr. R. RANGARAJAN B.E. (EEE), B.E. (MECH), M.S (AUTO),D.Sc., Foundress President Dr. R. SAGUNTHALA RANGARAJAN M.B.B.S.** Chairperson Managing Trustee and Vice President.

We are very much grateful to our beloved **Vice Chancellor Prof. S. SALIVAHANAN**, for providing us with an environment to complete our project successfully.

We record indebtedness to our **Professor & Dean, Department of Computer Science & Engineering, School of Computing, Dr. V. SRINIVASA RAO, M.Tech., Ph.D.**, for immense care and encouragement towards us throughout the course of this project.

We are thankful to our **Head, Department of Information Technology, Dr. J. VISUMATHI, M.E., Ph.D.**, for providing immense support in all our endeavors.

We also take this opportunity to express a deep sense of gratitude to our **Internal Supervisor and Coordinator Dr. N. KATHIRVEL, M.E., Ph.D.**, for his cordial support, valuable information and guidance, he helped us in completing this project through various stages.

We thank our department faculty, supporting staff and friends for their help and guidance to complete this project.

<b>B.LEELA KRISHNA MOHA</b>	<b>(21UTIT0009)</b>
<b>M.MANASWINI</b>	<b>(21UTIT0034)</b>
<b>U.VARSHITHA</b>	<b>(21UTIT0057)</b>

## ABSTRACT

Traditional farming is going out of date nowadays. Technologies are being introduced in the farming sector for the past decade and in recent years it is seen that the participation of deep learning and machine learning is playing an integral role in solving traditional problems. The introduction of new technology has increased the productivity of farmers and also increased the yields and quality of the crops too. Plant diseases are a serious concern for the consumers and the farmers too. It does not only carry some harmful bacteria within itself however it compromises the yield of the crops too. Deep learning-enabled developments in the field of computer vision have paved the path for computer assisted plant disease diagnosis Various deep learning algorithm like AlexNet and LeNet-5 is applied on a publicly available dataset (plantvillage dataset) so that the neural network can capture the various features of a specific disease and diagnose it accordingly using a human-like decision making skill. Detecting crop diseases using deep learning, specifically the LeNet architecture, represents a significant stride towards sustainable agriculture. By leveraging convolutional neural networks (CNNs) like LeNet, we can automate the identification and classification of various diseases that afflict crops, enabling timely interventions and minimizing yield losses. The LeNet architecture is renowned for its effectiveness in image classification tasks, making it an ideal candidate for crop disease detection. In this context, the process typically involves collecting a diverse dataset of images showcasing healthy crops as well as those affected by various diseases. These images serve as the foundation for training the LeNet model, which learns to discern patterns and features indicative of different crop ailments. Through successive layers of convolution, pooling, and fully connected layers, LeNet extracts hierarchical representations from input images, gradually learning to differentiate between healthy and diseased crops based on subtle visual cues. During the training phase, the model adjusts its internal parameters through backpropagation, optimizing its ability to accurately classify unseen images. the LeNet-based crop disease detection system can swiftly analyze images captured from fields or farms, swiftly identifying the presence of diseases such as powdery mildew, leaf rust, or blight.

**Keywords:** AlexNet, Convolutional Neural Networks, Deep Learning, Farming, Fully Connected, LeNet, Neural Network, Plant village, Plant Diseases, Pooling Layers

# LIST OF FIGURES

4.1	<b>Crop Disease Detection Architecture Diagram</b>	12
4.2	<b>Data Flow Diagram for Crop Disease Detection</b>	15
4.3	<b>Use Case Diagram of Crop Disease Detection</b>	16
4.4	<b>Class Diagram of Crop Disease Detection</b>	17
4.5	<b>Sequence Diagram of Crop Disease Detection</b>	18
4.6	<b>Collabration Diagram of Crop Disease Detection</b>	19
4.7	<b>Activity Diagram of Crop Disease Detection</b>	20
4.8	<b>LeNet5 Architecture</b>	24
4.9	<b>LENet-5 Metrics Graph</b>	27
5.1	<b>Input Design for Crop Diseases Identification from Plant Village Dataset</b>	29
5.2	<b>Output Design for Crop Diseases Identification</b>	30
5.3	<b>Test result of Crop Disease Identification</b>	32
6.1	<b>Output of Crop Leaf</b>	36
6.2	<b>Output of Different Leafs</b>	37
8.1	<b>Plagiarism Report</b>	39
9.1	<b>Poster Presentation for Crop Disease Detection</b>	45



# LIST OF ACRONYMS AND ABBREVIATIONS

S.NO	ABBREVIATION	DEFINITION
1.	FAO	Food and Agriculture Organization
2.	SVM	Support Vector Machine
3.	MLP	Multilayer Perceptron
4.	NIH	National Institutes of Health
5.	RNN	Recurrent Neural Network
6.	VGG	Visual Geometry Group
7.	CNN	Convolutional Neural Network

# TABLE OF CONTENTS

	Page.No
<b>ABSTRACT</b>	<b>v</b>
<b>LIST OF FIGURES</b>	<b>vi</b>
<b>LIST OF ACRONYMS AND ABBREVIATIONS</b>	<b>vii</b>
<b>1 INTRODUCTION</b>	<b>1</b>
1.1 Introduction . . . . .	1
1.2 Aim of the project . . . . .	1
1.3 Project Domain . . . . .	2
1.4 Scope of the Project . . . . .	2
<b>2 LITERATURE REVIEW</b>	<b>3</b>
<b>3 PROJECT DESCRIPTION</b>	<b>7</b>
3.1 Existing System . . . . .	7
3.2 Proposed System of Crop Disease Detection . . . . .	7
3.3 Feasibility Study . . . . .	8
3.3.1 Economic Feasibility . . . . .	8
3.3.2 Technical Feasibility . . . . .	8
3.3.3 Social Feasibility . . . . .	9
3.4 System Specification . . . . .	9
3.4.1 Hardware Specification . . . . .	9
3.4.2 Software Specification . . . . .	10
3.4.3 Standards and Policies . . . . .	10
<b>4 METHODOLOGY</b>	<b>12</b>
4.1 Proposed Architecture of Crop Disease Detection . . . . .	12
4.2 Design Phase . . . . .	15
4.2.1 Data Flow Diagram of Crop Disease Detection . . . . .	15
4.2.2 Use Case Diagram of Crop Disease Detection . . . . .	16
4.2.3 Class Diagram of Crop Disease Detection . . . . .	17

4.2.4	Sequence Diagram of Crop Disease Detection . . . . .	18
4.2.5	Collaboration diagram of Crop Disease Detection . . . . .	19
4.2.6	Activity Diagram of Crop Disease Detection . . . . .	20
4.3	Algorithm & Pseudo Code . . . . .	20
4.3.1	CNN Algorithm . . . . .	20
4.3.2	Pseudo Code of Crop Disease Detection . . . . .	21
4.4	Module Description . . . . .	21
4.4.1	Training the Dataset Module . . . . .	21
4.4.2	Step1:Data Collection . . . . .	22
4.4.3	Step2:Data Processing . . . . .	22
4.4.4	Applying LeNet-5 Architecture . . . . .	23
4.5	Steps to execute/run/implement the project . . . . .	27
4.5.1	Step1:Installing Anaconda . . . . .	27
4.5.2	Step2:Download Tensorflow and Keras . . . . .	28
4.5.3	Step 3:Detection of Crop Disease with LeNet-5 . . . . .	28
<b>5</b>	<b>IMPLEMENTATION AND TESTING</b>	<b>29</b>
5.1	Input and Output . . . . .	29
5.1.1	Crop Disease Identification Input Design . . . . .	29
5.1.2	Early Blight Late Blight Output Design . . . . .	30
5.2	Testing . . . . .	30
5.3	Types of Testing . . . . .	30
5.3.1	Unit testing . . . . .	30
5.3.2	Integration testing . . . . .	31
5.3.3	Test Result . . . . .	32
<b>6</b>	<b>RESULTS AND DISCUSSIONS</b>	<b>33</b>
6.1	Efficiency of the Proposed System . . . . .	33
6.2	Comparison of Existing and Proposed System . . . . .	33
6.3	Sample Code . . . . .	34
<b>7</b>	<b>CONCLUSION AND FUTURE ENHANCEMENTS</b>	<b>38</b>
7.1	Conclusion . . . . .	38
7.2	Future Enhancements . . . . .	38
<b>8</b>	<b>PLAGIARISM REPORT</b>	<b>39</b>

<b>9</b>	<b>SOURCE CODE &amp; POSTER PRESENTATION</b>	<b>40</b>
9.1	Source Code . . . . .	40
9.2	Poster Presentation . . . . .	45
	<b>References</b>	<b>46</b>

# Chapter 1

## INTRODUCTION

### 1.1 Introduction

India being an agriculture country, about 70 percent of the population depends on it as their main source of income and food. Agriculture plays an important part of the Indian economy as it contributes about 17 percent of the total GDP. Farmers have a wide range in selecting their crops and finding a suitable pesticide for it but in spite of all their efforts it can all be vain if they can't identify the disease plaguing their crops. Thus, disease on crops can significantly reduce the quality and quantity of agricultural products along with economical damage to the farmers. To successfully cultivate crops without incurring much loss we need to properly identify the disease and remedy it, this requires a lot of work and processing time as detecting each and every plant can be tedious and time consuming. To lessen the burden of the farmers along with their losses we propose the use of a system which can detect infected plants so that we can curb the spread of infection and diseases at an earlier step thus reducing losses and crop failure.

In most cases symptoms like fungal infection and rot can be seen on the leaves, stem and fruit. This project provides an insight into how we deal with the problem and further discuss the challenges of our work and how we can improve upon it in future work.

### 1.2 Aim of the project

To classify different plant diseases, we plan to design a deep learning system so that a person without expertise in software should also be able to use it easily. The proposed system is made to predict plant diseases using the leaves as an identifying factor. It explains the analysis of our methodology along with some of the feature engineering of the data. A large number of images is collected for each disease and is classified into database images and input images. The primary attributes of the leaves that are important are the shape and texture-oriented features. The figure provided

below gives us an insight into the basic principle of our system along with an idea about how the system works.

### **1.3 Project Domain**

Crop disease detection using deep learning is a pivotal domain in modern agriculture, revolutionizing the way farmers monitor and manage their crops. This project focuses on leveraging advanced deep learning techniques to accurately identify various diseases affecting crops at different growth stages. By analyzing images of leaves, stems, and fruits captured through drones or smartphones, the deep learning models are trained to detect subtle visual cues indicative of diseases such as fungal infections, bacterial blights, and viral symptoms. The domain encompasses a wide range of crops including staple like potato and grape, cash crops like cotton and sugarcane, as well as fruits and vegetables, making it applicable across diverse agricultural landscapes.

In this project domain, the primary objective is to develop robust deep learning models capable of not only detecting crop diseases but also providing insights into disease severity and progression. By integrating image processing algorithms with convolutional neural networks, the models can effectively analyze temporal changes in crop health, enabling timely interventions to prevent yield loss and optimize resource allocation.

### **1.4 Scope of the Project**

To classify different plant diseases, we plan to design a deep learning system so that a person without expertise in software should also be able to use it easily. The proposed system is made to predict plant diseases using the leaves as an identifying factor. It explains the analysis of our methodology along with some of the feature engineering of the data. A large number of images is collected for each disease and is classified into database images and input images. The primary attributes of the leaves that are important are the shape and texture-oriented features. The figure provided below gives us an insight into the basic principle of our system along with an idea about how the system works.

## Chapter 2

# LITERATURE REVIEW

[1]Agarwal et al.(2021) developed Plant diseases and pests detection based on deep learning Plant diseases and pests are important factors determining the yield and quality of plants. Plant diseases and pests identification can be carried out by means of digital image processing. In recent years, deep learning has made breakthroughs in the field of digital image processing, far superior to traditional methods. How to use deep learning technology to study plant diseases and pests identification has become a research issue of great concern to researchers.

[2]Amina Khatra.(2019)The presented work presents a color based segmentation techniques for extraction of yellow rust in whet crop images. Accurate segmentation of yellow rust in wheat crop images is very part of assessment of disease penetration into the wheat crop. And in turn to take the necessary preventive action for minimizing the crop damage. The jpeg images acquired from CCD camera are read into the matlab tool and a color-based segmentation algorithm is performed to segment the yellow rust. The segmentation of color is performed base on k-means algorithm.

[3]Ashwini T Sapka et al.(2018)The feature extraction technique plays a very critical and crucial role in automatic leaf disease diagnosis system. Many different feature extraction techniques are used by the researchers for leaf disease diagnosis which includes colour, shape, texture, HOG, SURF and SIFT features. Recently Deep Learning is giving very promising results in the field of computer vision. In this manuscript, two feature extraction techniques are discussed and compared. In first approach, the Gray Level Covariance Matrix is used which extracts 12 texture features for diagnosis purpose. In second appraoch, the pretrained deep learning model, Alexnet is used for feature extraction purpose. There are 1000 features extracted automatically with the help of this pretrained model.

[4]Behera S.K. et al.(2018) Disease Classification and Grading of Orange Us-

ing Machine Learning and Fuzzy Logic Proceedings of the 2018 IEEE International Conference on Communication and Signal Processing (ICCSP); Chennai, India. 3–5 April 2018 Challenges in disease clarification and grading of oranges involve color and texture variations, the need for consistent lighting conditions, and difficulties in accurately differentiating between various disease severity levels.

[5] Dubey S et al.(2012) Detection and Classification of Apple Fruit Diseases Using Complete Local Binary Patterns; Proceedings of the 2012 3rd International Conference on Computer and Communication Technology; Allahabad, India. 23–25 November 2012 Complete binary pattern-based disease detection may struggle with complex patterns, high computational costs, and sensitivity to image quality. Adapting to diverse disease manifestations and variations remains a challenge.

[6]Ishaan Dawar et al.(2023) Disease Detection in Fruits Using Image Processing; Proceedings of the International Conference on Inventive Computation Technologies (ICICT); Coimbatore, India. Drawbacks of fruit disease detection with image processing include limited accuracy due to variations in fruit appearance, dependence on high-quality images, and challenges in real-time processing for large-scale applications.

[7]Malathy S et al.(2021)There are many types of diseases which are present in plants. To detect these diseases, patterns are required to recognize them. There are many types of pattern recognition algorithm which gives detection of disease with accuracy. Image processing Techniques for Wheat Disease Detection most important research areas in computer science for last few decades. Based on literature review, we conclude that the engineering and research community is doing lot of work on Wheat disease detection, but the application of this techniques to solve practical agricultural This paper presents a survey on SVM Classifier method that use digital image processing techniques to detect, quantify and classify plant diseases from digital images in the visible spectrum

[8]Paul et al.(2023) It reviews, and summaries various techniques used for classifying and detecting various bacterial, fungal and viral wheat leaf diseases. The classification techniques help in automating the detection of wheat leaf diseases and categorizing them centered on their morphological features. It focuses on



identifying the wheat leaf diseases with CNN as classifier. It is also intended to focus on increasing the recognition rate and classification accuracy of severity of leaf diseases by using hybrid algorithms.

[9]Pallavi Pandey et al.(2023)Drawbacks of deep learning in plant disease detection include the need for large labeled datasets, and susceptibility to variations in environmental conditions affecting model generalization.

[10]Ruchi Rani et al.(2023) Role of Artificial Intelligence in Agriculture An Analysis and Advancements With Focus on Plant Diseases, IEEE Access, AI drawbacks include biases in algorithms, lack of transparency, potential job displacement, ethical concerns, and dependence on data quality. Striking a balance between innovation and responsible deployment is crucial.

[11]Rumpf T et al.(2010) Early detection and classification of plant diseases with Support Vector Machines based on hyperspectral reflectance. Comput Electron Agric. Support vector machines in crop disease classification face challenges like sensitivity to parameter tuning, limited scalability for large datasets, and difficulty handling overlapping classes, impacting their adaptability.

[12]Ramesh S et al.(2018) Plant Disease Detection Using Machine Learning; Proceedings of the 2018 International Conference on Design Innovations for 3Cs Compute Communicate Control Machine learning for plant disease detection faces challenges like the requirement for labeled data, sensitivity to environmental changes, potential misclassification, and difficulties in addressing new, unseen diseases effectively

[13]Geetanjali Babbar et al.(2018)Various Plant Disease Detection Using image Processing Methods.Identification of plant leaf diseases is the preventive measure for the loss happened in the yield and the overall agriculture crop quantity. Basically, the studies of the plant diseases are defined by visualizing and observing patterns observed and engraved on the leaves. So, the disease detection of any plant prior to any hazardous impact becomes very crucial factor for viable agriculture. However, it is so difficult to detect, monitor and derive conclusions from the plant

leaf diseases manually because, the costs emerging in the process demands huge amount of workdone, energy, expertize and last but not least the processing time.

[14]Sumit Nema et al.(2020)Android Application of Wheat Leaf Disease Detection and Prevention using Machine Learning. Crop quality and production plays an important role in agriculture and farmer's life. Famer's income highly depends on crop quantity and quality in India. Wheat is the main crop in India. Wheat leaves diseases majorly affect the production rate as well as farmer's profits. An android application has designed to detect the wheat plant leaf diseases in this work. Machine learning methods are easily applied and capable to quick recognizes these diseases. Simulation results show the effectiveness of the proposed method.

[15]Varun Jindal et al.(2023) Improving Agricultural Efficiency Severity-based Diagnosis of Bottle Gourd Leaf Diseases Using Federated Learning CNN. Challenges in enhancing agriculture efficiency involve high initial investment costs, resistance to technology adoption, lack of skilled workforce, and potential environmental impact.

## Chapter 3

# PROJECT DESCRIPTION

### 3.1 Existing System

The existing systems for crop disease detection vary widely in terms of technological sophistication and deployment scale. Traditional methods often rely on manual inspection by agronomists or extension workers who visually assess crop health in the field. While these methods are effective to some extent, they are labor-intensive, time-consuming, and susceptible to human error. Another common approach involves the use of handheld devices or cameras to capture images of diseased crops, which are then analyzed using basic image processing techniques or rule-based algorithms. While these methods provide rapid feedback to farmers, they often lack the accuracy and scalability required for large-scale disease monitoring across diverse agricultural landscapes.

### 3.2 Proposed System of Crop Disease Detection

The proposed system for crop disease identification using machine learning offers a multitude of advantages poised to revolutionize agricultural practices. By leveraging deep learning models like Convolutional Neural Networks, LeNet-5 the system can swiftly and accurately identify various crop diseases from images, aiding farmers in early detection and precise intervention. These models excel in pattern recognition, enabling them to distinguish between healthy and diseased crops across diverse environmental conditions, leading to timely responses and improved crop management strategies.

One of the key advantages of employing deep learning in crop disease identification is its potential to significantly enhance agricultural productivity and sustainability. However, the potential savings from reducing crop losses due to accurate disease identification and optimizing resource usage could outweigh these initial in-

vestments. Timely disease detection can lead to minimized pesticide and fungicide use, lowering input costs, and enhancing environmental sustainability, thereby offering substantial economic advantages to farmers.

### **3.3 Feasibility Study**

#### **3.3.1 Economic Feasibility**

Assessing the economic feasibility of employing machine learning for crop disease identification involves analyzing both initialExisting systems for crop disease identification using machine learning encounter several challenges. One of the primary limitations is the scarcity and quality of datasets containing diverse, well-labeled images representing various crop diseases across different stages and environmental conditions. This scarcity impedes the development of robust models, affecting their accuracy and generalizability. Additionally, the interpretability of machine learning models, particularly complex ones like deep neural networks, poses a challenge as these models often function making it difficult to understand the reasoning behind their predictions. This lack of transparency can hinder user trust and comprehension, crucial factors for adoption in practical agricultural settings. Furthermore, issues related to environmental variability, computational resources for training complex models, and the integration of these technologies within existing farming practices present significant hurdles for effective implementation.

#### **3.3.2 Technical Feasibility**

The technical feasibility of employing machine learning for crop disease identification hinges on several critical factors. Firstly, it involves the availability and quality of datasets comprising labeled images representing various crop diseases across different stages and environmental conditions. A robust dataset is essential for training machine learning models effectively. Moreover, the computational requirements for training and deploying these models need consideration. Deep learning models like Convolutional Neural Networks commonly used for image classification tasks require significant computational resources. Ensuring access to suitable hardware or cloud services capable of handling the computational demands is crucial for the feasibility of the system. Another key aspect is the scalability and efficiency of the machine learning models for real-time disease identification. The system's ability to

process and analyze images rapidly, especially in agricultural settings where quick decisions are crucial, is vital. Additionally, the integration of the technology into practical farming environments, including considerations for connectivity in remote areas and compatibility with existing infrastructure, plays a pivotal role in determining its technical feasibility. Ensuring that the system can operate effectively in various environmental conditions and adapt to different crops and disease patterns further contributes to its technical viability for widespread adoption in agricultural practices.

### **3.3.3 Social Feasibility**

The social feasibility of implementing crop disease identification using machine learning involves considerations beyond technological aspects, focusing on societal acceptance, usability, and the impact on farming communities. One critical factor is the accessibility and usability of the technology among farmers, agricultural workers, and local communities. The system should be user-friendly, intuitive, and easily integrated into existing farming practices to ensure acceptance and adoption. Providing adequate training, support, and guidance on utilizing the technology is essential for empowering users and overcoming potential barriers to acceptance.

Moreover, addressing ethical concerns and ensuring equitable access to the technology across different socioeconomic groups and geographic regions is crucial for social feasibility. Engaging with local communities, understanding their needs, and considering cultural factors in the deployment of machine learning-based solutions for crop disease identification is necessary. Building trust and demonstrating the tangible benefits of the technology, such as increased crop yield, reduced environmental impact, and improved livelihoods, are pivotal for social acceptance and the successful integration of these systems within farming communities. Overall, social feasibility necessitates a holistic approach that accounts for societal perceptions, community engagement, and the technology's alignment with the needs and values of the farming population.

## **3.4 System Specification**

### **3.4.1 Hardware Specification**

- CPU: Intel Core i5 or AMD Ryzen 5

- GPU: NVIDIA GTX 1060 or equivalent (6GB VRAM)
- RAM: 16GB DDR4
- Storage: 500GB SSD
- OS: Windows 10 or Linux (Ubuntu, CentOS)
- Deep Learning Framework: TensorFlow, PyTorch (with GPU support)

### **3.4.2 Software Specification**

- Operating System: Windows 10 (64-bit) or Linux (for compatibility and driver support)
- Deep Learning Framework: TensorFlow or PyTorch (GPU-accelerated deep learning development)
- Programming Language: Python (3.6 or later for framework compatibility and libraries)
- Development Environment: Jupyter Notebook, Visual Studio Code, or PyCharm (code editing, debugging, visualization)
- Additional Libraries: OpenCV (image processing), NiBabel (neuroimaging data), Scikit-learn (machine learning utilities), Matplotlib/Seaborn (visualization)
- Version Control: Git (code management and collaboration)
- Cloud Computing Platforms (optional): Google Colab, AWS, Azure (pre-configured environments and hardware)
- Data Handling Tools: BraTS dataset tools or custom scripts (for loading and preprocessing)
- Model Evaluation Metrics: Dice similarity coefficient, sensitivity, specificity (brain tumor segmentation performance assessment)

### **3.4.3 Standards and Policies**

#### **Anaconda Prompt**

Anaconda prompt is a type of command line interface which explicitly deals with

the ML( MachineLearning) modules.And navigator is available in all the Windows,Linux and MacOS.The anaconda prompt has many number of IDE's which make the coding easier. The UI can also be implemented in python.

**Standard Used: ISO/IEC 27001**

### **Jupyter**

It's like an open source web application that allows us to share and create the documents which contains the live code, equations, visualizations and narrative text. It can be used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning.

**Standard Used: ISO/IEC 27001**

### **Google Colab**

Google Colab is a free playground for anyone wanting to explore Python in the cloud. No downloads, just open your browser and start coding! It provides access to powerful computing resources like GPUs and TPUs, making it ideal for machine learning, data analysis, or simply learning the ropes. You can even collaborate on projects with friends in real-time, sharing and editing notebooks like magic. While resources are shared and temporary, it's perfect for quick experiments and learning bursts.

**Standard Used: ISO/IEC 27001**

## Chapter 4

# METHODOLOGY

### 4.1 Proposed Architecture of Crop Disease Detection

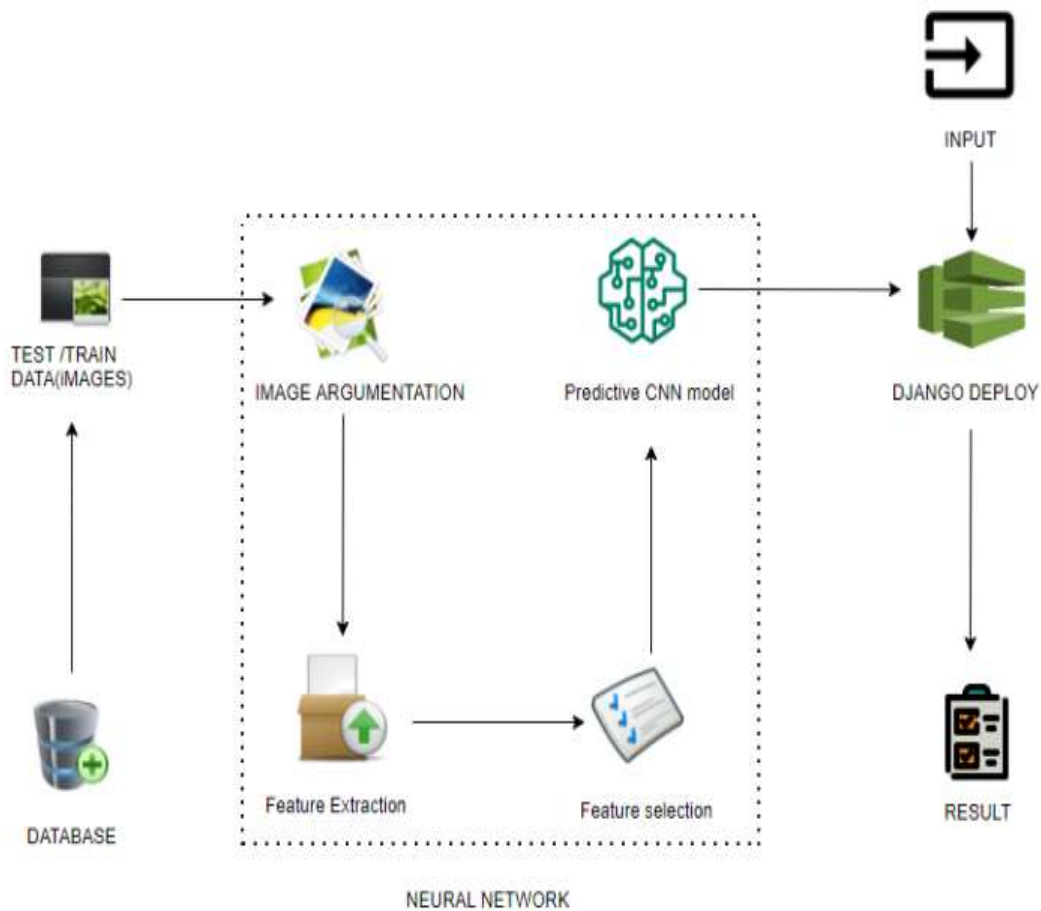


Figure 4.1: Crop Disease Detection Architecture Diagram

The above Figure 4.1 describes the working of crop disease detection. Initially it takes data and train and test the dataset by using LeNet-5 architecture and next extract the features of the image by using different layers and predict and classify the images after completing the modeling deploy using Django framework in that contains user interface and takes input from the user and classifies whether the disease contains or not.



## **1.Image Data Generator**

It is that rescales the image, applies shear in some range, zooms the image and does horizontal flipping with the image. This Image Data Generator includes all possible orientation of the image.

## **2.Training Process**

Traindatagen flow from directory is the function that is used to prepare data from the train dataset directory Target size specifies the target size of the image. Test datagen.flow from directory is used to prepare test data for the model and all is similar as above. fit generator is used to fit the data into the model made above, other factors used are steps per epochs tells us about the number of times the model will execute for the training data.

## **3.Arugments**

Rate Float between 0 and 1. Fraction of the input units to drop. noise shape 1D integer tensor representing the shape of the binary dropout mask that will be multiplied with the input. For instance, if your inputs have shape (batch size, timesteps, features) and you want the dropout mask to be the same for all timesteps, you can use noise shape=(batch size, 1, features).seed A Python integer to use as random seed.

## **4.Epochs**

It tells us the number of times model will be trained in forward and backward pass.

## **5.Validation process**

Validation data is used to feed the validation/test data into the model. Validation steps denotes the number of validation/test samples.

## **6.Input layer**

The input layer is the first layer of a convolutional neural network (CNN). It is responsible for receiving the input image. The input layer is typically a two-dimensional array of neurons, where each neuron corresponds to a pixel in the input image.

## **7. Filter (also known as Convolutional Kernel)**

A filter (also known as a kernel) is a small matrix of weights that is applied to the input image in a convolutional layer. The filter slides over the image, computing the dot product of its weights with the corresponding elements of the image at each position. The result of the dot product is the activation value for that position in the feature map. The filter is responsible for detecting specific features in the input image. The size of the filter determines the size of the receptive field, which is the

area of the input image that the filter is sensitive to. The weights of the filter are learned during training.

## **8. Pooling**

A downsampling technique that reduces the spatial dimensions of feature maps while preserving key information. Common types include max pooling (taking the maximum value within a window) and average pooling (taking the average value). Makes the model less sensitive to small translations and distortions in the input. Pooling layers are typically inserted between convolutional layers. There are two common types of pooling: max pooling and average pooling. Max pooling takes the maximum value within a window of the feature map. Average pooling takes the average value within a window of the feature map. Pooling layers help to make the CNN more robust to small variations in the input image, and they also help to reduce the number of parameters in the network.

## **9. CNN Layers (Convolutional Neural Network Layers)**

The building blocks of CNNs, responsible for feature extraction. Typically consist of a sequence of convolutional layers followed by pooling layers. Learn patterns and create increasingly abstract representations of the input data. It is responsible for extracting features from the input image. A convolutional layer consists of a set of filters, which are applied to the input image to produce feature maps. The filters are small matrices of weights that slide over the image, computing the dot product of their weights with the corresponding elements of the image at each position. The result of the dot product is the activation value for that position in the feature map.

## **10. Pooled Feature Maps**

Output of a convolutional layer, representing the activations of different filters applied to the input. Each feature map captures a different aspect or pattern in the image.

## **11. Fully Connected Layer**

A traditional neural network layer where each neuron is connected to every neuron in the previous layer. Typically used in the final layers of a CNN to make class predictions or generate outputs based on the extracted features.

## 4.2 Design Phase

### 4.2.1 Data Flow Diagram of Crop Disease Detection

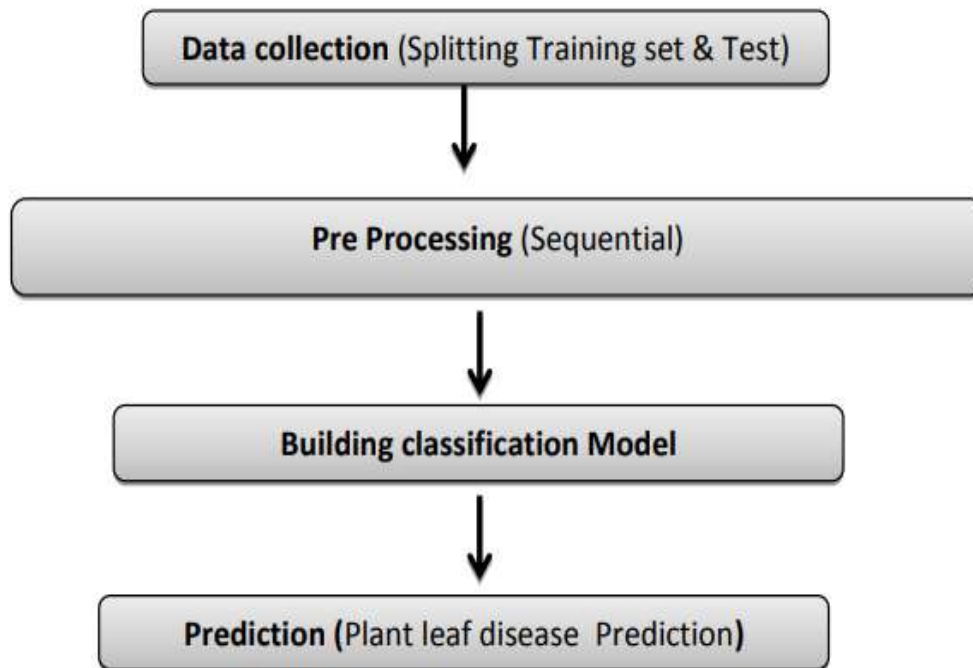


Figure 4.2: Data Flow Diagram for Crop Disease Detection

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). The Figure4.2 shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model. Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top down approach to Systems Design.

#### 4.2.2 Use Case Diagram of Crop Disease Detection

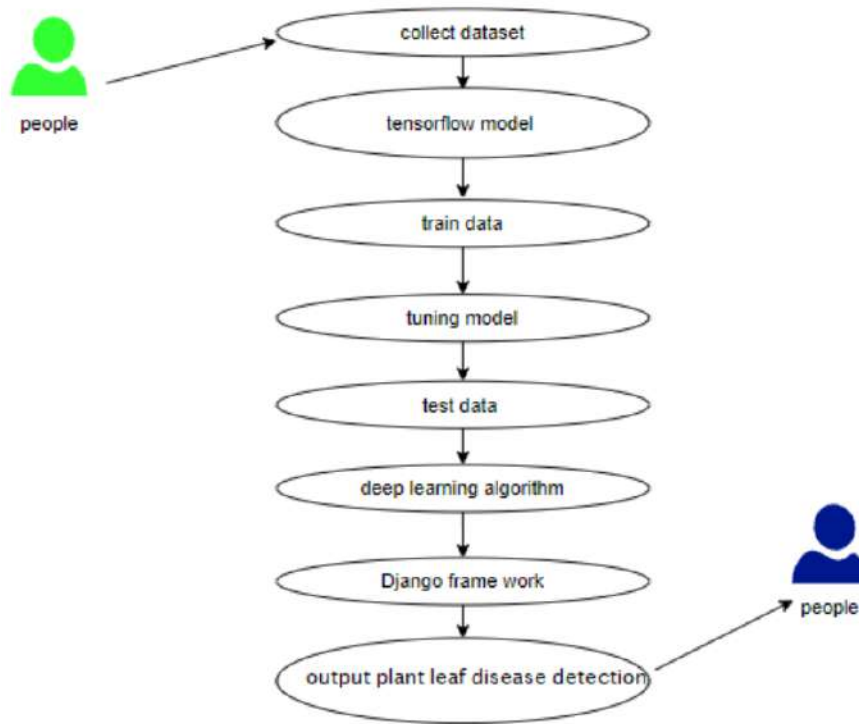


Figure 4.3: Use Case Diagram of Crop Disease Detection

The above Figure4.3 is considered for high level requirement analysis of a system. So, when the requirements of a system are analyzed the functionalities are captured in use cases. So, it can say that uses cases are nothing but the system functionalities written in an organized manner

### 4.2.3 Class Diagram of Crop Disease Detection

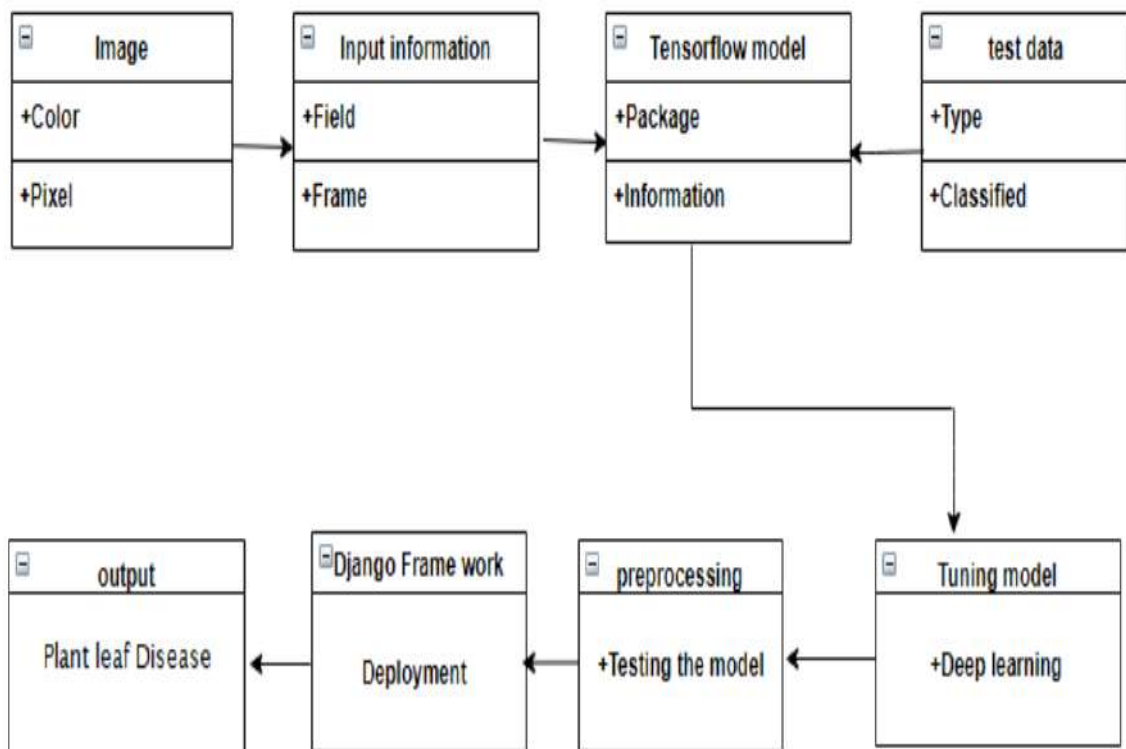


Figure 4.4: Class Diagram of Crop Disease Detection

The above Figure 4.4 is a graphical representation of the static view of the system and represents different aspects of the application. So, a collection of class diagrams represent the whole system. The name of the class diagram should be meaningful to describe the aspect of the system. Each element and their relationships should be identified in advance. Responsibility (attributes and methods) of each class should be clearly identified for each class. Minimum number of properties should be specified and because, unnecessary properties will make the diagram complicated. Use notes whenever required to describe some aspect of the diagram and at the end of the drawing it should be understandable to the developer/coder. Finally, before making the final version, the diagram should be drawn on plain paper and rework as many times as possible to make it correct.

#### 4.2.4 Sequence Diagram of Crop Disease Detection

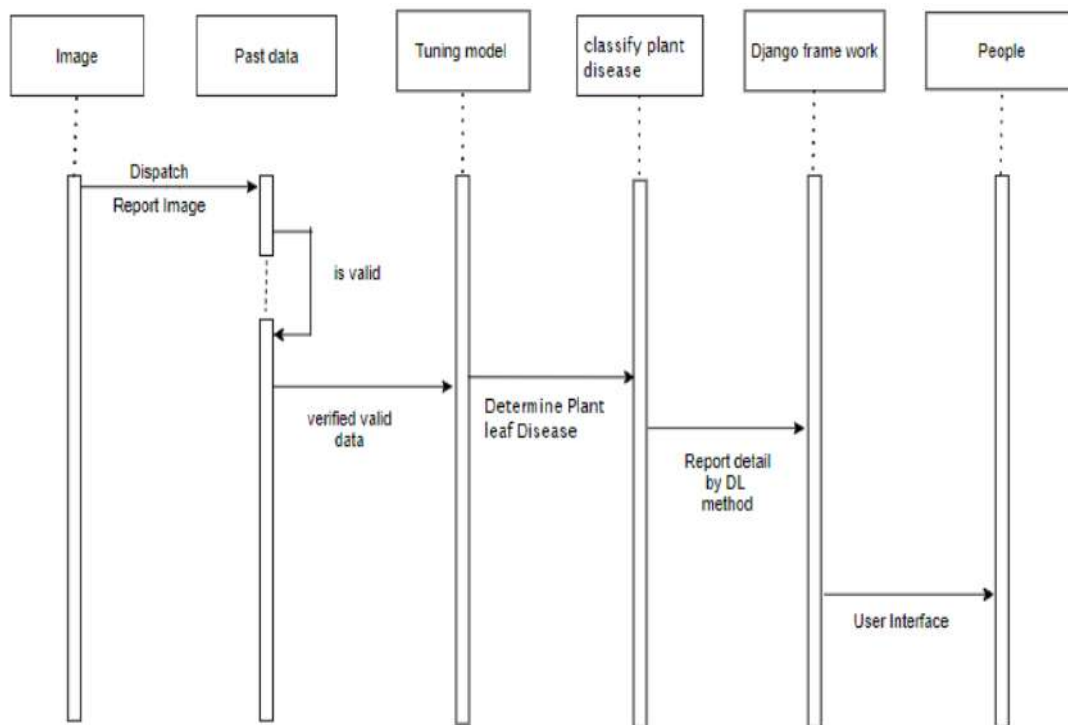


Figure 4.5: Sequence Diagram of Crop Disease Detection

Sequence diagram 4.5 shows model flow of logic within your system in a visual manner, enabling you both to document and validate your logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behavior within your system. Other dynamic modelling techniques include activity diagramming, communication diagramming, timing diagramming, and interaction overview diagramming. Sequence diagrams, along with class diagrams and physical data models are in my opinion the most important designlevel models for modern business application development.

#### 4.2.5 Collaboration diagram of Crop Disease Detection

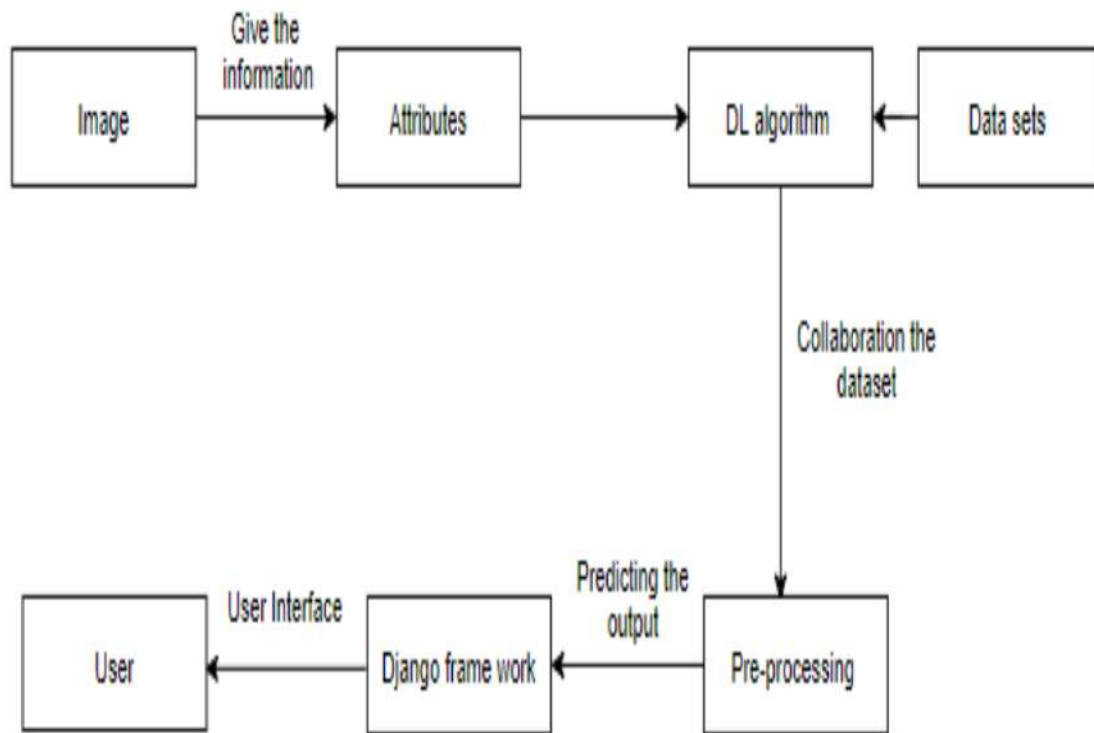


Figure 4.6: Collaboration Diagram of Crop Disease Detection

The above Collaboration Diagram shows crop disease detection using deep learning involves multiple stakeholders working synergistically to achieve common goals. At the core of this collaboration diagram are agricultural experts, including agronomists, plant pathologists, and extension workers, who provide domain knowledge and ground truth annotations for training datasets. Their expertise is crucial for accurately identifying crop diseases, understanding their dynamics, and validating the performance of deep learning models. Alongside them are data scientists and machine learning engineers responsible for developing and fine-tuning deep learning algorithms. Their role involves preprocessing image data, designing neural network architectures, and optimizing model hyperparameters to achieve high accuracy and robustness in disease detection.

#### 4.2.6 Activity Diagram of Crop Disease Detection

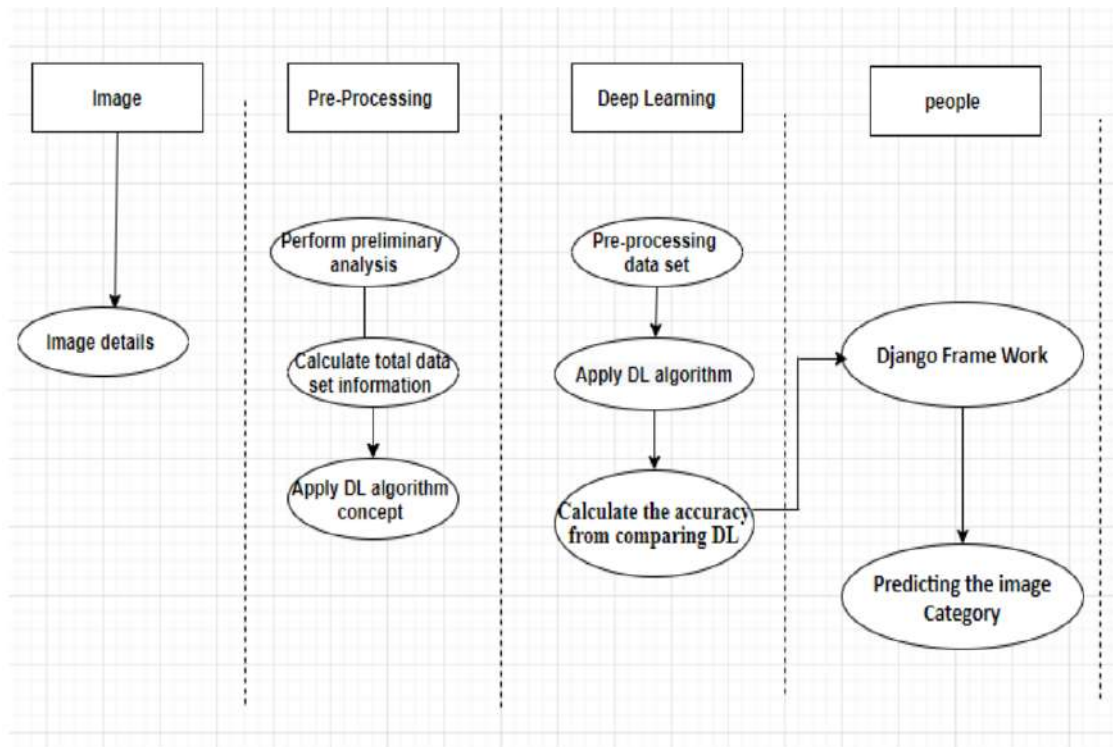


Figure 4.7: Activity Diagram of Crop Disease Detection

Activity Diagram Figure 4.7 is a particular operation of the system. Activity diagrams are not only used for visualizing dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in activity diagram is the message part. It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flow chart. Although the diagram looks like a flow chart but it is not. It shows different flow like parallel, branched, concurrent and single.

### 4.3 Algorithm & Pseudo Code

#### 4.3.1 CNN Algorithm

**Step1:-Data Collection:** Gather a diverse dataset of images representing both healthy crops and various disease symptoms across different crop types and growth stages.

**Step2:-Data Preprocessing:** Clean and preprocess the collected images to ensure consistency in size, resolution, and color balance.



**Step3:-Model Selection:** Choose an appropriate deep learning architecture for crop disease detection, such as convolutional neural networks

**Step4:-Model Training:** Split the annotated dataset into training, validation, and test sets. hyperparameters as needed to prevent overfitting.

**Step5:-Model Evaluation:** Evaluate the trained model's performance using metrics such as accuracy, precision, recall, and F1 score on the test set.

**Step6:-Deployment:** Deploy the trained model into a production environment for real-time inference on new images.

**Step7:-User Interface:** Develop a user-friendly interface for farmers or agricultural stakeholders to interact with the disease detection system. Provide features such as uploading images

#### **4.3.2 Pseudo Code of Crop Disease Detection**

**Step 1:** Data Collection images = collect images()

**Step 2:** Data Preprocessing preprocessed images = preprocess images(images)  
annotated images = annotate images(preprocessed images)

**Step 3:** Model Selection model = select model()

**Step 4:** Model Training train set, validation set, test set = split dataset(annotated images)  
trained model = train model(model, train set, validation set)

**Step 5:** Model Evaluation evaluation metrics = evaluate model(trained model, test set)

**Step 6:** Deployment deploy model(trained model)

**Step 7:** User Interface user interface = create user interface()

**Step 8:** Monitoring and Feedback while True: captured image = capture image()  
preprocessed image = preprocess image(captured image) disease prediction = infer disease(trained model, preprocessed image) display results(disease prediction) collect feedback()

### **4.4 Module Description**

#### **4.4.1 Training the Dataset Module**

**Title:** Training the Dataset Module:

**Description:** Gathers and analyzes diverse data sources, including environmental

conditions, historical crop data, and disease prevalence. Utilizes statistical models and machine learning algorithms to identify patterns, correlations, and potential outbreaks.

**Convolutional layers** Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.

**Pooling layers** Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network.

**Dense or Fully connected layers:** Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP.

#### 4.4.2 Step1:Data Collection

To classify different plant diseases, we plan to design a deep learning system so that a person without expertise in software should also be able to use it easily. The proposed system is made to predict plant diseases using the leaves as an identifying factor. It explains the analysis of our methodology along with some of the feature engineering of the data. A large number of images is collected for each disease and is classified into database images and input images. The primary attributes of the leaves that are important are the shape and texture-oriented features. The figure provided below gives us an insight into the basic principle of our system along with an idea about how the system works.

#### 4.4.3 Step2:Data Processing

The dataset is preprocessed such as Image reshaping, resizing and conversion to an array form. Similar processing is also done on the test image. A dataset consisting of about 10 different class of leaf, out of which any image can be used as a test image for the software. The train dataset is used to train the model (CNN) so that it can identify the test image and the disease it has CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D, and MaxPooling2D. After the model is trained successfully, the software can identify the Plant leaf disease prediction image

contained in the dataset. After successful training and preprocessing, comparison of the test image and trained model takes place to predict the Sign language.

#### **4.4.4 Applying LeNet-5 Architecture**

**Title:**Applying LeNet-5 Architecture: LeNet was one among the earliest convolutional neural networks which promoted the event of deep learning. After innumerable years of analysis and plenty of compelling iterations, the end result was named LeNet.

**Introduction:** Crop diseases pose significant challenges to global food security and agricultural sustainability. Timely detection and management of these diseases are critical for minimizing crop losses and ensuring food supply. Traditional methods of disease diagnosis often rely on visual inspection by experts, which can be subjective, time-consuming, and labor-intensive. With the advent of deep learning technologies, there is a growing interest in leveraging convolutional neural networks (CNNs) for automated crop disease detection. LeNet 5, with its proven track record in image recognition tasks, presents a compelling framework for addressing this pressing agricultural issue.

**Architecture of LeNet-5** LeNet-5 CNN architecture is made up of 7 layers. The layer composition consists of 3 convolutional layers, 2 subsampling layers and 2 fully connected layers.

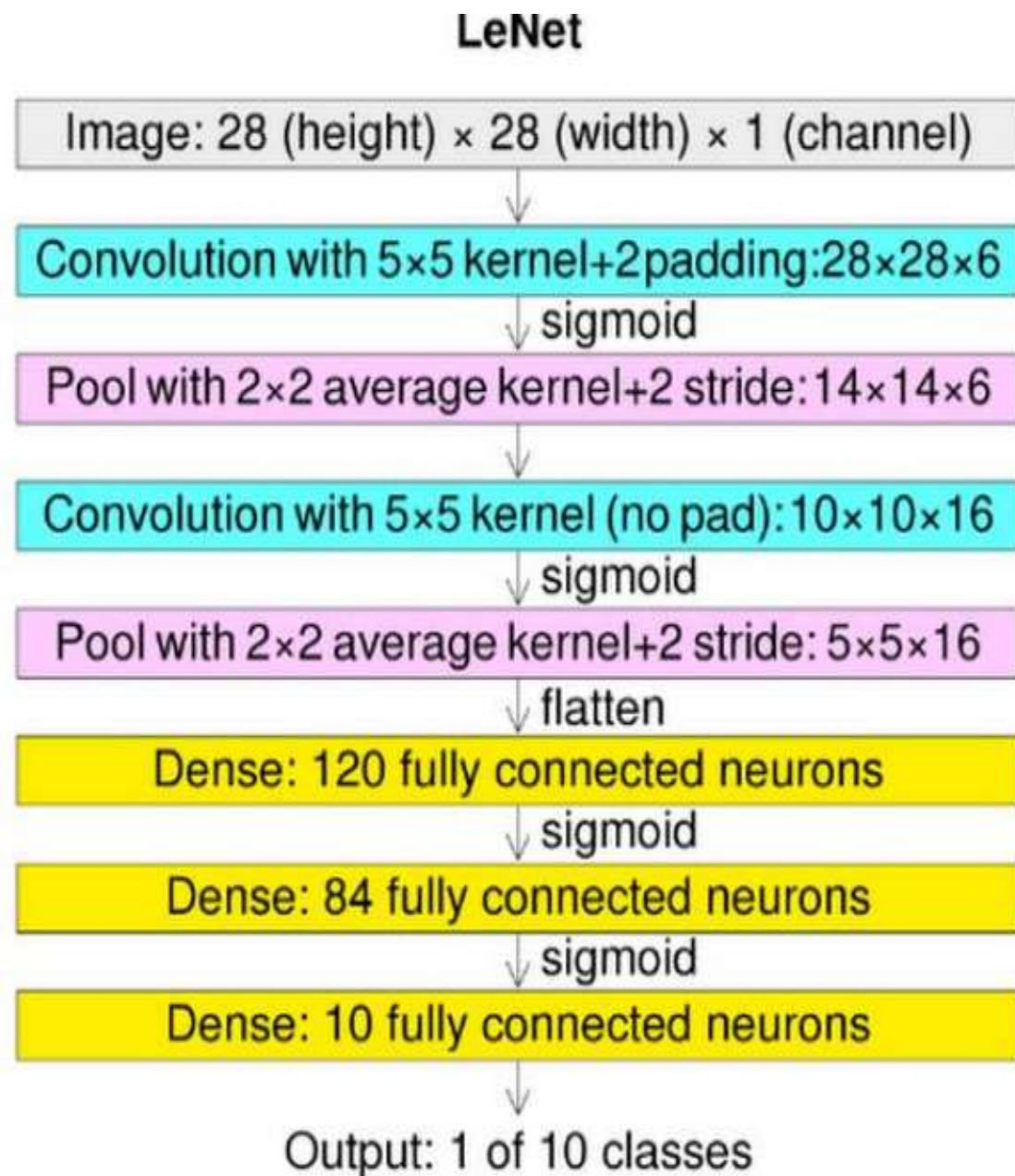


Figure 4.8: LeNet5 Architecture

**Architecture Overview** LeNet-5 combines convolutional, subsampling (pooling), and fully connected layers. It uses specific connectivity patterns to extract distinctive features from input data efficiently. The architecture aims to learn hierarchical representations from images.

**Input Layer** The input layer of LeNet 5 receives input images of crops captured using various imaging devices such as digital cameras, drones, or satellite sensors. These images may exhibit variations in lighting conditions, camera angles, and background clutter, necessitating preprocessing steps such as resizing, normalization, and augmentation to ensure consistency and quality.

**Convolutional layers** Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels. LeNet-5 comprises three convolutional layers. These layers are responsible for extracting local features from the input image. Each convolutional layer applies a set of learnable filters (also known as kernels) to the input data. These filters slide across the image, detecting patterns such as edges, corners, and textures.

**Pooling layers** Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network. Pooling layers play a crucial role in reducing the spatial dimensions of the feature maps while retaining essential information. LeNet-5 uses average pooling (also known as sub-sampling) to downsample the feature maps. Average pooling computes the average value within a small window (usually 2x2 or 3x3) and replaces the original values with these averages. This process reduces the computational load and helps prevent overfitting.

**Dense or Fully connected layers** Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP. LeNet-5 includes two fully connected layers (also known as multi-layer perceptrons). These layers act as the classifier. They take the flattened feature vector and learn to make predictions based on the learned features. The final fully connected layer produces class probabilities.

**Activation Function** LeNet-5 uses the tanh (hyperbolic tangent) activation function. Tanh squashes the output values between -1 and 1, making it suitable for neural networks. However, modern architectures often use ReLU (Rectified Linear Unit) activations due to their faster convergence and sparsity properties.

**Batch Normalization** Batch normalization layers are incorporated after each convolutional layer to improve the stability and convergence of the training process. By normalizing the activations of each layer, batch normalization mitigates the internal covariate shift problem and accelerates the optimization process. This leads to faster convergence and better generalization performance of LeNet 5.

**Flattening Convolutional Layer** After the pooling layers, LeNet-5 introduces a flattening convolutional layer. This layer reshapes the 2D feature maps into a 1D vector.

The flattened vector serves as the input to the fully connected layers.

**Sparse Connectivity** LeNet-5's sparse connectivity reduces the number of connections between layers. Instead of connecting every neuron to every other neuron, it selectively connects only specific neurons. This design choice reduces memory requirements and accelerates training.

**Output Layer** The final layer of LeNet 5 is a softmax layer, which outputs the probabilities of each class (e.g., different crop diseases) based on the learned features. The class with the highest probability is considered the predicted disease for a given input image.

**Forward Propagation** During each training iteration, a batch of training examples is forward-propagated through the network to compute the predicted outputs. The output probabilities are compared to the ground truth labels using a loss function such as categorical cross-entropy, quantifying the disparity between the predicted and actual classes.

**Backpropagation** After computing the loss, gradients are backpropagated through the network to update the model parameters using optimization algorithms such as stochastic gradient descent (SGD) or Adam. The gradients indicate the direction and magnitude of parameter adjustments required to minimize the loss function.

**Parameter Updates** Using the computed gradients, the model parameters are updated iteratively to minimize the loss function. Regularization techniques such as dropout or weight decay may be employed to prevent overfitting and improve generalization performance. The training process continues for multiple epochs until convergence or a predefined stopping criterion is reached.

**Evaluation Metrics and Performance** Once training is complete, LeNet 5 is evaluated on a separate validation set to assess its performance. Evaluation metrics such as accuracy, precision, recall, and F1-score are computed to measure the model's ability to correctly classify healthy and diseased crops. Additionally, confusion matrices and receiver operating characteristic (ROC) curves may be used to visualize the model's performance across different classes.

**Applications in Agriculture** The application of LeNet 5 in crop disease detection has significant implications for agriculture and food security. By automating the process of disease diagnosis, farmers can identify and mitigate crop diseases more effectively, leading to improved yields and reduced economic losses. Additionally, early detection of diseases enables timely intervention strategies such as targeted pesticide application or crop rotation, minimizing the spread of pathogens.

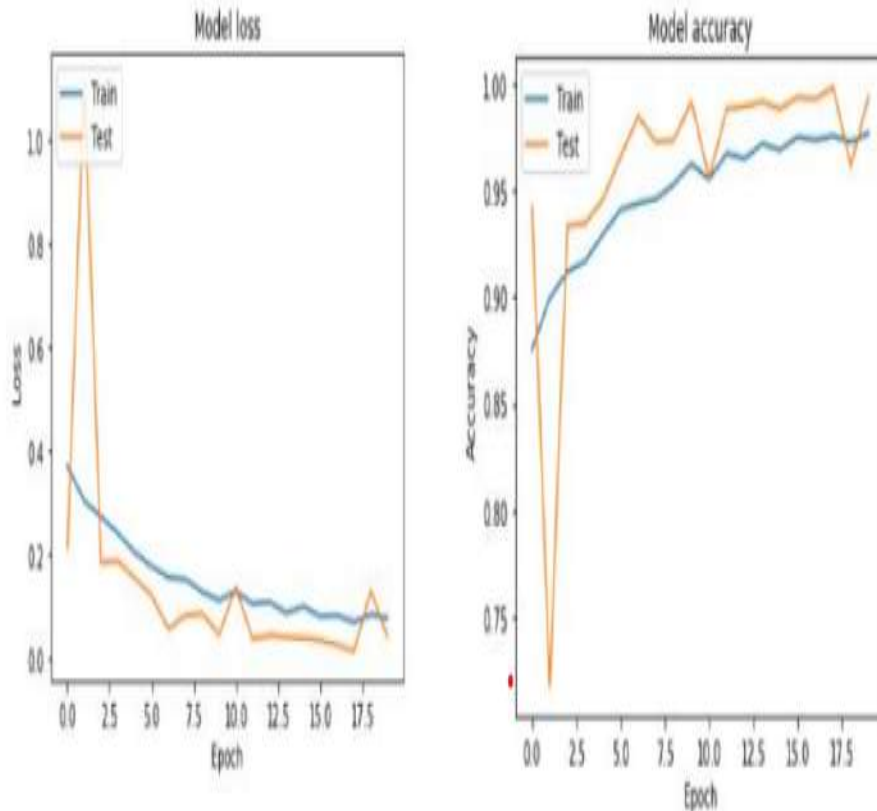


Figure 4.9: LENet-5 Metrics Graph

## 4.5 Steps to execute/run/implement the project

### 4.5.1 Step1:Installing Anaconda

To install Anaconda, begin by visiting the Anaconda website and downloading the appropriate distribution for your operating system—whether it’s Windows, macOS, or Linux. After the download completes, run the installer executable file. Follow the installer instructions diligently as it guides you through the installation process, prompting you to review the license agreement, select an installation location, and optionally add Anaconda to your system PATH. During installation, you’ll choose between installing Anaconda for just your user or for all users on the system, based on your requirements. Proceed with the installation by clicking the “Install” button and patiently wait for the process to complete, which may take a few minutes depending on your system. Once the installation finishes, you may be prompted to click “Next” or “Finish” to finalize the process. To verify that Anaconda was installed correctly, open a terminal or command prompt and type ‘conda –version‘,

which should display the installed version. Optionally, it's advisable to update Anaconda after installation to ensure you have the latest packages and bug fixes; this can be done by running 'conda update conda' in the terminal or command prompt.

#### 4.5.2 Step2:Download Tensorflow and Keras

In a Jupyter Notebook, installing TensorFlow and Keras is a straightforward process using the !pip command within code cells. Firstly, to install TensorFlow, you can execute !pip install tensorflow. This command installs the latest version by default, but you can specify a particular version if needed, such as !pip install tensorflow==2.7.0. Subsequently, you can install Keras, which is integrated within TensorFlow, by running !pip install keras. As with TensorFlow, you can specify a version for Keras installation, like !pip install keras==2.7.0. Ensure to execute these commands in separate cells within your Jupyter Notebook. Once completed, TensorFlow and Keras will be installed and accessible for utilization within your notebook environment.

#### 4.5.3 Step 3:Detection of Crop Disease with LeNet-5

**Convolutional layers:** Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user specified parameters are in the network. The most important parameters are the number of kernels and the size of the kernels.

**Pooling layers:** Pooling layers are similar to convolutional layers, but they perform a specific function such as max pooling, which takes the maximum value in a certain filter region, or average pooling, which takes the average value in a filter region. These are typically used to reduce the dimensionality of the network.

**Dense or Fully connected layers:** Fully connected layers are placed before the classification output of a CNN and are used to flatten the results before classification. This is similar to the output layer of an MLP.



# Chapter 5

## IMPLEMENTATION AND TESTING

### 5.1 Input and Output

#### 5.1.1 Crop Disease Identification Input Design



Figure 5.1: Input Design for Crop Diseases Identification from Plant Village Dataset

### 5.1.2 Early Blight Late Blight Output Design

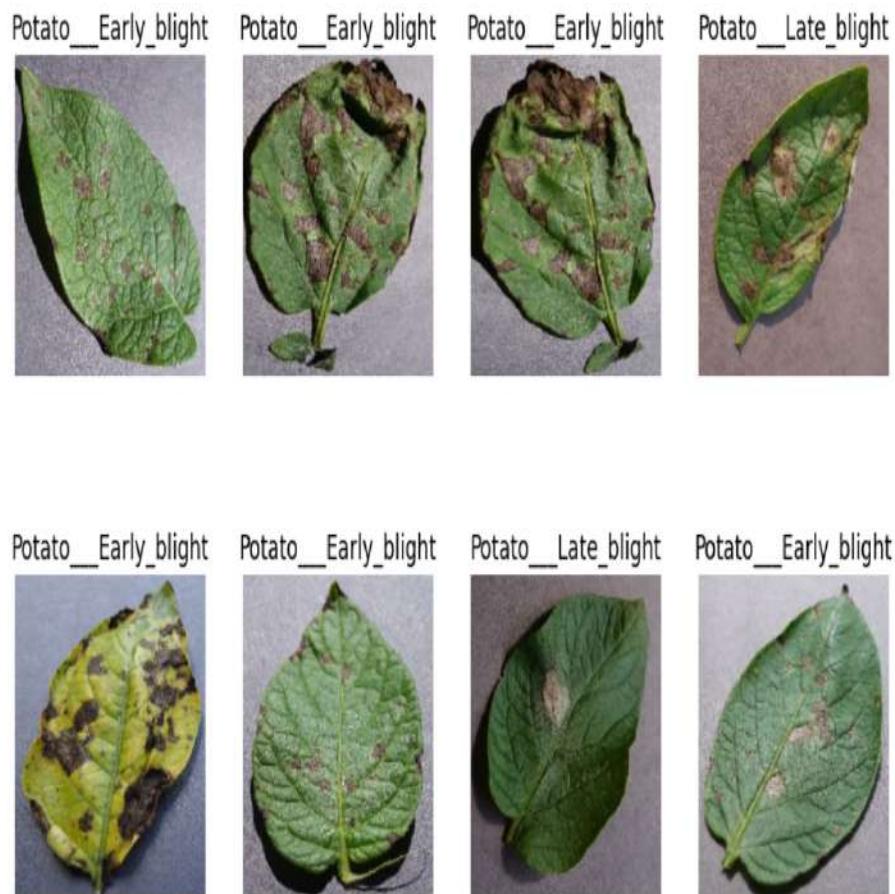


Figure 5.2: Output Design for Crop Diseases Identification

## 5.2 Testing

### 5.3 Types of Testing

#### 5.3.1 Unit testing

Unit testing focuses on individual components of the crop disease detection system. Here are some examples of unit tests Image Preprocessing Test functions that resize images, convert color spaces, and apply noise reduction. Verify if these functions handle invalid inputs gracefully. Feature Extraction Test functions that extract features like color histograms, textures, or shapes from images. Ensure these functions produce expected outputs for known input images containing specific diseases. Deep Learning Model Test the model's ability to classify healthy vs. diseased crops on a small dataset of labeled images. Verify the model performs well on different

disease severities or variations.

### **5.3.2 Integration testing**

Integration Testing for Crop Disease Detection Systems Integration testing focuses on how well different components work together. Here are some examples of integration tests Data Pipeline Test if the entire image processing pipeline (preprocessing, feature extraction, model prediction) functions smoothly with real-world data. User Interface Test if user input (e.g., uploading an image) gets processed correctly and the disease prediction is displayed accurately. Database Integration Test if the system interacts with a database to store historical data or retrieve past diagnoses. Ensure data is saved and retrieved correctly.

5.3.3 Test Result

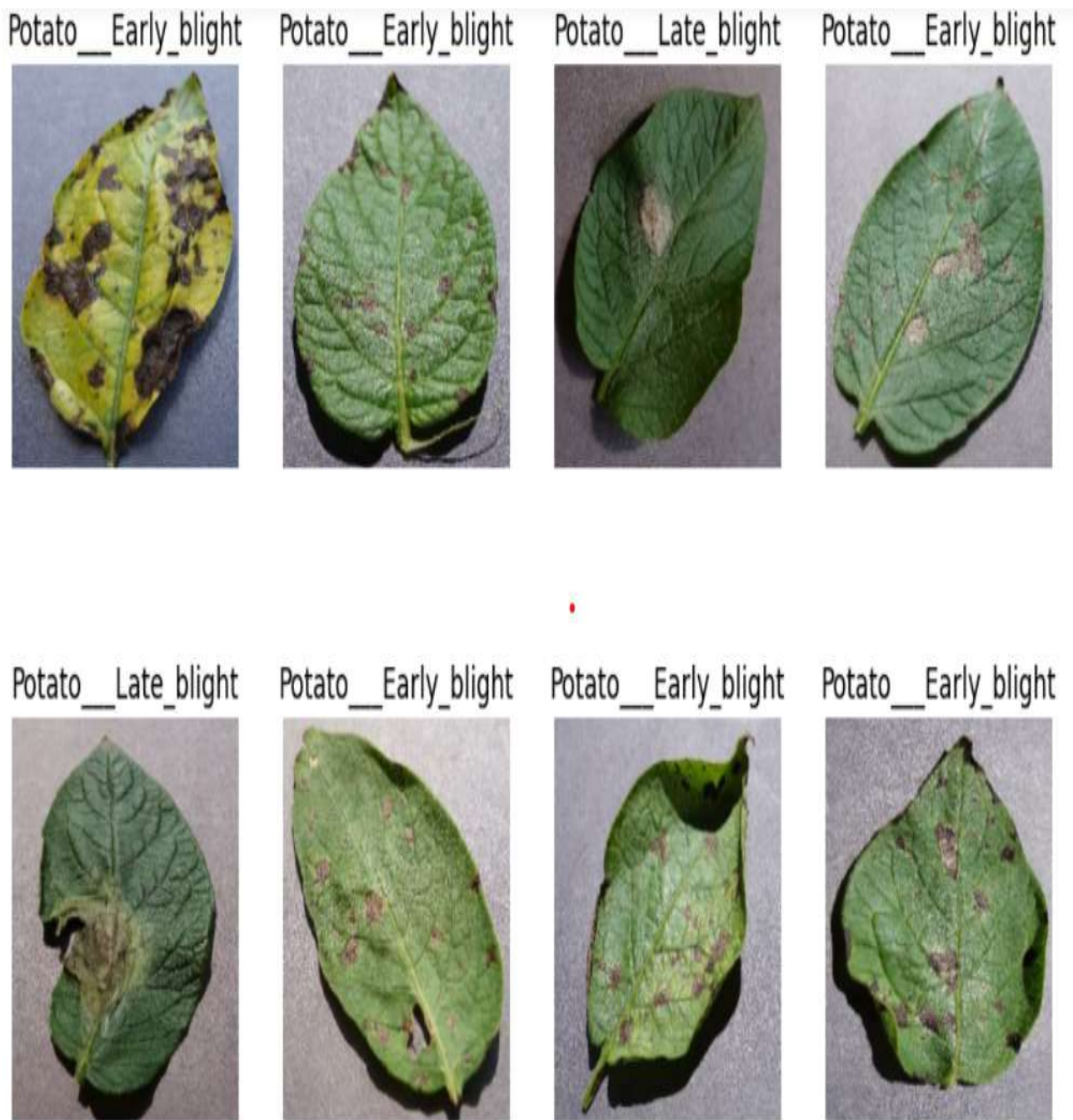


Figure 5.3: Test result of Crop Disease Identification

## Chapter 6

# RESULTS AND DISCUSSIONS

### 6.1 Efficiency of the Proposed System

The proposed system for classifying the different crop disease has a good accuracy and gives us a good result. The system has potential to reduce the burden of the farmers as well as researchers as it acts as an early detector for the crops. This application can also reduce the loss of crops as it can pre-emptively give warnings as well as help new farmers and researchers from making a mistake by double checking their doubts. Further, future iteration can add more diseases and better detection algorithm. In this project, a research to classify Plant leaf Disease Classification over static facial images using deep learning techniques was developed. This is a complex problem that has already been approached several times with different techniques. While good results have been achieved using feature engineering, this project focused on feature learning, which is one of DL promises. While feature engineering is not necessary, image pre-processing boosts classification accuracy. Hence, it reduces noise on the input data. Nowadays, Agriculture based AI Plant leaf disease includes is heavily required. The solution totally based on feature learning does not seem close yet because of a major limitation. Thus, leaf Disease classification could be achieved by means of deep learning techniques.

### 6.2 Comparison of Existing and Proposed System

#### **Existing system:**

The existing systems for crop disease detection vary widely in terms of technological sophistication and deployment scale. Traditional methods often rely on manual inspection by agronomists or extension workers who visually assess crop health in the field. While these methods are effective to some extent, they are labor-intensive, time-consuming, and susceptible to human error. Another common approach involves the use of handheld devices or cameras to capture images of diseased crops,



which are then analyzed using basic image processing techniques or rule-based algorithms. While these methods provide rapid feedback to farmers, they often lack the accuracy and scalability required for large-scale disease monitoring across diverse agricultural landscapes.

### **Proposed system:**

The proposed system for crop disease identification using machine learning offers a multitude of advantages poised to revolutionize agricultural practices. By leveraging deep learning models like Convolutional Neural Networks, LeNet-5 the system can swiftly and accurately identify various crop diseases from images, aiding farmers in early detection and precise intervention. These models excel in pattern recognition, enabling them to distinguish between healthy and diseased crops across diverse environmental conditions, leading to timely responses and improved crop management strategies.

## **6.3 Sample Code**

```
1 import tensorflow as tf
2 from tensorflow.keras import models, layers
3 import matplotlib.pyplot as plt
4 from IPython.display import HTML
5
6 BATCH_SIZE = 32
7 IMAGE_SIZE = 256
8 CHANNELS=3import tensorflow as tf
9 from tensorflow.keras import models, layers
10 import matplotlib.pyplot as plt
11 from IPython.display import HTML
12 EPOCHS=50
13
14 dataset = tf.keras.preprocessing.image_dataset_from_directory(
15     "PlantVillage",
16     seed=123,
17     shuffle=True,
18     image_size=(IMAGE_SIZE, IMAGE_SIZE),
19     batch_size=BATCH_SIZE
20 )
21
22 class_names = dataset.class_names
23 class_names
24
25 for image_batch, labels_batch in dataset.take(1):
26     print(image_batch.shape)
27     print(labels_batch.numpy())
```

```
28
29 plt.figure(figsize=(10, 10))
30 for image_batch, labels_batch in dataset.take(1):
31     for i in range(12):
32         ax = plt.subplot(3, 4, i + 1)
33         plt.imshow(image_batch[i].numpy().astype("uint8"))
34         plt.title(class_names[labels_batch[i]])
35         plt.axis("off")
36
37 len(dataset)
38
39 train_size = 0.8
40 len(dataset)*train_size
41
42 train_ds = dataset.take(54)
43 len(train_ds)
44
45 test_ds = dataset.skip(54)
46 len(test_ds)
47
48 val_size=0.1
49 len(dataset)*val_size
50
51 val_ds = test_ds.take(6)
52 len(val_ds)
53
54 test_ds = test_ds.skip(6)
55 len(test_ds)
```

## Output

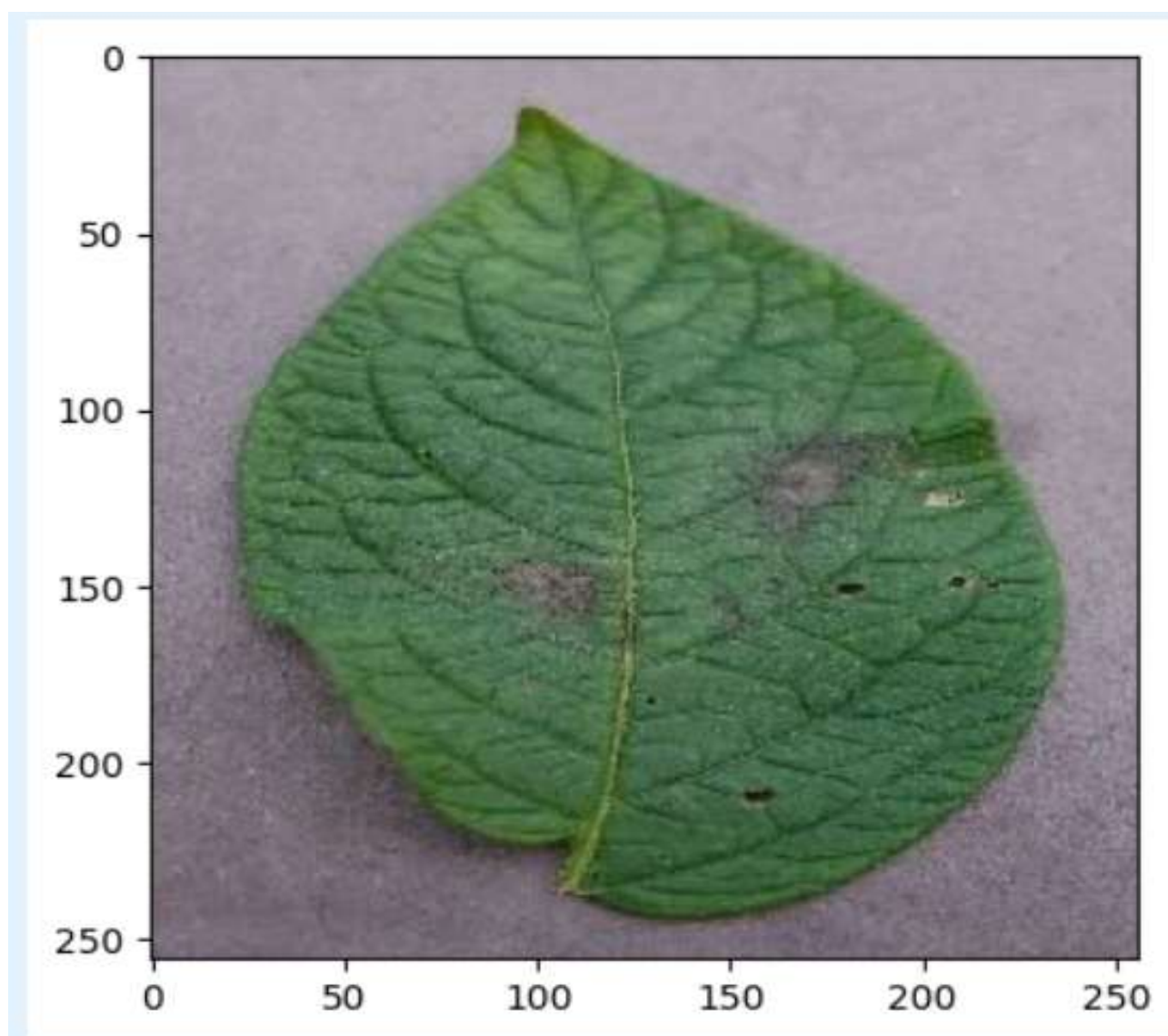


Figure 6.1: **Output of Crop Leaf**



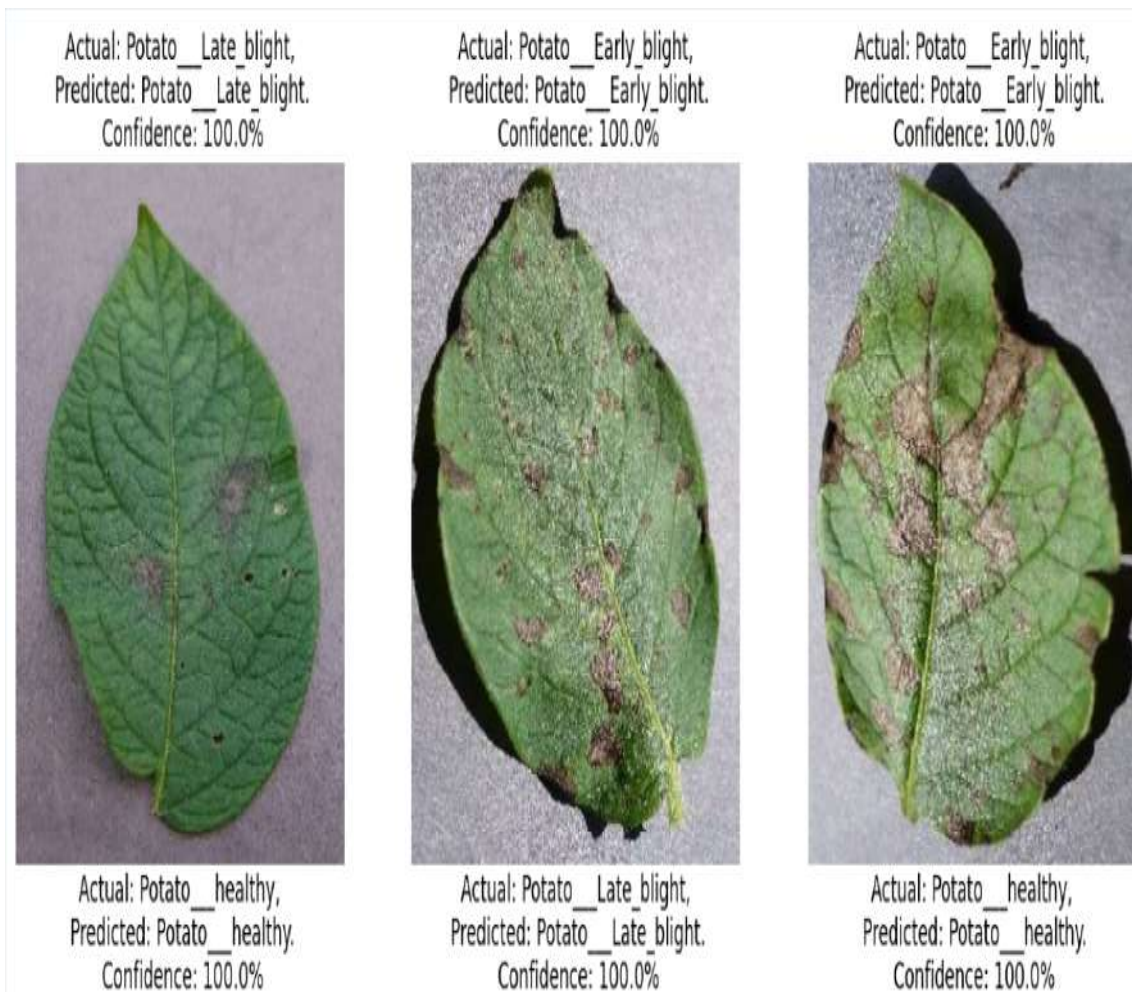


Figure 6.2: **Output of Different Leafs**

## **Chapter 7**

# **CONCLUSION AND FUTURE ENHANCEMENTS**

### **7.1 Conclusion**

In this project, a research to classify Plant leaf Disease Classification over static facial images using deep learning techniques was developed. This is a complex problem that has already been approached several times with different techniques. While good results have been achieved using feature engineering, this project focused on feature learning, which is one of DL promises. While feature engineering is not necessary, image pre-processing boosts classification accuracy. Hence, it reduces noise on the input data. Nowadays, Agriculture based AI Plant leaf disease includes is heavily required. The solution totally based on feature learning does not seem close yet because of a major limitation. Thus, leaf Disease classification could be achieved by means of deep learning techniques.

### **7.2 Future Enhancements**

This work can be further improved by adding more data into our dataset. Furthermore, with the advancement and proposal of more algorithms we can improve our model and update it if required. Also, for our web application we can make it more interactive by providing important knowledge about the plant disease along with measures to prevent them. Lastly, making our model accessible to more people can make it such that they can further develop on our work

## Chapter 8

# PLAGIARISM REPORT

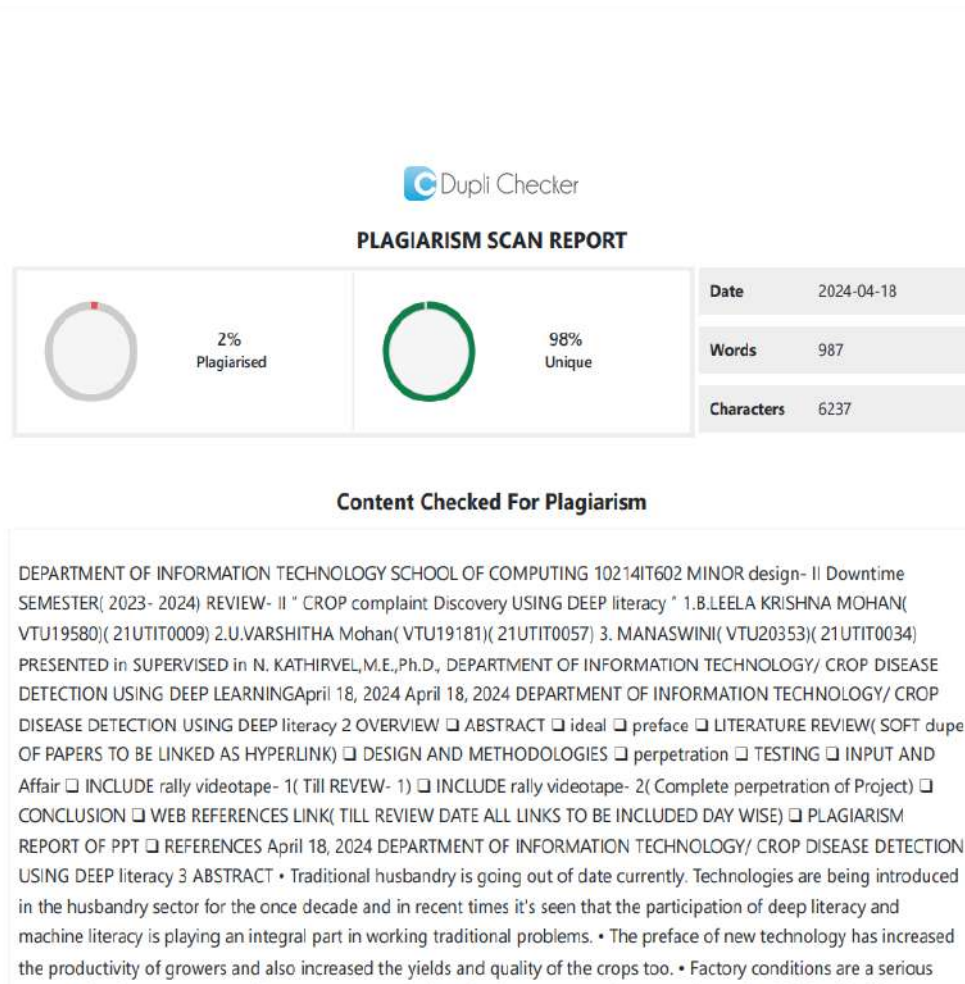


Figure 8.1: Plagiarism Report

# Chapter 9

## SOURCE CODE & POSTER PRESENTATION

### 9.1 Source Code

```
1 import tensorflow as tf
2 from tensorflow.keras import models, layers
3 import matplotlib.pyplot as plt
4 from IPython.display import HTML
5
6 BATCH_SIZE = 32
7 IMAGE_SIZE = 256
8 CHANNELS=3import tensorflow as tf
9 from tensorflow.keras import models, layers
10 import matplotlib.pyplot as plt
11 from IPython.display import HTML
12 EPOCHS=50
13
14 dataset = tf.keras.preprocessing.image_dataset_from_directory(
15     "PlantVillage",
16     seed=123,
17     shuffle=True,
18     image_size=(IMAGE_SIZE, IMAGE_SIZE),
19     batch_size=BATCH_SIZE
20 )
21
22 class_names = dataset.class_names
23 class_names
24
25 for image_batch, labels_batch in dataset.take(1):
26     print(image_batch.shape)
27     print(labels_batch.numpy())
28
29 plt.figure(figsize=(10, 10))
30 for image_batch, labels_batch in dataset.take(1):
31     for i in range(12):
32         ax = plt.subplot(3, 4, i + 1)
33         plt.imshow(image_batch[i].numpy().astype("uint8"))
34         plt.title(class_names[labels_batch[i]])
35         plt.axis("off")
```

```

36
37 len(dataset)
38
39 train_size = 0.8
40 len(dataset)*train_size
41
42 train_ds = dataset.take(54)
43 len(train_ds)
44
45 test_ds = dataset.skip(54)
46 len(test_ds)
47
48 val_size=0.1
49 len(dataset)*val_size
50
51 val_ds = test_ds.take(6)
52 len(val_ds)
53
54 test_ds = test_ds.skip(6)
55 len(test_ds)
56
57 def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0.1, shuffle=True,
58                               shuffle_size=10000):
59     assert (train_split + test_split + val_split) == 1
60
61     ds_size = len(ds)
62
63     if shuffle:
64         ds = ds.shuffle(shuffle_size, seed=12)
65
66     train_size = int(train_split * ds_size)
67     val_size = int(val_split * ds_size)
68
69     train_ds = ds.take(train_size)
70     val_ds = ds.skip(train_size).take(val_size)
71     test_ds = ds.skip(train_size).skip(val_size)
72
73     return train_ds, val_ds, test_ds
74
75 train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)
76 len(train_ds)
77
78 len(val_ds)
79
80 len(test_ds)
81
82 train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
83 val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
84 test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)

```

```

85
86 resize_and_rescale = tf.keras.Sequential([
87     layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
88     layers.experimental.preprocessing.Rescaling(1./255),
89 ])
90
91 data_augmentation = tf.keras.Sequential([
92     layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
93     layers.experimental.preprocessing.RandomRotation(0.2),
94 ])
95
96 train_ds = train_ds.map(
97     lambda x, y: (data_augmentation(x, training=True), y)
98 ).prefetch(buffer_size=tf.data.AUTOTUNE)
99
100 input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
101 n_classes = 3
102
103 model = models.Sequential([
104     resize_and_rescale,
105     layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
106     layers.MaxPooling2D((2, 2)),
107     layers.Conv2D(64, kernel_size = (3,3), activation='relu'),
108     layers.MaxPooling2D((2, 2)),
109     layers.Conv2D(64, kernel_size = (3,3), activation='relu'),
110     layers.MaxPooling2D((2, 2)),
111     layers.Conv2D(64, (3, 3), activation='relu'),
112     layers.MaxPooling2D((2, 2)),
113     layers.Conv2D(64, (3, 3), activation='relu'),
114     layers.MaxPooling2D((2, 2)),
115     layers.Conv2D(64, (3, 3), activation='relu'),
116     layers.MaxPooling2D((2, 2)),
117     layers.Flatten(),
118     layers.Dense(64, activation='relu'),
119     layers.Dense(n_classes, activation='softmax'),
120 ])
121
122 model.build(input_shape=input_shape)
123
124 model.summary()
125
126 model.compile(
127     optimizer='adam',
128     loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
129     metrics=['accuracy']
130 )
131
132 history = model.fit(
133     train_ds,
134     batch_size=BATCH_SIZE,

```

```

135     validation_data=val_ds ,
136     verbose=1,
137     epochs=50,
138 )
139
140 scores = model.evaluate(test_ds)
141
142 scores
143
144 history
145
146 history.params
147
148 history.history.keys()
149
150 type(history.history['loss'])
151
152 len(history.history['loss'])
153
154 history.history['loss'][:5]
155
156 acc = history.history['accuracy']
157 val_acc = history.history['val_accuracy']
158
159 loss = history.history['loss']
160 val_loss = history.history['val_loss']
161
162 plt.figure(figsize=(8, 8))
163 plt.subplot(1, 2, 1)
164 plt.plot(range(EPOCHS), acc, label='Training Accuracy')
165 plt.plot(range(EPOCHS), val_acc, label='Validation Accuracy')
166 plt.legend(loc='lower right')
167 plt.title('Training and Validation Accuracy')
168
169 plt.subplot(1, 2, 2)
170 plt.plot(range(EPOCHS), loss, label='Training Loss')
171 plt.plot(range(EPOCHS), val_loss, label='Validation Loss')
172 plt.legend(loc='upper right')
173 plt.title('Training and Validation Loss')
174 plt.show()
175
176 import numpy as np
177 for images_batch, labels_batch in test_ds.take(1):
178
179     first_image = images_batch[0].numpy().astype('uint8')
180     first_label = labels_batch[0].numpy()
181
182     print("first image to predict")
183     plt.imshow(first_image)
184     print("actual label:", class_names[first_label])

```




```

185
186     batch_prediction = model.predict(images_batch)
187     print("predicted label:", class_names[np.argmax(batch_prediction[0])])
188
189 def predict(model, img):
190     img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
191     img_array = tf.expand_dims(img_array, 0)
192
193     predictions = model.predict(img_array)
194
195     predicted_class = class_names[np.argmax(predictions[0])]
196     confidence = round(100 * (np.max(predictions[0])), 2)
197     return predicted_class, confidence
198
199 plt.figure(figsize=(15, 15))
200 for images, labels in test_ds.take(1):
201     for i in range(9):
202         ax = plt.subplot(3, 3, i + 1)
203         plt.imshow(images[i].numpy().astype("uint8"))
204
205         predicted_class, confidence = predict(model, images[i].numpy())
206         actual_class = class_names[labels[i]]
207
208         plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence: {
209             confidence}%")
210
211         plt.axis("off")
212
213
214 import os
215 model_version=max([int(i) for i in os.listdir("../crop-disease-detection") + [0]])+1
216 model.save(f"../models/{model_version}")
217
218 model.save("../potatoes.h5")

```



## 9.2 Poster Presentation



### "CROP DISEASE DETECTION USING DEEP LEARNING"

Department of Information Technology  
School of Computing  
10214IT602- MINOR PROJECT-II  
WINTER SEMESTER 2023-2024

#### ABSTRACT

Traditional farming is going out of date nowadays. Technologies are being introduced in the farming sector for the past decade and in recent years it is seen that the participation of deep learning and machine learning is playing an integral role in solving traditional problems.

Deep learning-enabled developments in the field of computer vision have paved the path for computer-assisted plant disease diagnosis.

Deep Learning has achieved great success in the categorization of a number of plant diseases by exploiting its ability to recognize objects with the help of convolutional neural networks.

Various deep learning algorithms like AlexNet and VGGNet are applied in a publicly available dataset (plantvillage dataset) so that the neural network can capture the various features of a specific disease and diagnose it accordingly using a human-like decision-making skill.


#### INTRODUCTION

India being an agriculture country about 70% of the population depends on it as their main source of income and food. Agriculture plays an important part of the Indian economy as it contributes about 17% of the total GDP. Farmers have wide range in selecting their crops and finding a suitable pesticide for it but in spite of all their efforts it can all be vain if they can't identify the disease plaguing their crops. Thus, disease on crops can significantly reduce the quality and quantity of agricultural products along with economical damage to the farmers. To successfully cultivate crops without incurring much loss we need to properly identify the disease and remedy it, this requires a lot of work and processing time as detecting each and every plant can be tedious and time-consuming. To lessen the burden of the farmers along with their losses we propose the use of a system which can detect infected plants so that we can curb the spread of infection and diseases at an earlier stage thus reducing losses and crop failure. In most cases symptoms like fungal infection and rot can be seen on the leaves, stem and fruit. This project provides an insight into how we deal with the problem and further discuss the challenges of our work and how we can improve upon it in future work.

#### RESULTS


The proposed system for classifying the different crop disease has a good accuracy and gives us a good result.

- The system has potential to reduce the burden of the farmers as well as researchers as it acts as an early detector for the crops.
- This application can also reduce the loss of crops as it can pre-emptively give warnings as well as help new farmers and researchers from making a mistake by double-checking their doubts.
- Further future iteration can add more diseases and better detection algorithms.
- In this project, a research to classify Plant Leaf Disease Classification over static facial images using deep learning techniques was developed.
- This is a complex problem that has already been approached several times with different techniques.
- While good results have been achieved using feature engineering, this project focused on feature learning, which is one of DL promises.
- While feature engineering is not necessary, image pre-processing boosts classification accuracy. Hence, it reduces noise in the input data.



#### STANDARDS AND POLICIES

Anacoda Prompt: Anacoda prompt is a type of command line interface which explicitly deals with all the ML (Machine Learning) modules. And navigator is available in all the Windows, Linux, and MacOS. The anacoda prompt has many number of IDEs which make the coding easier. The UI can also be implemented in python. Standard Used: ISO/IEC 27001. Jupyter: It's like an open source web application that allows us to share and create the documents which contain the live code, equations, visualizations and narrative text. It can be used for data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning. Standard Used: ISO/IEC 27001. Google Colab: Google Standard Used: ISO/IEC 27001.



#### METHODOLOGIES

Preprocessing and Training the model (CNN): The dataset is preprocessed such as Image reshaping, resizing and conversion to an array form. Similar processing is also done on the test image. A dataset consisting of about 10 different classes of leaf, out of which any image can be used as a test image for the software. Methodology of the system: The train dataset is used to train the model (CNN) so that it can identify the test image and the disease it has. CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D, and MaxPooling2D. After the model is trained successfully, the software can identify the Plant Leaf Disease prediction image contained in the dataset. After successful training and preprocessing, comparison of the test image and trained model takes place to predict the Sign language.

#### CONCLUSIONS

The proposed system for classifying the different crop disease has a good accuracy and gives us a good result. The system has potential to reduce the burden of the farmers as well as researchers as it acts as an early detector for the crops. This application can also reduce the loss of crops as it can pre-emptively give warnings as well as help new farmers and researchers from making a mistake by double-checking their doubts. Further, future iteration can add more diseases and better detection algorithm.

#### ACKNOWLEDGEMENT

- Dr. N. KATHIRVEL, M.E., Ph.D., ASSISTANT PROFESSOR(SG)
- 9994545289
- drkathirveln@veltech.edu.in

#### TEAM MEMBER DETAILS

B. Leela Krishna Mohan  
(VTU19580)  
M. Manaswini (VTU20353)  
U. Varshitha (VTU19181)  
999462792  
8019157421  
7702469369  
v192532@veltech.edu.in  
v1920353@veltech.edu.in  
v19181@veltech.edu.in

Figure 9.1: Poster Presentation for Crop Disease Detection

# References

- [1] Agarwal, G., Pandey, P., Dubey, S., Somvanshi, V. S, "Plant diseases and pests detection based on deep learning a review. *Plant Methods*", 17(1), 1-21. 2021.
- [2] Amina Khatra, "Yellow Rust Extraction in Wheat Crop based on Color Segmentation Techniques", *IEEE Crop Disease*, vol.1, no.2,, 2019.
- [3] Ashwini T Sapka, Uday V Kulkarni, "Comparative study of Leaf Disease Diagnosis system using Texture features and Deep Learning Features", 2018.
- [4] Behera S.K., Shreelekha Pandey, Shivani, "Disease Classification and Grading of Orange Using Machine Learning and Fuzzy Logic"; *Proceedings of the 2018 IEEE International Conference on Communication and Signal Processing (ICCSP)*; Chennai, India. 3–5 April 2018.
- [5] Dubey S, Vrushali Kharad, Nileshsingh V. Thakur, "Detection and Classification of Apple Fruit Diseases Using Complete Local Binary Patterns"; *Proceedings of the 2012 3rd International Conference on Computer and Communication Technology*; Allahabad, India. 23–25 November 2012.
- [6] Ishaan Dawar, Sanchit Wadhwan, "Predicting Cardiovascular Disease using Deep Learning Techniques", *2023 IEEE Silchar Subsection Conference (SILCON)*, pp.1-6, 2023.
- [7] Malathy S, De Lemmus, Brendan P. McAntosh, ABM Rezbaul Islam, "Disease Detection in Fruits Using Image Processing"; *Proceedings of the 2021 6th International Conference on Inventive Computation Technologies (ICICT)*; Coimbatore, India. 20–22 January 2021.
- [8] Paul, J., Sikdar, P. K, "Plant Disease Detection Using Image Segmentation and Soft Computing Techniques. *International Journal of Computer Applications*", 139(11), 19-26. 2022.
- [9] Pallavi Pandey, Diksha Tandekar, Snehlata Dongre, "Plant Disease Detection Using Deep Learning Model - Application FarmEasy", *2023 International Conference on Advanced Computing Technologies and Applications (ICACTA)*, pp.1-6, 2023.

- [10] Ruchi Rani,"Role of Artificial Intelligence in Agriculture: An Analysis and Advancements With Focus on Plant Diseases", IEEE Access, vol.11, pp.137999-138019, 2023.
- [11] Rumpf T,.R. Sowjanya, T. Lakshmi Prasanna, P. Ashwak Khan,"Early detection and classification of plant diseases with Support Vector Machines based on hyperspectral reflectanc"e. Comput. Electron. Agric. 2010.
- [12] Ramesh S,Rakiba Rayhana, Zhenyu Ma, Zheng Liu, "Plant Disease Detection Using Machine Learning"; Proceedings of the 2018 International Conference on Design Innovations for 3Cs Compute Communicate Control (ICDI3C); Bangalore, India. 25–28 April 2018.
- [13] Simranjeet Kaur, Geetanjali Babbar, Navneet Sandhu, Dr. Gagan Jindal,"Various Plant Disease Detection Using Image Processing Methods",2018.
- [14] Sumit Nema, Bharat Mishra and Mamta Lambert,"Android Application of Wheat Leaf Disease Detection and Prevention using Machine Learning",2020.
- [15] Varun Jindal,"Improving Agricultural Efficiency: Severity-based Diagnosis of Bottle Gourd Leaf Diseases Using Federated Learning CNN", 2023.