# A PROJECT REPORT

# ON

# “Plant Disease Detection through Image Classification”


A project report submitted in partial fulfillment for requirements of 4th semester of M.Sc. – Computer Science course during the session 2021-23

Submitted by:
**Oishani Ghosh**
**(Roll No.: 2110015015013)**

Under the guidance of

**Dr. S.P. Kannojia**

**Department of Computer Science**
University of Lucknow
**Lucknow**


# CERTIFICATE

*This is to certify that* ***Oishani Ghosh*** *(2110015015013) has carried out the major project work presented in this report entitled "Plant Disease Detection through Image Classification" in partial fulfillment for the award of* ***Master of Science*** *in* ***Computer Science*** *at* ***University of Lucknow****, under the supervision of Dr. S.P. Kannojia for the session of 2021-23.*

**Dr. S.P. Kannojia**
Project Guide
Department of Computer Science
University of Lucknow
Lucknow

**Prof. Vibhuti Rai**
H.O.D.
Department of Computer Science
University of Lucknow
Lucknow

# Acknowledgement

I would like to take this opportunity to express my sincere gratitude to everyone who has contributed to the successful completion of my Computer Science project on Plant Disease Detection System.

To begin with, I would first like to express my deepest appreciation to my project guide, **Dr. S.P. Kannojia**, for providing me with the necessary resources, mentorship and support throughout the project. His expertise in the field has been invaluable in helping me navigate through the technical challenges I encountered. I would also like to extend my heartfelt thanks to **Dr. Vibhuti Rai**, the Head of the CS department for giving me the opportunity to present my work.

I would also like to extend my thanks to my classmates and friends who have helped me with data collection, proofreading, and providing constructive feedback. Their insights and assistance have been invaluable in improving the quality of this project. I am also grateful to the staff of the college library and computer lab for providing me with access to the necessary resources and equipment.

At last but not the least, I would like to thank my parents for their unwavering support and encouragement throughout this project work and associated research.

Thank you, all!

# Declaration

*I hereby certify that the work which is being presented in the project report entitled "Plant Disease Detection System" in partial fulfillment of the requirement for the award of the Degree of Master's in Computer Science and submitted in the Department of Computer Science of University of Lucknow is an authentic record of our own work carried out during a period from March 1, 2023 to June 30, 2023 under the supervision of Dr. S.P. Kannojia, Department of Computer Science, University of Lucknow. The matter presented in this report has not been submitted by us for the award of any other degree of this or any other Institute/University.*

**Oishani Ghosh**

2110015015013

*This is to certify that the above statement made by the candidate is correct to the best of my knowledge.*

Date:

**Dr. S. P. Kannojia**

Assistant Professor

# Preface

In an age defined by technological progress and a growing global population, ensuring food security through sustainable agriculture has emerged as a critical imperative. Amid various challenges, plant diseases pose a significant threat to crop yields and food supply chains. Detecting these diseases accurately and promptly is vital for averting widespread damage and ensuring agricultural resilience.

The convergence of cutting-edge technologies and agriculture offers new solutions to age-old problems. This project explores the potential of deep learning and transfer learning to transform plant leaf disease detection. Deep learning excels in image analysis, while transfer learning harnesses pre-existing model knowledge for efficient training. This project tackles a notable research gap by investigating how transfer learning can enhance the accuracy and efficiency of disease detection systems. Through literature exploration, method development, rigorous experiments, and insightful analysis, the project sheds light on transfer learning's impact in agriculture.

The upcoming pages unveil a roadmap spanning literature, methodology, experimentation, and interpretation. This research not only contributes to plant disease detection discussions but also offers practical insights for future agriculture. Amidst deep learning and transfer learning intricacies, this work strives to create resilient, efficient, and sustainable farming practices.

This journey exemplifies interdisciplinary collaboration, technological innovation, and commitment to global food security. Amidst climate shifts and population growth, this research could guide practices that blend traditional agriculture with transformative technology.

This preface invites readers to explore data-driven insights, algorithmic discoveries, and the fusion of data science and agriculture. The ensuing pages narrate an expedition through data, algorithms, and insights, sowing innovation in agriculture's fertile ground.

# Table of Contents


1. INTRODUCTION ..... 9

1.1. Background ..... 9

1.2. Plant Disease Detection ..... 9

1.3. Problem Statement ..... 11

1.4. Aims and Objectives ..... 12

2. LITERATURE REVIEW ..... 13

2.1. Traditional Methods of Plant Disease Detection ..... 13

*2.1.1. Visual Observation* ..... 13

2.1.2. Indicator Plants and Biological Diagnostics ..... 14

*2.1.3. DNA-Based and Serological Methods* ..... 14

2.1.4. Microscopy ..... 16

2.1.5. Mycological Analysis ..... 17

2.1.6. Significance and Limitations ..... 18

2.1.7. Evolution and Integration ..... 19

2.2. Plant Disease Detection using Machine Learning ..... 19

2.2.1. Plant Disease Detection using Deep Learning ..... 22

2.3. Advantages of Plant Disease Detection Systems ..... 23

2.4. Related Work ..... 26

Table-1: Summary of discussed works ..... 32

2.5. Research Gap and Transfer Learning ..... 33

3. METHODOLOGY ..... 35

3.1. SDLC Model – Waterfall Model ..... 37

4. DESIGN & IMPLEMENTATION ..... 41

4.1. Data Flow Diagram ..... 41

4.2. ML Model Implementation ..... 43
4.3. Flask Code for UI ..... 57
4.4. HTML Code for UI ..... 61
5. EVALUATION & RESULTS ..... 67
5.1. ML Model Evaluation & Results ..... 67
5.2. End-to-end Evaluation & Results ..... 73
6. DISCUSSION ..... 76
7. RECOMMENDATIONS ..... 78
8. CHALLENGES AND FUTURE SCOPE ..... 81
9. CONCLUSION ..... 82
REFERENCES ..... 84

# 1. INTRODUCTION

## 1.1. Background

In the ever-evolving realm of agriculture and plant sciences, the advent of technology has brought about revolutionary advancements that are transforming the way we approach plant health and disease detection. Among these innovations, the development and application of plant disease detection systems utilizing image classification techniques stand as a beacon of hope and progress. These systems harness the power of deep learning and computer vision to scrutinize leaf images, dissecting intricate details imperceptible to the human eye, in order to discern the health status of plants with remarkable accuracy.

Plants, being the lifeblood of our ecosystems and crucial to global food security, are vulnerable to a myriad of diseases caused by bacteria, viruses, fungi, and environmental stressors. Timely identification of these diseases is pivotal for farmers and researchers, enabling swift intervention and preventing catastrophic yield losses. Traditional methods of disease detection often relied on visual inspection by agronomists or plant pathologists, a process prone to subjectivity and limited scalability. In this context, the marriage of cutting-edge technology with plant science has given birth to automated and objective disease detection systems, with leaf image classification at their core.

## 1.2. Plant Disease Detection

At the heart of plant disease detection systems lies the principle of transfer learning, a technique that has revolutionized the field of image classification.

Transfer learning leverages pre-trained convolutional neural networks (CNNs) that have already been trained on vast image datasets, enabling them to extract intricate features from images. These features, which range from simple edges to complex textures, have the potential to be highly informative in discerning the visual cues associated with various plant diseases. By fine-tuning these pre-trained models on specific datasets of plant images, researchers are able to develop highly accurate disease detection models, capable of identifying subtle deviations from healthy leaf patterns.

The process commences with the collection of extensive datasets comprising images of healthy leaves as well as those afflicted with various diseases. These datasets are meticulously curated to encompass a wide range of environmental conditions, plant types, and disease severities. The pre-trained CNN models, often derived from architectures like ResNet, VGG, or Inception, serve as feature extractors. The initial layers of these networks, which are responsible for detecting basic features like edges and shapes, are retained while the final classification layers are replaced with custom layers suited for the disease detection task. This amalgamation of generic feature extraction with specialized classification forms the crux of transfer learning, allowing the model to acquire both the general and specific knowledge necessary for accurate disease identification.

One of the remarkable advantages of utilizing transfer learning in plant disease detection is its ability to surmount the limitations of limited datasets. Training deep neural networks from scratch requires vast amounts of data, an often unattainable luxury in the realm of agriculture due to factors like the cost of data collection, diversity of plant types, and temporal variations. Transfer learning circumvents this bottleneck by capitalizing on the knowledge encapsulated within the pre-trained models, enabling accurate disease detection even with smaller

datasets. This democratization of sophisticated technology ensures that farmers across the globe, regardless of their resources, can benefit from state-of-the-art disease detection systems.

A critical facet of the success of these systems is the quality and diversity of the datasets used for training. The datasets must encapsulate the natural variability inherent in plant leaves, spanning different growth stages, lighting conditions, disease severities, and even interferences from pests or nutrient deficiencies. Augmentation techniques, such as rotation, flipping, and changing lighting conditions, are often employed to artificially expand the dataset and improve the model's ability to generalize across variations.

As these trained models are deployed in real-world scenarios, they become an indispensable tool in the arsenal of plant pathologists and farmers. A farmer could capture an image of a potentially diseased leaf using a smart phone, upload it to a cloud-based platform, and receive an instant diagnosis. The model's predictions could encompass not only the presence of disease but also its type and severity, equipping farmers with actionable insights to mitigate the issue effectively. This real-time diagnosis, untethered from the limitations of human availability and expertise, can play a pivotal role in curbing disease spread and optimizing crop management strategies.

## 1.3. Problem Statement

The identification of plant diseases traditionally relies on visual inspection by farmers and experts, which can be time-consuming, subjective, and prone to human error. The expertise required for accurate diagnosis may not always be available in remote or resource-constrained areas.

## 1.4. Aims and Objectives

The aim of this project is to develop an accurate and efficient system for the detection of plant diseases through image classification. The goal is to showcase the effectiveness of deep learning techniques in identifying and classifying diseases in plant leaves, which can have significant implications for crop management and agriculture.

With the following objectives, this work seeks to create an efficient and accurate plant disease detection model using transfer learning with the EfficientNetB3 architecture.

- Implementing Deep Learning model with Transfer Learning technique to classify plant leaf images
- Creating a UI for detection of plant diseases
- Real-time detection of disease in plant leaves

# 2. LITERATURE REVIEW

Plant diseases pose significant threats to global agriculture, causing substantial economic losses and impacting food security. Timely and accurate disease detection is crucial to mitigate these effects and ensure sustainable crop production. Over the years, advancements in technology have led to the development of various methods and techniques for plant leaf disease detection, ranging from traditional visual scouting to state-of-the-art deep learning approaches. This literature review explores and analyzes key research works in the domain of plant leaf disease detection, highlighting their methodologies, contributions, and potential implications.

## 2.1. Traditional Methods of Plant Disease Detection

The field of plant pathology has long been engaged in developing methods to identify and diagnose plant diseases. Traditional methods for the detection of plant diseases have laid the foundation for understanding pathogen types and their symptoms. These methods, though often manual and time-consuming, have provided crucial insights into disease control and prevention.

### *2.1.1. Visual Observation*

(Khakimov et al., 2022) and (Buja et al., 2021) emphasize the significance of visual observation as one of the oldest and most fundamental methods for disease detection. In agricultural practices spanning centuries, plant pathologists and farmers have consistently turned to the power of human observation to glean insights into the health and vitality of their crops. This approach capitalizes on the

remarkable ability of experienced observers to detect subtle changes in plant appearance, indicative of underlying diseases or stressors.

Plant symptoms, ranging from discolorations and wilting to deformations and the formation of lesions, are readily apparent to the trained eye. Such visible signs often serve as red flags, alerting practitioners to potential issues that demand further investigation. The discoloration of leaves, for instance, might indicate the presence of a fungal infection or nutrient deficiency. Similarly, the wilting of stems might hint at water stress or the colonization of vascular tissues by pathogens.

Visual observation operates as an initial filter, allowing practitioners to identify plants or areas that deviate from the norm. This early warning system is particularly valuable in large-scale agricultural settings, where rapid interventions can make the difference between minor losses and catastrophic outbreaks. The efficacy of visual observation hinges on the acuity of the observer's senses and their familiarity with the specific crops and diseases in question.

### 2.1.2. Indicator Plants and Biological Diagnostics

Indicator plants, highlighted by (Khakimov et al., 2022), have been utilized as a diagnostic tool to identify the presence of pathogens. Certain plant species are highly susceptible to specific pathogens and exhibit visible symptoms when infected. By exposing these indicator plants to suspected pathogens, researchers can observe the development of symptoms, confirming the presence of the pathogen. Additionally, biological diagnostics involve infecting healthy plants with extracts from diseased plants to induce similar symptoms, aiding in pathogen identification.

### *2.1.3. DNA-Based and Serological Methods*

(Martinelli et al., 2015) emphasize the pivotal role of accurate plant disease diagnosis in mitigating agricultural losses and enhancing food security. They underscore the significant contributions made by DNA-based and serological methods in reshaping disease detection strategies. These techniques, grounded in the principles of molecular biology and immunology, have ushered in a new era of precision and specificity in pathogen identification. Through DNA-based methods such as polymerase chain reaction (PCR) and serological assays like enzyme-linked immunosorbent assay (ELISA), researchers can pinpoint the presence of specific genetic material or antigens associated with pathogens.

Despite their transformative impact, DNA-based and serological methods are not immune to challenges. As elucidated by (Martinelli et al., 2015), one of the principal limitations lies in their ability to detect pathogens only after they have reached a detectable level, often corresponding to the symptomatic stage of infection. This drawback severely hampers their utility in identifying asymptomatic infections or detecting pathogens during the crucial window of early establishment within plant tissues. Furthermore, the sample collection, processing, and subsequent analysis associated with these methods entail a time-consuming process, often spanning a day or more. In the fast-paced realm of agriculture, where rapid intervention is key to averting widespread damage, the delay caused by these methods could be detrimental.

Recognizing these limitations, (Martinelli et al., 2015) call for the development and adoption of innovative methods that can offer rapid insights into disease dynamics, especially during the critical initial stages of infection. The review emphasizes the need for tools that can provide instantaneous results, ideally in the field itself, enabling immediate action. This imperative is particularly relevant in scenarios where early detection can influence disease management decisions, guiding the deployment of targeted interventions or preventive measures. As

agricultural landscapes evolve and pathogens adapt, embracing cutting-edge technologies that combine the precision of DNA-based and serological methods with the speed of real-time diagnostics will be crucial in shaping the future of plant disease detection and management strategies.

### 2.1.4. Microscopy

Microscopy, as underscored by (Khakimov et al., 2022), stands as a timeless yet indispensable tool in the arsenal of plant disease detection. Delving into the microscopic realm, this method unveils the intricate world of pathogens and their interactions with plant tissues. Through the lens of a microscope, plant pathologists can unravel the hidden details that often evade the naked eye. The method involves the careful preparation of plant tissues or samples, which are then subjected to close scrutiny under different types of microscopes.

The power of microscopy lies in its ability to reveal the microscopic structures that hold the key to pathogen identification. As (Khakimov et al., 2022) highlight, fungal spores, bacterial colonies, and the delicate threads of mycelia become discernible under the microscope's gaze. These structures serve as microscopic fingerprints, aiding in the precise identification of the pathogens responsible for plant afflictions. By observing the size, shape, color, and arrangement of these structures, plant pathologists can infer the type of pathogen and its potential impact on the plant.

Microscopy, however, extends beyond mere identification—it serves as a dynamic tool for monitoring disease progression and assessing the extent of infection. Through the examination of stained tissues or the observation of live pathogens, practitioners can track the spread of infections and the strategies employed by pathogens to colonize plant tissues. The method thus provides a

window into the arms race between plants and pathogens, shedding light on the battle at the cellular level.

While microscopy offers unparalleled insights, it also presents challenges. The process demands precision in sample preparation to ensure that structures are not distorted or damaged. Moreover, microscopy's efficacy hinges on the proficiency of the observer, as the interpretation of microscopic images requires expertise in distinguishing subtle variations. Despite these challenges, microscopy remains a cornerstone in laboratories dedicated to understanding plant diseases, serving as a bridge between the macroscopic symptoms observed in the field and the intricate molecular mechanisms at play within plant cells.

### 2.1.5. Mycological Analysis

Mycological analysis, as discussed by (Khakimov et al., 2022), constitutes a foundational pillar in the arsenal of traditional methods for plant disease detection. This method constitutes a carefully orchestrated process of cultivation and scrutiny, executed within the controlled confines of a laboratory setting. With roots in classical mycology, this technique centers on the growth and observation of fungal pathogens in a meticulously controlled environment, serving as a powerful means of pathogen identification and characterization.

The process commences with the collection of plant samples suspected of harboring fungal pathogens. These samples, gathered from various parts of the infected plant, range from leaves and stems to fruits and roots, as different fungi exhibit a predilection for specific plant tissues. Once secured, these samples become the subjects of a controlled yet dynamic experiment aimed at unraveling the mysteries of plant-fungus interactions.

Central to mycological analysis is the cultivation of collected samples on specialized growth media. These growth media, composed of a carefully balanced concoction of nutrients, mimic the ecological conditions conducive to the growth of the pathogenic fungi. Through the art of inoculation, tiny fragments of the collected plant tissues are transferred onto the growth media, where they serve as substrates for fungal growth. In the nurturing embrace of these artificial environments, the latent pathogens spring to life, offering researchers a glimpse into their hidden world.

As the fungal colonies burgeon, they take on distinctive forms and structures that provide valuable clues for identification. Mycologists and plant pathologists pore over these colonies, meticulously observing their macroscopic and microscopic features. The size, color, texture, and pattern of the colonies, along with the microscopic examination of spores, hyphae, and other structures, contribute to the process of accurate identification. Through this detailed scrutiny, practitioners draw upon their extensive knowledge of fungal taxonomy to assign a name to the pathogenic intruder.

However, it is important to note that mycological analysis is not without its challenges. The process demands technical expertise, a controlled laboratory environment, and time for the pathogens to manifest and grow. Moreover, certain fungi may exhibit slow growth rates or require specific conditions that complicate their cultivation. The method also primarily focuses on fungal pathogens, leaving out other types of plant pathogens such as bacteria or viruses.

### 2.1.6. Significance and Limitations

The traditional methods of plant disease detection discussed by (Khakimov et al., 2022) and (Buja et al., 2021) have played a pivotal role in disease management and prevention. These methods provide the foundational knowledge needed to

understand disease progression, causative agents, and symptom development. However, they are not without limitations. Visual observation is subjective and reliant on the observer's expertise. Microscopy and mycological analysis can be time-consuming and require specialized equipment. Additionally, these methods may not be suitable for detecting diseases at asymptomatic stages.

### 2.1.7. Evolution and Integration

As plant pathology advances, researchers are working to overcome the limitations of traditional methods by integrating them with modern technologies. (Buja et al., 2021) highlight the transition from lab-based analyses to field-use diagnostics. Portable systems and Internet of Things (IoT) technologies are being employed to enhance the speed and accuracy of disease diagnosis. Moreover, innovative methods, such as immunodiagnostics and molecular-genetic identification, are being integrated to improve reliability and efficiency.

## 2.2. Plant Disease Detection using Machine Learning

Plant diseases pose a significant threat to agricultural productivity and food security worldwide. The timely and accurate identification of these diseases is crucial for effective disease management and mitigation of crop losses. In recent years, machine learning techniques have emerged as promising tools for automated and accurate plant disease detection. This literature review delves into various studies that highlight the application of machine learning in plant disease detection, showcasing its potential to revolutionize agricultural practices.

Ramesh et al. (2018) underscore the role of machine learning algorithms in enhancing the recognition rate and accuracy of plant disease detection systems. They emphasize the significance of machine learning in diagnosing diseases by

analyzing patterns and features extracted from plant images. The study explores the effectiveness of traditional machine learning methods in identifying plant diseases and highlights the need for continued research to improve the performance of these techniques.

Deep learning, a subset of machine learning, has gained substantial attention in plant disease detection. (Ferentinos, 2018) presents an extensive exploration of deep learning models for plant disease identification and diagnosis. The review highlights the scalability and adaptability of deep learning methods in handling diverse plant species and diseases. The study also advocates for expanding the utilization of deep learning to encompass a wider range of plant pathogens, thereby bolstering its effectiveness in real-world scenarios.

(Shruthi et al., 2019) provide a comprehensive overview of machine learning classification techniques for plant disease detection. The review outlines the stages of general plant disease detection systems and conducts a comparative analysis of various machine learning approaches. By scrutinizing the strengths and limitations of different techniques, the study contributes to the understanding of how machine learning algorithms can be harnessed to improve disease identification accuracy.

A work by (Saleem et al., 2019) delves into the realm of deep learning for image-based plant disease detection. The study emphasizes the pivotal role of large and verified datasets in training accurate image classifiers. Leveraging deep learning architectures, the research showcases how convolutional neural networks (CNNs) can discern subtle differences between healthy and diseased plants, enabling automated and rapid disease identification.

Mohanty et al. (2016) illuminate the potential of deep learning in the context of plant disease detection using images. The study underscores the importance of

image datasets and introduces a dataset named "PlantVillage" to support research in this domain. Through the deployment of CNNs, the authors demonstrate the efficacy of deep learning models in achieving high accuracy in plant disease classification, reaffirming their utility in practical agricultural applications.

(Trivedi et al., 2020) contribute to the discourse by focusing on tomato leaf disease detection using machine learning. The study accentuates the significance of color-based features in differentiating between healthy and diseased tomato leaves. By employing machine learning techniques such as Support Vector Machine (SVM) and Random Forest, the research attests to the potential of machine learning algorithms to efficiently address specific crop diseases.

The literature also recognizes the evolution of machine learning approaches in plant disease detection. (Kothari, 2018) explores the integration of artificial intelligence and machine learning models to enhance the efficiency of disease detection strategies. By leveraging AI techniques, the study envisions a comprehensive cycle of plant disease detection and management that can significantly impact agricultural practices.

In conclusion, the studies presented in this literature review collectively highlight the transformative impact of machine learning, particularly deep learning, in the field of plant disease detection. These approaches demonstrate the potential to expedite disease identification, enable precision agriculture, and inform targeted interventions, ultimately contributing to the sustainable enhancement of global food production. As the agricultural landscape evolves, the integration of machine learning techniques holds promise in shaping a more resilient and productive agricultural sector.

### 2.2.1. Plant Disease Detection using Deep Learning

The application of deep learning techniques in plant disease detection has garnered substantial attention due to its potential for automating and improving the accuracy of disease identification. This literature review examines several studies that showcase the effectiveness of deep learning models in revolutionizing the field of plant disease detection.

Chowdhury et al. (2021) highlight the significance of computer vision and artificial intelligence in mitigating the adverse effects of plant diseases. The study emphasizes the role of deep learning architectures, particularly convolutional neural networks (CNNs), in achieving automatic and reliable leaf disease detection. By leveraging the power of CNNs, the research addresses the critical need for accurate disease diagnosis, which is essential for preventing significant crop losses.

Tiwari et al. (2020) delve into the domain of potato leaf disease detection using deep learning. The study emphasizes the superiority of deep learning models in mapping complex relationships within image data. The authors present a deep learning model tailored for potato leaf disease detection, underscoring the potential of these models to effectively identify specific diseases affecting different crops.

Barbedo (2019) contributes to the discourse by focusing on plant disease identification at the level of individual lesions and spots. The study validates the capabilities of deep neural networks in recognizing plant diseases, emphasizing the need for diverse and realistic image datasets. By demonstrating the effectiveness of deep learning approaches in classifying specific disease symptoms, the research underscores the potential of these methods for fine-grained disease detection.

Militante et al. (2019) emphasize the role of early plant disease detection in optimizing crop yields. The study investigates the application of deep learning techniques for plant disease and pest infestation recognition. By utilizing deep learning models, the researchers highlight the potential to accurately identify and classify diseases, contributing to more efficient disease management practices.

Li et al. (2021) present a comprehensive review of plant disease detection and classification using deep learning techniques. The study provides insights into current trends and challenges within the field, showcasing the progress made in leveraging deep learning for plant disease diagnosis. By synthesizing existing research, the authors shed light on the potential of deep learning to enhance disease detection accuracy.

The integration of deep learning with other technologies is explored by Bari et al. (2021), who propose a real-time approach for diagnosing rice leaf diseases using a deep learning-based faster R-CNN framework. The study leverages deep learning to overcome the limitations of conventional methods, offering a novel and efficient approach to disease diagnosis.

In conclusion, the studies discussed in this literature review collectively underline the transformative impact of deep learning techniques in plant disease detection. The utilization of CNNs and other deep learning architectures holds immense promise in automating disease identification, enhancing accuracy, and enabling timely interventions that can significantly mitigate the economic and environmental impact of plant diseases on global agriculture. As deep learning continues to evolve, its application in plant disease detection is poised to revolutionize the field, offering innovative solutions to age-old challenges in agriculture.

## 2.3. Advantages of Plant Disease Detection Systems

Plant leaf disease detection systems offer several advantages that significantly contribute to the management and sustainability of agricultural practices. These systems leverage advanced technologies, such as machine learning, deep learning, and image processing, to detect and classify diseases affecting plant leaves. The advantages of using such systems are as follows:

**Early Detection**: One of the primary advantages of plant leaf disease detection systems is their ability to identify diseases at an early stage. Detecting diseases in their early phases enables timely intervention and treatment, minimizing the risk of widespread infection and crop loss. Early detection is crucial for implementing effective control measures and preventing economic losses.

**Accuracy and Precision**: Modern plant disease detection systems utilize sophisticated algorithms that can analyze leaf images with high accuracy and precision (Geetha et al., 2020). These systems can differentiate between healthy and infected leaves based on subtle visual cues and patterns that might not be noticeable to the naked eye. This level of accuracy ensures reliable disease identification.

**Objective Diagnosis**: Traditional methods of disease detection often involve subjective human judgment, which can lead to variations in diagnosis. Plant leaf disease detection systems, on the other hand, offer objective and standardized results (Buja et al., 2021). This reduces the potential for human error and increases the consistency of diagnoses across different settings.

**Speed and Efficiency**: Automated detection systems operate at a significantly faster pace compared to manual methods (Fang et al., 2018). They can analyze a large number of leaf samples in a short period, making them highly efficient for disease surveillance in large-scale agricultural settings. Quick results allow for rapid decision-making and implementation of necessary actions.

**Remote Monitoring**: Some plant disease detection systems can be integrated with remote sensing technologies. This enables farmers and agricultural experts to monitor the health of crops from a distance. Remote monitoring is particularly valuable for monitoring vast fields and areas that are difficult to access (Zhang et al., 2019).

**Reduced Labor and Costs**: Automated disease detection systems reduce the need for manual labor involved in inspecting and diagnosing plant leaf diseases (Ramaswamy et al., 2023). This not only saves time but also cuts down labor costs associated with disease surveillance. Moreover, early detection and targeted treatment can lead to cost savings by minimizing the use of pesticides.

**Non-Destructive Analysis**: In many cases, traditional disease diagnosis methods require destructive sampling of plant tissues for laboratory analysis. Plant leaf disease detection systems can analyze images of leaves without harming the plants (Sankaran et al., 2010). This non-destructive approach is especially beneficial when studying rare or valuable plants.

**Data-Driven Insights**: The data collected by these systems can be valuable for generating insights into leaf disease patterns, trends, and the spread of pathogens. Aggregated data can help researchers and policymakers make informed decisions about disease control strategies and resource allocation (Roberts et al., 2021).

**Innovative Technologies**: The adoption of state-of-the-art technologies, such as deep learning and artificial intelligence, for leaf disease detection reflects innovation in agriculture. These systems demonstrate the potential of technology to revolutionize traditional farming practices, making them more efficient and sustainable (Li et al., 2020).

**Global Impact**: Plant diseases have a global impact on food security, affecting crop yields and economic stability. By providing accurate disease detection and management tools, these systems contribute to ensuring a stable and sufficient food supply for the growing global population.

In summary, plant leaf disease detection systems offer a range of benefits that enhance disease management practices, increase agricultural productivity, and promote sustainable farming methods. Their ability to provide early, accurate, and objective disease diagnoses plays a crucial role in minimizing crop losses, optimizing resource use, and ensuring food security.

## 2.4. Related Work

A paper by (Fayyaz et al., 2020) discusses the urgent need to predict and manage plant diseases in agriculture, particularly at an early stage, to ensure food security for a growing population. The authors propose leveraging cutting-edge technologies to address leaf diseases in tomato plants using machine learning and image processing techniques. The study aims to raise awareness among farmers about these advanced tools to mitigate plant diseases effectively. The key focus is on detecting diseases in tomato leaves by employing accurate algorithms and innovative methodologies.

The research is organized into three main stages: identity, analysis, and verification. Initially, tomato leaf samples with disorders are collected and processed. The images are resized and subjected to Histogram Equalization to enhance their quality. K-means clustering is utilized to segment the data space into Voronoi cells, aiding in the identification of early-stage disease symptoms. The boundary of leaf samples is extracted through contour tracing. Multiple descriptors, including Discrete Wavelet Transform, Principal Component

Analysis, and Grey Level Co-occurrence Matrix, are employed to extract informative features from the leaf samples. The model is created using the IP and ML methods for leaf disease detection described in this section. The proposed model (DWT+PCA+GLCM+CNN) for leaf disease diagnosis utilising computer vision and machine learning methodologies is displayed in Figure-1.

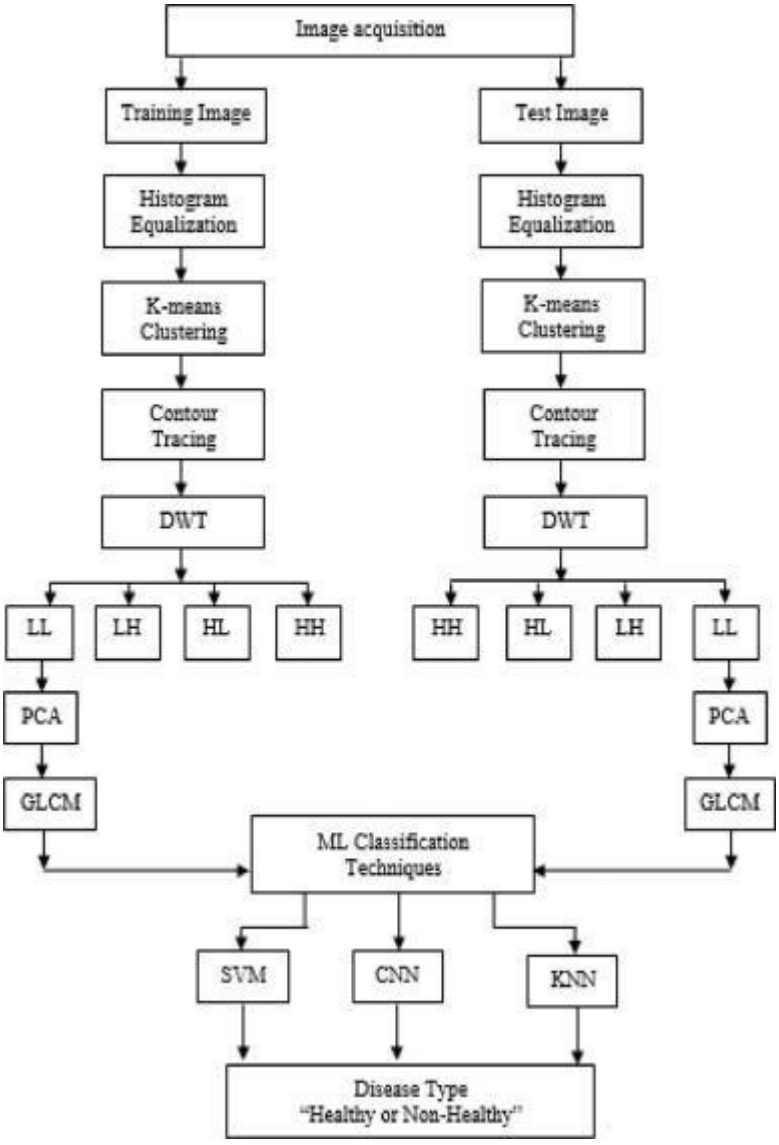


**Figure-1: IP and ML based approach for plant disease detection (Fayyaz et al., 2020)**

These features are then classified using machine learning techniques such as Support Vector Machine (SVM), Convolutional Neural Network (CNN), and K-Nearest Neighbor (K-NN). The accuracy of the model is evaluated, yielding SVM (88%), K-NN (97%), and CNN (99.6%) accuracy rates on tomato disordered samples.

The authors highlight the challenges associated with leaf disease detection, including the quality of leaf images, availability of datasets, noisy data, segmentation difficulties, variations in leaf color due to environmental factors, and the complexity of identifying different diseases in various plants. They emphasize that their proposed model addresses these challenges by combining image processing and machine learning techniques to achieve enhanced accuracy in disease detection.

A review article by (Saleem et al., 2019) discusses the significance of early identification of plant diseases and how Deep Learning (DL), a subset of Machine Learning (ML), has shown great potential in enhancing accuracy for disease detection and classification. The authors highlight that while many ML models have been used for this purpose, the advancements in DL have introduced new possibilities. Various DL architectures have been developed and modified to detect and classify plant disease symptoms, accompanied by visualization techniques and performance evaluation metrics.

The review emphasizes the use of performance metrics for evaluating DL algorithms, including top-1%/top-5% error, precision and recall, F1 score, training/validation accuracy and loss, and classification accuracy. The implementation process of DL models is depicted, starting from dataset collection to visualization mappings.

The paper aims to provide a comprehensive explanation of DL models employed for visualizing plant diseases, highlighting their potential in early disease detection. Implementing Deep Learning (DL) models for plant disease detection involves several key steps, as illustrated in Figure-2. The process begins with collecting a dataset of plant leaf images showcasing various diseases. This dataset is then split into training and validation subsets. DL models are trained using the training data, either from scratch or through transfer learning. Training and validation plots monitor the model's progress. Performance metrics evaluate the model's accuracy in classifying disease types. Finally, visualization techniques are applied to analyze the model's behavior and accurately detect and classify disease images. This systematic approach ensures effective and accurate plant disease detection using DL techniques.

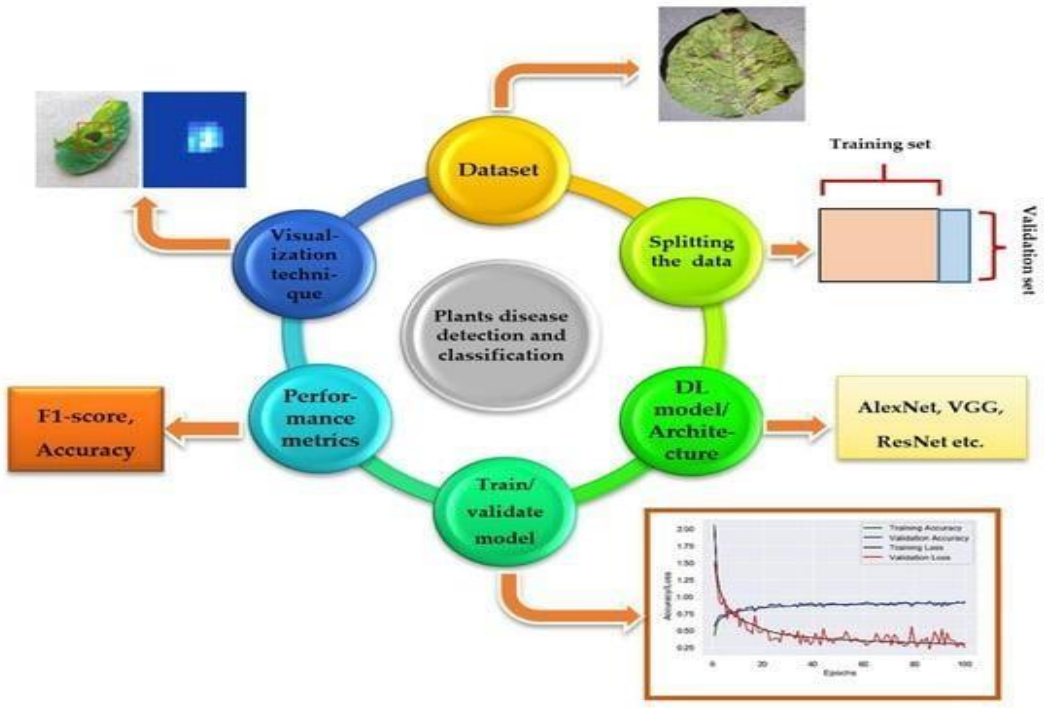


**Figure-2: Implementing Deep Learning model for Plant Disease Detection and Classification (Saleem et al., 2019)**

Another study by (Agarwal et al., 2020) focuses on tomato crop disease identification, crucial for timely treatment. Leaves often exhibit disease indicators. Machine Learning (ML) algorithms and Convolutional Neural Network (CNN) models have been explored for this purpose. CNN models, part of Deep Learning, differ from traditional ML algorithms and can be computationally intensive. The paper presents a simplified CNN model with 8 hidden layers, outperforming traditional ML and pretrained models. Using the PlantVillage dataset with 39 classes, including 10 tomato disease classes, the proposed model achieves an accuracy of 98.4%, surpassing traditional k-NN (94.9%) and VGG16 pretrained models (93.5%). Image pre-processing and augmentation further enhance the proposed model's performance, also demonstrating high accuracy (98.7%) on other datasets beyond PlantVillage.

Another study by (Liang et al., 2019) introduces a robust image-based network called PD2SE-Net for plant disease diagnosis and severity estimation. This network incorporates a residual structure and shuffle units to address common challenges in the agriculture sector. The paper aims to create an improved and practical diagnosis system for plant diseases by simultaneously handling disease diagnosis and severity estimation. The proposed PD2SE-Net50 model employs ResNet50 architecture as its foundation and shuffle units as auxiliary structures. It achieves remarkable performance in disease severity estimation (accuracy of 0.91), plant species recognition (accuracy of 0.99), and plant disease classification (accuracy of 0.98). The network utilizes data augmentation and visualization of convolutional neural networks (CNNs) to enhance accuracy and streamline hyper-parameter selection during training.

The paper titled "Plant Disease Detection Using Image Processing and Machine Learning," authored by (Kulkarni et al., 2021), presents a system for plant disease detection. The authors employ a combination of digital image processing

and the Random Forest machine learning algorithm. Their developed system achieves an average accuracy of 93% and an F1 score of 0.93 in plant disease detection. Notably, the proposed system is computationally efficient due to its utilization of statistical image processing techniques in conjunction with the machine learning model.

Similarly, another paper titled "Leaf Disease Detection and Classification by Decision Tree," authored by (Rajesh et al., 2020), introduces an innovative approach for identifying and classifying leaf diseases. The proposed system employs a decision tree algorithm, which enables accurate disease detection and classification while also significantly reducing the required processing time when compared to existing methods. By leveraging the decision tree technique, the authors have developed a streamlined and efficient framework for addressing the challenges associated with leaf disease detection. This method not only improves the accuracy of disease identification but also enhances the overall efficiency of the diagnostic process. The utilization of a decision tree as the foundation of this system showcases the potential of machine learning algorithms in revolutionizing the field of plant disease detection and classification, offering promising results that can contribute to the advancement of agricultural practices.

All these works contributing to the development of advance and accurate plant or plant leaf disease detection frameworks have been summarized in Table-1.

| Authors | Main Focus & Contribution | Methodology & Results |
|---|---|---|
| Fayyaz et al., 2020 | Detection of tomato leaf diseases using machine learning and image processing techniques. | Proposed model combines IP and ML techniques for leaf disease detection. Achieved SVM (88%), K-NN (97%), CNN (99.6%) |

| | | |
|---|---|---|
| | | accuracy on tomato disordered samples. |
| Saleem et al., 2019 | Comprehensive review of Deep Learning (DL) models for visualizing plant diseases. | Detailed explanation of DL implementation process, covering dataset collection, training, validation, metrics, and visualization. |
| Agarwal et al., 2020 | Identification of tomato crop diseases using simplified CNN model. | Proposed CNN model with 8 hidden layers outperforms traditional ML and pretrained models. Achieved 98.4% accuracy on PlantVillage dataset. |
| Liang et al., 2019 | Introduction of PD2SE-Net for plant disease diagnosis and severity estimation. | PD2SE-Net50 model integrates ResNet50 architecture and shuffle units, achieving high accuracy in disease severity, species recognition, and classification. |
| Kulkarni et al., 2021 | Plant disease detection using digital image processing and Random Forest. | Developed system with digital image processing and Random Forest achieves 93% accuracy and 0.93 F1 score in plant disease detection. |
| Rajesh et al., 2020 | Leaf disease detection and classification using decision tree algorithm. | Proposed decision tree-based approach improves disease detection accuracy and reduces processing time. |

**Table-1: Summary of discussed works**

## 2.5. Research Gap and Transfer Learning

While the literature review highlights the application of machine learning and deep learning techniques in plant disease detection, there appears to be an opportunity to investigate the effectiveness of transfer learning specifically within the context of deep learning models for plant leaf disease detection. Transfer learning is a technique that involves leveraging pre-trained models on large datasets for tasks similar to the target task. In the realm of plant leaf disease detection, transfer learning can potentially offer several advantages:

- **Limited Data Availability**: Often, collecting a large and diverse dataset for training deep learning models from scratch can be challenging due to factors like plant species variation and disease prevalence. Transfer learning allows researchers to benefit from existing pre-trained models that have learned generic features from large datasets, which can be fine-tuned on smaller plant disease datasets.

- **Reduced Training Time and Computational Resources**: Training deep learning models from scratch can be computationally expensive and time-consuming. Transfer learning allows researchers to utilize pre-trained models as a starting point, reducing the training time and computational resources required.

- **Enhanced Generalization**: Transfer learning's ability to learn relevant features from diverse datasets can potentially enhance the generalization of deep learning models to different plant species and disease types.

- **Addressing Class Imbalance**: Plant disease datasets often suffer from class imbalance, with certain diseases being less common. Transfer learning can assist in learning discriminative features even for less prevalent diseases by leveraging knowledge from other related tasks.

# 3. METHODOLOGY

For developing a system for detecting plant leaf disease, Transfer Learning in fusion with EfficientNetB3 has been used in the ML-based backend. The methodology involves implementing a robust ML model through Transfer Learning and EfficientNetB3, followed by the development of a user-friendly web interface using Flask, as shown in Figure-3. The integration phase ensures seamless communication between the model and the user interface, culminating in a comprehensive plant leaf disease detection system that offers accurate predictions and an intuitive user experience.

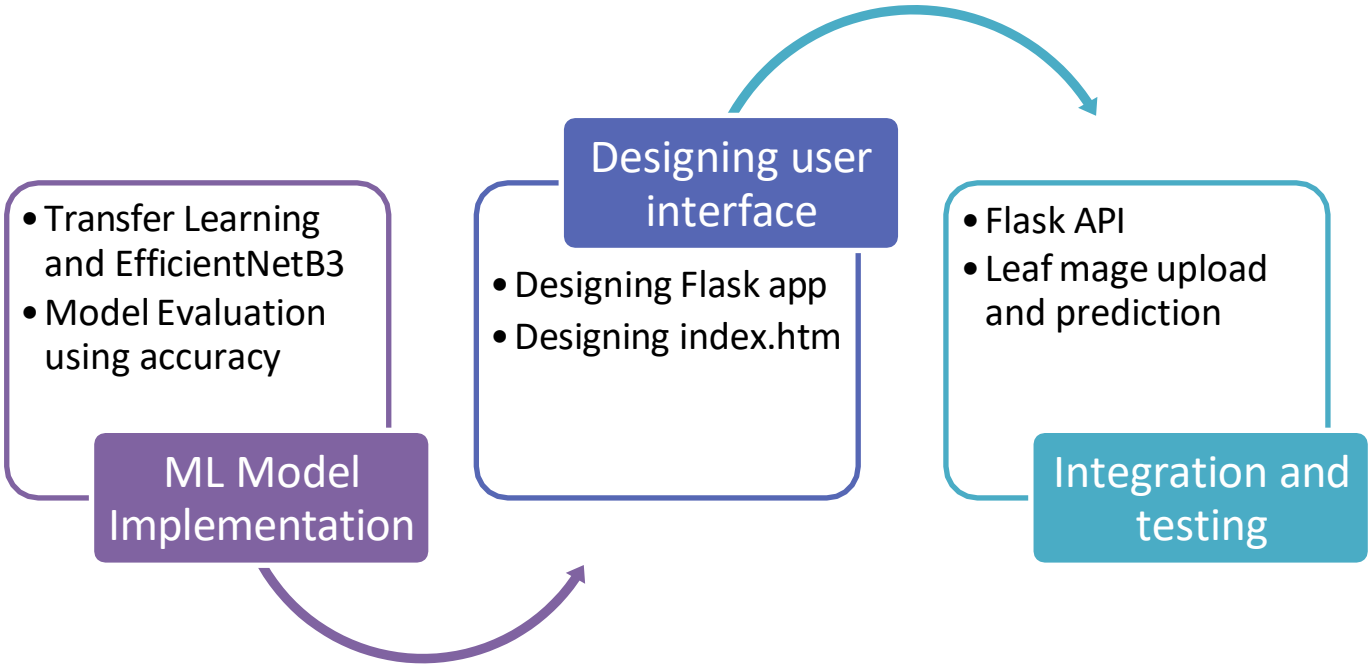


**Figure-3: Methodology**

**ML Model Implementation:**

Transfer Learning and the EfficientNetB3 architecture are adopted for building the machine learning model. Transfer Learning leverages pre-trained models, enhancing the model's efficiency and accuracy. EfficientNetB3, a convolutional neural network, serves as the foundation for feature extraction and disease classification. The model's effectiveness is evaluated through accuracy metrics, quantifying its ability to correctly classify diseases.

Transfer Learning is a machine learning technique that leverages knowledge gained from training a model on one task to improve the performance of a related task. Instead of training a model from scratch, transfer learning starts with a pre-trained model that has already learned representations of features from a large dataset, typically from a different but related problem. This pre-trained model is then fine-tuned on the target task using a smaller dataset specific to that task.

Transfer learning is particularly beneficial when working with limited data, as it allows the model to benefit from the general patterns and features it learned from the source task's dataset. This helps accelerate the training process and can lead to improved performance on the target task. Transfer learning is widely used in various domains, including computer vision, natural language processing, and more.

EfficientNetB3 is a specific neural network architecture that gained attention for its efficiency and effectiveness in image recognition tasks. It belongs to the EfficientNet family of models, which are designed to balance model size, computational resources, and accuracy. EfficientNetB3 is an enhancement of the original EfficientNet architecture, offering improved performance while still being computationally efficient.

**Designing User Interface:**

The user interface development involves creating a Flask web application. Flask, a Python web framework, facilitates the creation of dynamic and user-friendly applications. The design process includes crafting the user interface's front-end using HTML and CSS. The Flask app provides a platform for users to interact with the ML model seamlessly.

**Integration and Testing:**

Integration of components is realized through the Flask API. This interface enables communication between the ML model and the user interface. Users can upload leaf images, and the Flask app orchestrates the prediction process using the ML model. The integration is rigorously tested to ensure smooth functionality, accurate predictions, and a user-friendly experience.

## 3.1. SDLC Model – Waterfall Model

The Waterfall model is a traditional and linear approach to software development that follows a structured sequence of phases. Each phase must be completed before moving on to the next, making it a well-defined and sequential process. This model is particularly suitable when the project's requirements are clear, fixed, and unlikely to change significantly throughout development. Here are the key phases of the Waterfall model shown in Figure-4.

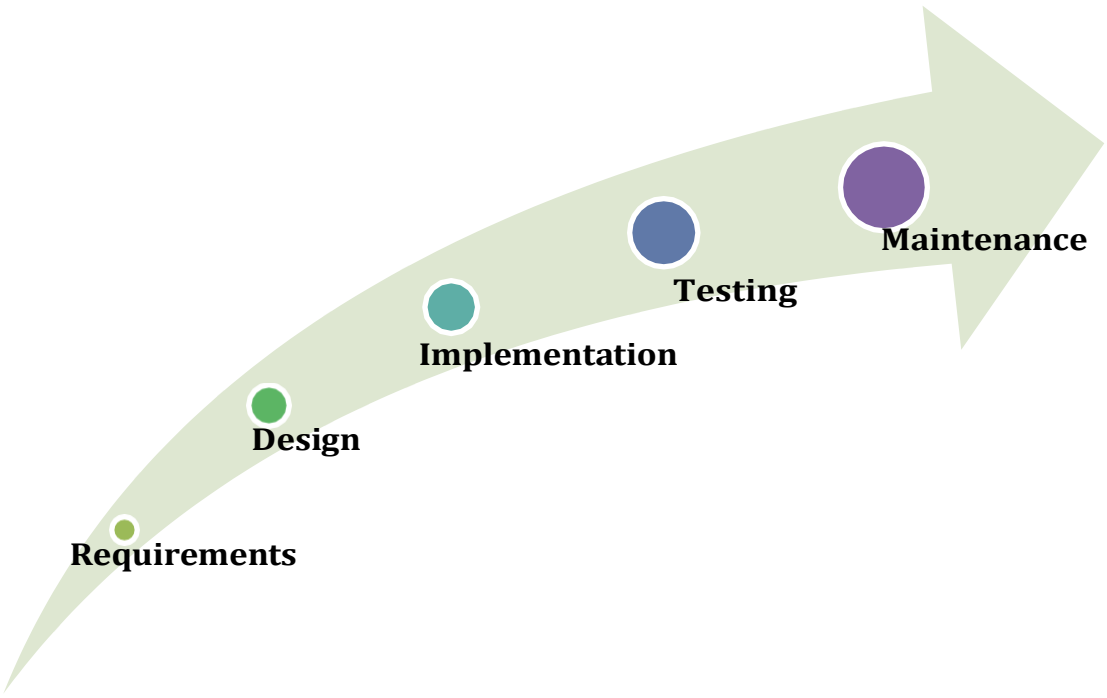


**Figure-4: Waterfall Model Phases**

**Requirements**: In this phase, the project's requirements are gathered from stakeholders, users, and other sources. A detailed understanding of the system's functionality, features, and constraints is established. The requirements are documented in a clear and comprehensive manner to serve as a foundation for the subsequent phases. This involves understanding the goals of the plant leaf disease detection system, the features it needs to have, and any specific constraints or preferences.

**Design**: Based on the gathered requirements, the system's architecture and design are planned. This phase involves creating high-level and low-level designs that outline the structure of the software, user interfaces, databases, and other components. Design documentation is produced to guide the implementation phase. This includes creating a design for the user interface, deciding on the architecture of the system (including the use of Transfer Learning and EfficientNetB3), and planning the integration of the Flask app. The designs

for the user interface, the user experience, and the overall flow of the system are also defined.

**Implementation**: This is where the actual coding and development of the software take place. The designs from the previous phase are translated into executable code. Programmers write the code, integrate different modules, and ensure that the software is functioning according to the design specifications. This phase involves developing the Flask app, integrating the machine learning model, setting up the user interface, and ensuring that all components are working together as intended.

**Testing**: Once the implementation is complete, thorough testing is conducted to ensure that the software meets the specified requirements and functions correctly. Testing includes various levels such as unit testing, integration testing, system testing, and user acceptance testing. Defects are identified, reported, and fixed in this phase. Each component of the system, including the Flask API, image upload functionality, prediction accuracy, and user interface responsiveness are tested.

**Maintenance**: Even after deployment, the application may require updates, bug fixes, and enhancements. The maintenance phase involves addressing any issues that arise, updating the software to meet changing requirements, and ensuring its long-term functionality.

The Waterfall model's structured approach offers several benefits, such as clear documentation at each stage, reduced ambiguity, and a well-defined progression from one phase to the next. It can be particularly useful for projects with stable and predetermined requirements, and it provides a clear framework for planning and execution.

However, the Waterfall model has limitations too. Its rigid nature makes it less adaptable to changes in requirements, and it may not be ideal for projects where stakeholder feedback and iteration are essential. Additionally, since testing is conducted at the end, any issues discovered during testing could require significant rework of earlier phases.

Overall, while the Waterfall model has its strengths, it's important to carefully consider the nature of your project and whether its characteristics align with the model's sequential approach before deciding to use it.

# 4. DESIGN & IMPLEMENTATION

## 4.1. Data Flow Diagram

A Data Flow Diagram (DFD) is a graphical representation that illustrates the flow of data within a system or process. It provides a clear and concise visual depiction of how data moves from one component or stage to another, showing the relationships and interactions between different elements of the system. DFDs are commonly used in system analysis and design to model and document the data flow and transformations that occur within a complex system.

At its core, a DFD consists of processes, data stores, data flows, and external entities. These elements are interconnected to show how data is input, processed, stored, and output within the system. Processes represent the actions or activities that transform data, while data stores depict the locations where data is stored and retrieved. Data flows are the pathways through which data moves from one element to another, and external entities are the sources or destinations of data outside the system boundaries.

The beauty of a DFD lies in its simplicity and clarity. It abstracts away unnecessary technical details, focusing solely on the movement of data and the interactions between system components. This makes DFDs valuable tools for communication and collaboration among stakeholders, including analysts, designers, developers, and users. They provide a common language for discussing how a system works and enable stakeholders to identify potential bottlenecks, inefficiencies, and opportunities for improvement.

DFDs come in various levels of detail, allowing analysts to zoom in on specific areas of interest or zoom out for a high-level overview of the entire system. They can be used to model a wide range of processes and systems, from business processes to software applications. As a system evolves, DFDs can be updated and refined to accurately reflect any changes in data flow and process interactions.

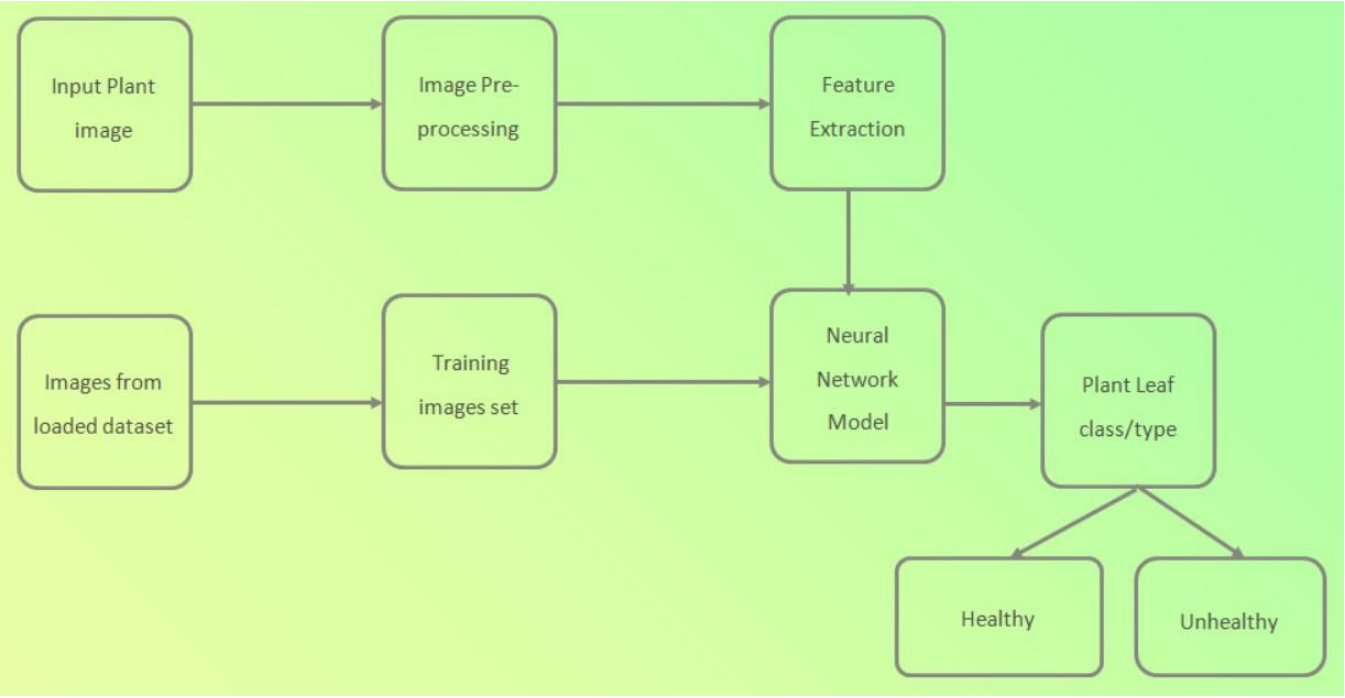


**Figure-5: DFD for Plant Disease Detection System**

In this Level 0 DFD shown in Figure-5, the flow of data starts with the user providing an input plant image. The image goes through pre-processing to enhance its quality. Feature extraction is then performed to identify relevant characteristics of the image.

Images from a loaded dataset are used to train the neural network model. The trained model can then classify the input plant image into different classes/types, such as "Healthy" or "Unhealthy."

The final classification result is provided as an output to the user. This DFD represents the high-level flow of data through the various stages of the plant leaf disease detection system. Keep in mind that this is a simplified representation, and the actual system may involve more detailed processes and interactions.

## 4.2. ML Model Implementation

The PlantVillage dataset is used in this model which is downloaded from Kaggle. The dataset encompasses more than 50,000 meticulously curated images. These images showcase both healthy and infected leaves of various crop plants. The images have been made available through the PlantVillage online platform, which serves as the host for this valuable dataset. The ML implementation goes through the following main steps:

1. **Importing Dependencies and Installing Libraries:**

- The code starts by installing TensorFlow version 2.9.1 using !pip install tensorflow==2.9.1.
- It also installs required packages and dependencies.

2. **Uploading Kaggle Dataset to Google Colab:**

- The code creates a directory named .kaggle and copies a Kaggle API token (kaggle.json) into that directory. This is used for authenticating with Kaggle to download datasets.
- The !kaggle datasets download command is used to download the "plantvillage-dataset" from Kaggle. This dataset presumably contains images of various plants for disease detection.

3. **Extracting Zip Files - PlantVillage Dataset:**

- The downloaded dataset is a zip file named "plantvillage-dataset.zip".

- The code uses the zipfile module to extract the contents of this zip file.

4. **Importing Required Modules and Libraries:**

- Several modules and libraries are imported to be used later in the code, such as TensorFlow, Keras, Matplotlib, NumPy, OpenCV, etc.
- Functions to Create Data Frame from Dataset:

5. **The code defines three functions to organize and process the dataset:**

- define_paths(img_dir): This function generates a list of file paths and corresponding labels for each image in the dataset.
- define_df(files, classes): This function combines file paths and labels into a Pandas DataFrame.
- split_data(img_dir): This function splits the data into training, validation, and test dataframes.

6. **Creating Data Generators:**

- The code creates data generators using the ImageDataGenerator class. Data generators help preprocess and feed data to the model during training.
- It defines functions for data augmentation and creates generators for training, validation, and testing data.

7. **Displaying Sample Images:**

- The code defines a function show_images(gen) to display a grid of sample images from the data generator.

8. **Splitting Data and Generating Data Generators:**

- The dataset is split into training, validation, and test data using the split_data function.
- Data generators for training, validation, and testing are created using the create_gens function.

**9. Building the Model:**

- The code uses the EfficientNetB3 architecture as the base model for transfer learning. It loads the pre-trained weights from the "imagenet" dataset.
- The base model's layers are set to non-trainable to keep their pre-trained weights.
- Additional layers are added on top of the base model to create the complete model architecture.
- The model is compiled with a loss function, optimizer, and evaluation metric.

**10. Training the Model:**

- The code defines callbacks for early stopping and model checkpointing.
- The model is trained using the fit function, and training progress is monitored using the provided callbacks.
- The training history (loss and accuracy) is saved for visualization.

**11. Visualizing Training Progress:**

- The code uses Matplotlib to plot the training and validation accuracy and loss curves.

**12. Evaluating the Model:**

- The trained model is evaluated on the training, validation, and test data using the evaluate method.
- The loss and accuracy metrics are displayed.

**13. Making Predictions:**

- The model is used to predict labels for the test data.
- The predictions are compared against the true labels to calculate accuracy using accuracy_score.

**14. Saving and Loading the Model:**

- The trained model is saved to a file using the save method.
- A saved model is loaded back into memory using load_model.

**15. Predicting Using the Saved Model for evaluation:**

- An example image is loaded and preprocessed. The saved model is then used to make predictions on this image.

The code provides an end-to-end pipeline for building, training, evaluating, and using a plant disease detection model using a pre-trained CNN architecture. It covers data handling, model building, training, evaluation, and prediction stages.

```
#Installingtensorflow 2.9.1 version
!pip install tensorflow==2.9.1

#Upload kaggle dataset directly into notebook
!mkdir -p ~/.kaggle
!cpkaggle.json ~/.kaggle/



!kaggle datasets download -d abdallahalidev/plantvillage-dataset

#Extract zip files - plantvillage dataset
importzipfile
zip_ref = zipfile.ZipFile('/content/plantvillage-dataset.zip','r')
zip_ref.extractall('/content')
zip_ref.close()

#import modules
```

```
fromkeras.layers import Input, Lambda, Dense, Flatten
fromkeras.models import Model
importtensorflow as tf
fromtensorflow import keras
importtensorflow as tf
importmatplotlib.pyplot as py
from keras.applications.resnet_v2 import ResNet152V2
from keras.applications.mobilenet_v2 import MobileNetV2
from keras.applications.inception_v3 import InceptionV3
fromkeras.applications.efficientnet import EfficientNetB3,preprocess_input
from keras.applications.inception_v3 import preprocess_input
fromkeras.preprocessing import image
fromkeras.preprocessing.image import ImageDataGenerator
fromkeras.utils import load_img,img_to_array
fromkeras.models import Sequential
fromkeras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Activation,
Dropout,  BatchNormalization
fromkeras import regularizers
importnumpy as np
from glob import glob
import pandas as p
importos
import cv2
fromkeras.models import load_model
importshutil
fromsklearn.model_selection import train_test_split
# from PIL import image
fromsklearn.metrics import accuracy_score
```

```
# Functions to Create Data Frame from Dataset
# Generate data paths with labels
defdefine_paths(img_dir):
filepaths = [] #creating empty list
labels = []

folds = os.listdir(img_dir) # list directories inside img_dir
for fold in folds:
foldpath = os.path.join(img_dir, fold) #concatenate each dir inside img_dir with
img_dir and storing them in foldpath
filelist = os.listdir(foldpath) #now access image files
forfiles_l in filelist:
fpath = os.path.join(foldpath, files_l)
filepaths.append(fpath) #append those image directories in filepaths
labels.append(fold)# append folder name which are their corresponding label

returnfilepaths, labels

# Concatenate data paths with labels into one dataframe ( to later be fitted into
the model )
defdefine_df(files, classes):
    F = p.Series(files, name= 'file_dir') #creating series of file paths
    L = p.Series(classes, name='labels') #creating series of labels
returnp.concat([F, L], axis= 1)

# Split dataframe to train, valid, and test
defsplit_data(img_dir):
```

```
    # train dataframe
files, classes = define_paths(img_dir) #calling define_paths function which will return file_paths and labels
df = define_df(files, classes) #calling define_df function
strat = df['labels'] #using stratify to uniformly distribute instances of classes in both dataframe , i.e. train and dummy to prevent model from getting biased towards any particular class.
train_df, dummy_df = train_test_split(df,  train_size= 0.8, shuffle= True, random_state= 123, stratify= strat)

    # valid and test dataframe
strat = dummy_df['labels']
valid_df, test_df = train_test_split(dummy_df,  train_size= 0.5, shuffle= True, random_state= 123, stratify= strat)

returntrain_df, valid_df, test_df

'''
    This function takes train, validation, and test dataframe and fit them into image data generator, because model takes data from image data generator.
    Image data generator converts images into tensors. '''
defcreate_gens (train_df, valid_df, test_df, batch_size):

img_size = (300, 300) # setting it according to image size used in pretrained model
channels = 3
color = 'rgb'
img_shape = (img_size[0], img_size[1], channels) #defines that image has 300 X
```

```
300 pixels dimension and RGB system



ts_length = len(test_df)
test_batch_size = max(sorted([ts_length // n for n in range(1, ts_length + 1) if
ts_length%n == 0 and ts_length/n <= 80]))

test_steps = ts_length // test_batch_size

    # This function which will be used in image data generator for data
augmentation, it just take the image and return it again.
def scalar(img):
returnimg

tr_gen = ImageDataGenerator(preprocessing_function= scalar,horizontal_flip=
True)
ts_gen = ImageDataGenerator(preprocessing_function= scalar)

train_gen = tr_gen.flow_from_dataframe(train_df, x_col= 'file_dir', y_col= 'labels',
target_size= img_size, class_mode= 'sparse',
color_mode= color, shuffle= True, batch_size= batch_size)

valid_gen = ts_gen.flow_from_dataframe(valid_df, x_col= 'file_dir', y_col=
'labels', target_size= img_size, class_mode= 'sparse',
color_mode= color, shuffle= True, batch_size= batch_size)

test_gen = ts_gen.flow_from_dataframe(test_df, x_col= 'file_dir', y_col= 'labels',
target_size= img_size, class_mode= 'sparse',
```

```
color_mode= color, shuffle= False, batch_size= test_batch_size)

returntrain_gen, valid_gen, test_gen

defshow_images(gen):

    # return classes , images to be displayed
g_dict = gen.class_indices        # defines dictionary {'class': index}
classes = list(g_dict.keys())     # defines list of dictionary's kays (classes), classes
names : string
images, labels = next(gen)        # get a batch size samples from the generator

    # calculate number of displayed samples
length = len(labels)        # length of batch size
sample = min(length, 25)    # check if sample less than 25 images

py.figure(figsize= (20, 20))

for i in range(sample):
py.subplot(5, 5, i + 1)
image = images[i] / 255       # scales data to range (0 - 255)
py.imshow(image)
index = int(labels[i])  # get image index
class_name = classes[index]   # get class of image
py.title(class_name, color= 'black', fontsize= 8)
py.axis('off')
py.show()
```

```
data_dir = '/content/plantvillage dataset/segmented'

try:
    # Get splitted data
train_df, valid_df, test_df = split_data(data_dir)

    # Get Generators
batch_size = 40
train_gen, valid_gen, test_gen = create_gens(train_df, valid_df, test_df,
batch_size)

except Exception as e:
print('Error:', e)

train_gen[0]

train_df

#Calling function to display images
show_images(train_gen)

#Getting number of output classes
class_count = len(list(train_gen.class_indices.keys()))

#Defining pretrainedmodel ,  Model used - EfficientNetB3
base_model = tf.keras.applications.efficientnet.EfficientNetB3(include_top=
False, weights= "imagenet", input_shape= (300,300,3), pooling= 'max')
```

```
#Using model's trained parameters without changing it
for layer in base_model.layers:
layer.trainable = False

#Creating model
new_model = Sequential()
new_model.add(base_model)
new_model.add(Flatten())
new_model.add(BatchNormalization(axis= -1, momentum= 0.99, epsilon=
0.001))
new_model.add(Dense(256, kernel_regularizer= regularizers.l2(l= 0.016),
activity_regularizer= regularizers.l1(0.006),
                activation= 'relu'))
new_model.add(Dropout(rate= 0.45, seed= 123))
new_model.add(Dense(class_count, activation= 'softmax'))



new_model.compile(loss='sparse_categorical_crossentropy',optimizer=tf.keras.o
ptimizers.Adam(0.0005),metrics=['accuracy'])

new_model.summary()

#Set callbackparameter , saving best parameter in final_model_weights1.hdf5
checkpointer = [tf.keras.callbacks.EarlyStopping(monitor = 'val_accuracy',
verbose = 1, restore_best_weights=True, mode="max",patience = 10),
tf.keras.callbacks.ModelCheckpoint(
filepath='final_model_weights1.hdf5',
monitor="val_accuracy",
```

```
verbose=1,
save_best_only=True,
                mode="max")]

#implementearlystopping
steps_per_epoch = train_gen.n // train_gen.batch_size
validation_steps = valid_gen.n // valid_gen.batch_size
history = new_model.fit(x=train_gen,
validation_data=valid_gen,
epochs=25,
callbacks=[checkpointer],
steps_per_epoch=steps_per_epoch,
validation_steps=validation_steps,shuffle = False)

#plottting accuracy curve for training and validation data
py.plot(history.history['accuracy'],color='blue',label='train')
py.plot(history.history['val_accuracy'],color='red',label='validation')
py.legend()
py.show()

#plottting loss curve for training and validation data
py.plot(history.history['loss'],color='blue',label='train')
py.plot(history.history['val_loss'],color='red',label='validation')
py.legend()
py.show()

#Evaluate model
test_batch  = 32;
```

```
train_eval = new_model.evaluate(train_gen,steps = test_batch,verbose=1)
#checking for loss and accuracy for training data
val_eval = new_model.evaluate(valid_gen,steps =
test_batch,verbose=1)#checking for loss and accuracy for validation data
test_eval = new_model.evaluate(test_gen,steps =
test_batch,verbose=1)#checking for loss and accuracy for testing data
print("Train Loss: ", round(train_eval[0]*100,2))
print("Train Accuracy: ", round(train_eval[1]*100,2))
print('-' * 20)
print("Validation Loss: ", round(val_eval[0]*100,2))
print("Validation Accuracy: ", round(val_eval[1]*100,2))
print('-' * 20)
print("Test Loss: ", round(test_eval[0]*100,2))
print("Test Accuracy: ",round(test_eval[1]*100,2))

#Making Predictions
preds = new_model.predict(test_gen)

# print(y_pred)

preds

a=np.argmax(preds, axis=1)

a

test_labels = test_gen.classes
```

```
#checking accuracy score for test generators
accuracy_score(test_labels,a)

test_labels[2]

#Saving Model
new_model.save('Plant_Detection_model_final.h5')

#Loading saved model
model = load_model('Plant_Detection_model_final.h5')

img_path = '/content/download (1).jpg' #storing path of img to be predicted

img = load_img(img_path,target_size = (300,300))

img

x = img_to_array(img) #converting image to array

x = np.expand_dims(x,axis=0) #expanding dimension of image to 4D array
(batch_size,img_height,img_width,channels)

x

x = preprocess_input(x) #processing image according pretrained model

pred = model.predict(x) #predicting the new image
```

```
pred

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

p[0]
```

## 4.3. Flask Code for UI

```
import os
from flask import Flask, request, redirect, render_template, jsonify
from werkzeug.utils import secure_filename
import numpy as np
from keras.preprocessing.image import load_img, img_to_array
import tensorflow as tf

app = Flask(_name_)

UPLOAD_FOLDER = 'D:/PlantDiseaseDetect/uploads'
ALLOWED_EXTENSIONS = {'jpg', 'jpeg', 'png'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

class_labels = [
    "Apple__Apple_scab", "Apple_Black_rot", "Apple__Cedar_apple_rust",
    "Apple__healthy", "Blueberry_healthy",
```

```
"Cherry(including_sour) ___Powdery_mildew",
    "Cherry_(including_sour) __healthy", "Corn(maize)___Cercospora_leaf_spot
Gray_leaf_spot",
    "Corn_(maize)__Common_rust", "Corn_(maize)___Northern_Leaf_Blight",
    "Corn_(maize)__healthy", "Grape_Black_rot", "Grape_Esca(Black_Measles)",
    "Grape__Leaf_blight(Isariopsis_Leaf_Spot)", "Grape__healthy",
"Orange_Haunglongbing(Citrus_greening)",
    "Peach__Bacterial_spot", "Peach_healthy", "Pepper,_bell__Bacterial_spot",
    "Pepper,bell_healthy", "Potato_Early_blight", "Potato__Late_blight",
    "Potato__healthy", "Raspberry_healthy", "Soybean_healthy",
"Squash__Powdery_mildew",
    "Strawberry__Leaf_scorch", "Strawberry_healthy", "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_Yellow_Leaf_Curl_Virus", "Tomato_Tomato_mosaic_virus",
"Tomato__healthy"
]

model = None

def load_model():
    global model
    model = tf.keras.models.load_model('Plant_Detection_model_final.h5')

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS
```

```
@app.route('/', methods=['GET', 'POST'])
def index():
    if request.method == 'POST':
        if 'file' not in request.files:
            return redirect(request.url)

        file = request.files['file']
        if file.filename == '':
            return redirect(request.url)

        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
            file.save(file_path)

            img = load_img(file_path, target_size=(300, 300))
            img_array = img_to_array(img)
            img_array = np.expand_dims(img_array, axis=0)
            img_array = tf.keras.applications.efficientnet.preprocess_input(img_array)
            prediction = model.predict(img_array)
            predicted_class_index = np.argmax(prediction)
            predicted_class = class_labels[predicted_class_index]
            confidence = prediction[0][predicted_class_index] * 100

            return jsonify({
                'predicted_class': predicted_class,
                'confidence': confidence
```

```
        })

    return render_template('index.html')

if _name_ == '_main_':
    load_model()
    app.run(host='127.0.0.1', port=5001)
```

To begin, the code imports essential libraries that are crucial for constructing the web application. These libraries include Flask, which is essential for creating the web server, numpy for numerical computations, Keras for image preprocessing, and TensorFlow for machine learning tasks.

The Flask application is initialized with the creation of an instance named app. Furthermore, the code specifies a directory named UPLOAD_FOLDER where the images uploaded by users will be stored. In addition, a set of allowed file extensions (ALLOWED_EXTENSIONS) is defined to ensure that only valid image files are processed.

In the code, there's a list named class_labels, which contains the labels corresponding to different plant disease classes and healthy states. These labels are associated with the potential predictions that the loaded model can make.

The code contains a crucial function named load_model. This function is responsible for loading a pre-trained deep learning model from a file named 'Plant_Detection_model_final.h5'. The loaded model is then stored in the variable model.

For the purpose of verifying the validity of uploaded file extensions, the code includes a function named allowed_file. This function examines whether a given

filename possesses an allowable extension, thus ensuring that only legitimate image files are processed.

The main interaction with users is established through a Flask route, namely the main page defined by @app.route('/', methods=['GET', 'POST']). When a user initiates a POST request, typically stemming from submitting an HTML form, the code handles the uploaded image. The uploaded image is loaded and prepared for analysis. This includes saving the image to the designated upload folder, loading it, preprocessing it, and feeding it into the pre-trained model for disease prediction. The result of this analysis, encompassing the predicted disease class and the confidence level of the prediction, is then returned in a JSON format.

Finally, the code concludes by initiating the Flask app to launch the web application. The load_model function is executed prior to starting the app, ensuring that the model is loaded and ready. The app commences by running on the local host (127.0.0.1) and listens on port 5001, allowing users to access the application through their web browser.

In essence, this Flask code blueprint facilitates the creation of a user-friendly web application for plant disease detection. By utilizing a pre-trained deep learning model, the application empowers users to upload images of plant leaves, receive predictions about potential diseases, and gain insights into the health status of their plants. This innovation has the potential to significantly contribute to agricultural productivity and management.

## 4.4. HTML Code for UI

```
<!DOCTYPE html>
<html>
<head>
    <title>Plant Disease Detection</title>
    <style>
        body {
            background-image: url('./static/bg2.jpg');
            background-repeat: no-repeat;
            background-size: cover;
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 50px;
            margin: 0;
        }
        h1 {
            background-color: lightpink;
            padding: 10px;
            border-radius: 10px;
        }
        form {
            display: flex;
            flex-direction: column;
            align-items: center;
            margin-top: 20px;
        }
        input[type="file"] {
            margin-bottom: 10px;
        }
```

```
        input[type="submit"] {
            background-color: #4CAF50;
            color: white;
            border: none;
            padding: 10px 20px;
            text-align: center;
            text-decoration: none;
            display: inline-block;
            font-size: 16px;
            border-radius: 5px;
            cursor: pointer;
        }
        #results {
            margin-top: 30px;
            background-color: rgba(255, 255, 255, 0.8);
            border-radius: 10px;
            padding: 20px;
            max-width: 500px;
            margin-left: auto;
            margin-right: auto;
        }
        #results p {
            margin: 10px;
        }
    </style>
</head>
<body>
    <h1>Plant Disease Detection</h1>
```

```
<form id="file-form" action="/" method="post" enctype="multipart/form-data">
    <input type="file" name="file" id="file-input">
    <input type="submit" value="Upload and Detect">
</form>
<div id="results" style="display: none;">
    <h2>Prediction Result:</h2>
    <p><strong>Leaf type:</strong> <span id="leaf-type">Upload a leaf image to detect its health</span></p>
    <p><strong>Confidence:</strong> <span id="confidence"></span>%</p>
    <p><strong>Category:</strong> <span id="category"></span></p>
</div>
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
<script>
    $('#file-form').submit(function(event) {
        event.preventDefault();
        var formData = new FormData(this);
        $.ajax({
            url: '/',
            type: 'POST',
            data: formData,
            processData: false,
            contentType: false,
            success: function(response) {
                $('#leaf-type').text(response.predicted_class);
                $('#confidence').text(response.confidence.toFixed(2));

                var category = "Unhealthy";
                var healthyLabels = [
```

```
                "Apple__healthy", "Blueberry_healthy",
"Cherry(including_sour)___healthy",
                "Corn_(maize)__healthy", "Grape_healthy", "Potato__healthy",
                "Raspberry__healthy", "Soybean_healthy", "Strawberry_healthy",
"Tomato__healthy"
            ];

            if (healthyLabels.includes(response.predicted_class)) {
                category = "Healthy";
            }

            $('#category').text(category);

            // Show the results
            $('#results').css('display', 'block');
        },
        error: function() {
            // Show the results even if there's an error
            $('#results').css('display', 'block');
        }
    });
});
</script>
</body>
</html>
```

The HTML document starts with a DOCTYPE declaration, followed by the opening <html> tag. The document's <head> section contains a <title> tag that sets the title of the web page to "Plant Disease Detection." Additionally, a <style> block is included to define the visual styling of various elements on the page.

The <style> block contains CSS rules that govern the appearance of the web page. The body element is styled to have a background image, specific font, text alignment, padding, and no margins. This gives the page a visually appealing background and text formatting.

Inside the <body> of the document, the main content of the page is structured. An <h1> heading is used to display the title "Plant Disease Detection." This heading is styled with a light pink background, padding, and rounded corners.

Following the heading, a <form> element is defined with the ID "file-form." This form serves as the input mechanism for users to upload an image file. It's set to send a POST request to the root URL ("/") upon submission. Inside the form, an <input> element of type "file" allows users to select an image file, and another <input> of type "submit" is provided for uploading and initiating the detection process.

A  with the ID "results" is included within the form. This division holds the space for displaying the prediction results, including the leaf type, confidence level, and category (healthy/unhealthy). Initially, this division is set to be hidden (display: none;) until the results are obtained.

In the <script> section at the bottom of the page, the jQuery library is loaded from an external source. This library simplifies JavaScript operations and interactions.

Within the script, a function is defined to handle the form submission. It prevents the default form submission behavior, collects the form data using the FormData

API, and sends an AJAX request to the server. The server's response contains the prediction results.

Upon a successful response from the server, the script updates the content of the "results" division with the predicted leaf type, confidence level, and category. The category is determined based on whether the predicted leaf type belongs to a set of predefined healthy labels. If the leaf type is in the healthy label set, the category is set to "Healthy." In case of an error during the AJAX request, the "results" division is still displayed, ensuring that users receive feedback even in the event of a failure.

In conclusion, this HTML code forms the foundation of the web interface for the Plant Disease Detection application. It establishes an easy-to-use form for image uploads, displays prediction results, and utilizes JavaScript and jQuery to handle form submissions and dynamically update the results section based on the server's response. The careful styling and layout ensure a visually pleasing and user-friendly experience for the application's users.

# 5. EVALUATION & RESULTS

## 5.1. ML Model Evaluation & Results

The evaluation metrics used here to evaluate the performance of the implemented ML model is Accuracy. In order to visualize the performance, training and validation accuracy and loss curves are also plotted.

The functions - define_paths(img_dir), define_df(files, classes), split_data(img_dir), and create_gens(train_df, valid_df, test_df, batch_size) – set the foundation for handling the dataset, creating data generators, and preparing the data for training a machine learning model. These functions establishes an

organized workflow that covers essential preprocessing steps required for developing an effective plant leaf disease detection system. The images present in the training set look like the following:

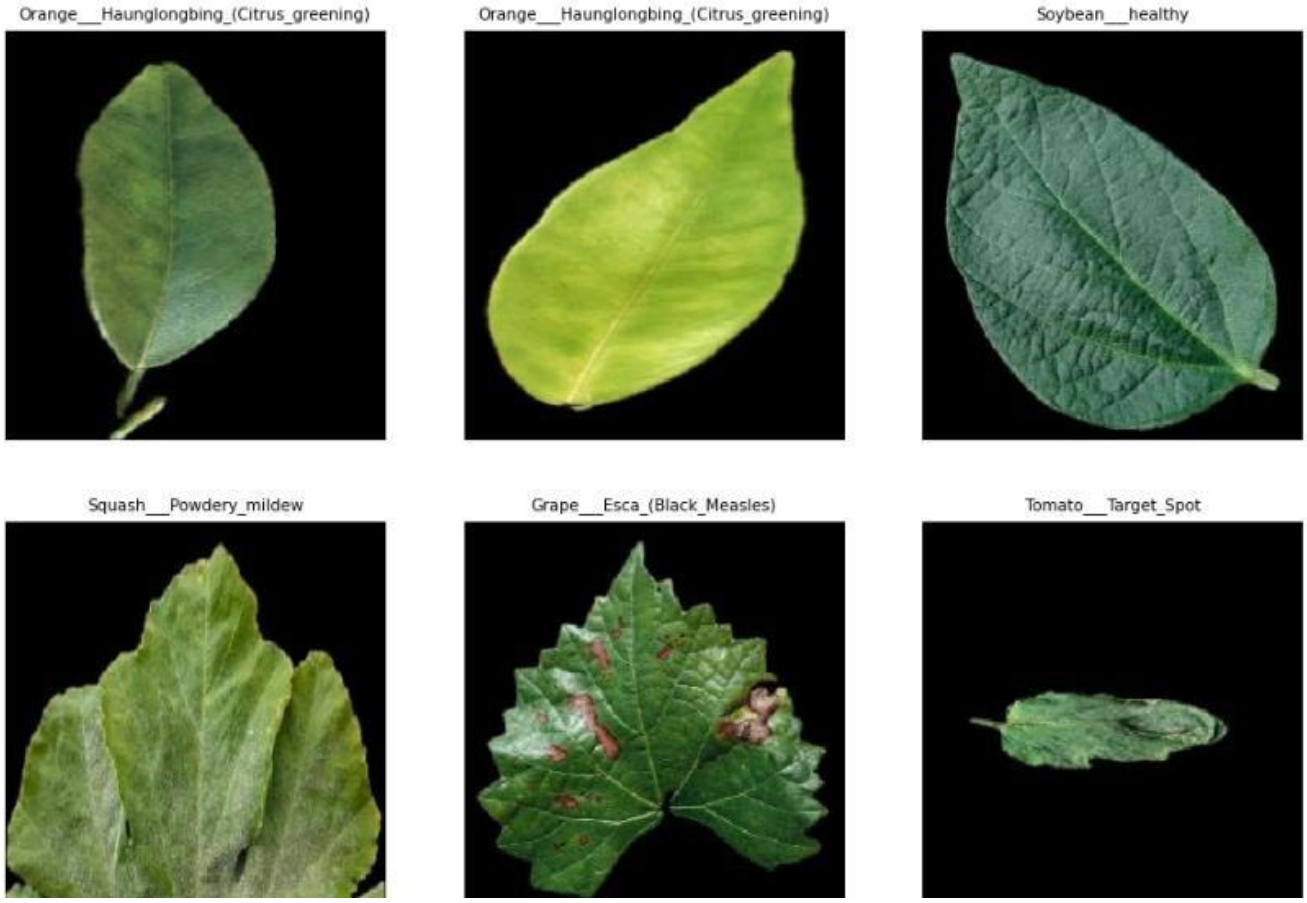


**Figure-6: train_gen images**

After fine tuning and optimizing the pre-trained EfficientNetB3 model using Adams optimizer, it is modified as shown in the new model summary given in Figure-7. Adam's optimizer, short for Adaptive Moment Estimation, is a widely used optimization algorithm in the field of machine learning and deep learning. It combines the benefits of both the Adagrad and RMSProp optimizers, offering efficient and adaptive adjustments to the learning rates of individual model parameters. Adam maintains an adaptive learning rate for each parameter by

calculating exponential moving averages of the gradient's first moment (mean) and second moment (uncentered variance). This approach helps the optimizer automatically adjust learning rates for each parameter based on the historical gradient behavior, resulting in faster convergence and improved training efficiency. With its ability to handle sparse gradients and perform well on a variety of tasks, Adam has become a popular choice for optimizing neural network models, contributing to their quicker convergence and enhanced generalization.

```
new_model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 efficientnetb3 (Functional)  (None, 1536)             10783535

 flatten (Flatten)           (None, 1536)              0

 batch_normalization (BatchN  (None, 1536)             6144
 ormalization)

 dense (Dense)               (None, 256)               393472

 dropout (Dropout)           (None, 256)               0

 dense_1 (Dense)             (None, 38)                9766

=================================================================
Total params: 11,192,917
Trainable params: 406,310
Non-trainable params: 10,786,607
_________________________________________________________________
```

**Figure-7: New Model summary**

The initial layer, labeled as "efficientnetb3," represents the EfficientNetB3 model, which serves as the backbone of the network for feature extraction. This pre-trained model has a substantial number of parameters (10,783,535), enabling it to capture intricate patterns and features from input images effectively.

Following the feature extraction layer, there is a "flatten" layer that reshapes the output from the previous layer into a one-dimensional vector. This prepares the data for the subsequent fully connected layers.

The "batch_normalization" layer plays a role in normalizing the activations of the previous layer, contributing to more stable and efficient training.

The next layer is a fully connected "dense" layer with 256 units, responsible for learning higher-level representations and patterns in the extracted features. The "dropout" layer, which follows the dense layer, helps prevent overfitting by randomly setting a fraction of the input units to zero during each training iteration.

Finally, the last "dense_1" layer contains 38 units, representing the number of different classes (plant leaf diseases) that the model can predict. The output from this layer is the predicted probability distribution across these classes.

The total number of parameters in the model is 11,192,917, out of which 406,310 are trainable parameters that the model will adjust during the training process to fit the specific dataset. The remaining parameters (10,786,607) are non-trainable, originating from the pre-trained EfficientNetB3 model.

The training and validation accuracy curves and training and validation loss curves in Figure-8 and Figure-9 respectively show the effective performance of the model.

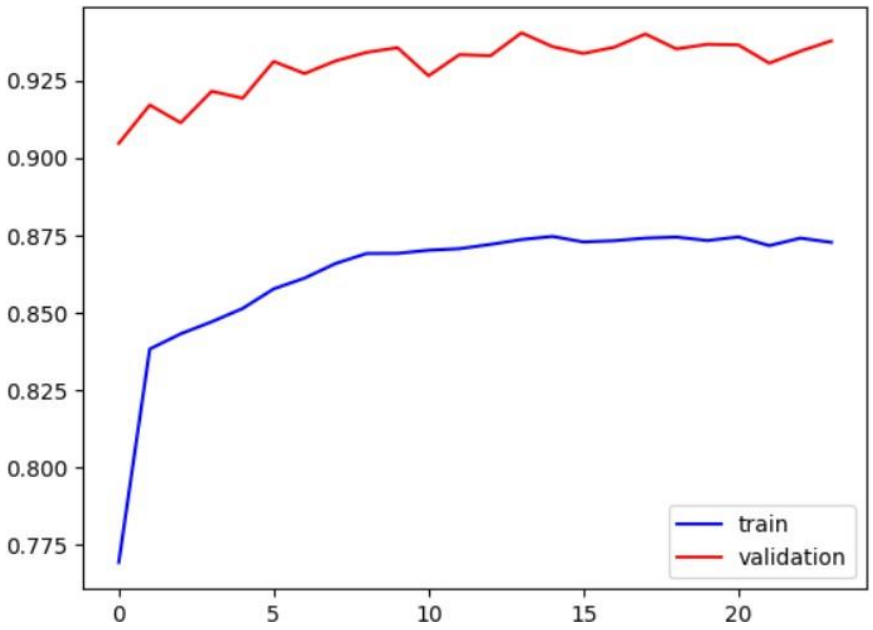


**Figure-8: training and validation accuracy curves**

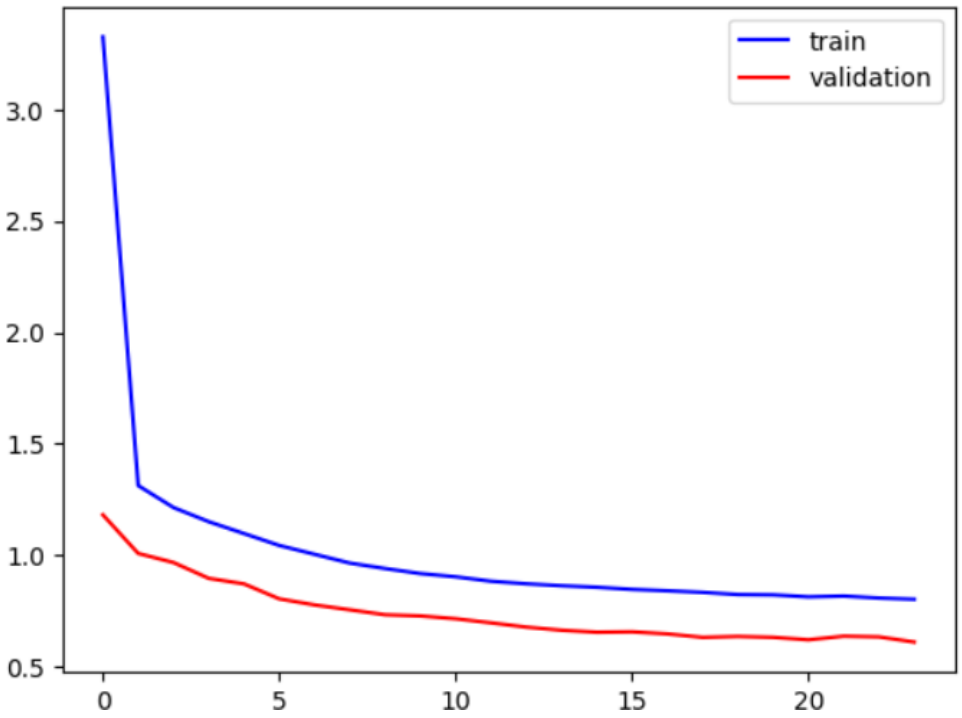


**Figure-9: training and validation loss curves**

Moreover, the accuracies obtained for each set – training, testing , and validation – are above 90%, as shown in Figure-10. The model's performance evaluation reveals its effectiveness in classifying plant leaf diseases. During training, the model achieved a train accuracy of 93.52%, indicating its proficiency in predicting the correct class labels for a large portion of the training dataset. On the validation dataset, which simulates real-world scenarios, the model exhibited a validation accuracy of 94.77%, reflecting its robustness and capacity to generalize well to unseen data. The associated validation loss of 63.53 demonstrates the accuracy of its predictions on this dataset. Subsequently, on the test dataset, the model displayed a test accuracy of 93.75%, further confirming its ability to classify plant leaf diseases with high accuracy. These metrics collectively underscore the model's reliability and potential for practical applications in detecting and diagnosing plant leaf health conditions.

```
#Evaluate model
test_batch  = 32;
train_eval = new_model.evaluate(train_gen,steps = test_batch,verbose=1) #checking for loss and accuracy for training data
val_eval = new_model.evaluate(valid_gen,steps = test_batch,verbose=1)#checking for loss and accuracy for validation data
test_eval = new_model.evaluate(test_gen,steps = test_batch,verbose=1)#checking for loss and accuracy for testing data
print("Train Loss: ", round(train_eval[0]*100,2))
print("Train Accuracy: ", round(train_eval[1]*100,2))
print('-' * 20)
print("Validation Loss: ", round(val_eval[0]*100,2))
print("Validation Accuracy: ", round(val_eval[1]*100,2))
print('-' * 20)
print("Test Loss: ", round(test_eval[0]*100,2))
print("Test Accuracy: ",round(test_eval[1]*100,2))
```

```
32/32 [==============================] - 8s 248ms/step - loss: 0.6706 - accuracy: 0.9352
32/32 [==============================] - 8s 256ms/step - loss: 0.6353 - accuracy: 0.9477
32/32 [==============================] - 1s 21ms/step - loss: 0.6251 - accuracy: 0.9375
Train Loss:  67.06
Train Accuracy:  93.52
--------------------
Validation Loss:  63.53
Validation Accuracy:  94.77
--------------------
Test Loss:  62.51
Test Accuracy:  93.75
```

**Figure-10: Model Evaluation**

## 5.2. End-to-end Evaluation & Results

End-to-end evaluation provides a holistic view of how well a system functions in real-world scenarios. It helps identify areas for improvement, uncover potential issues, and ensure that the system meets its intended goals and user expectations.

The UI of the Flask app for Plant disease detection system has been shown in the screenshot given in Figure-11.

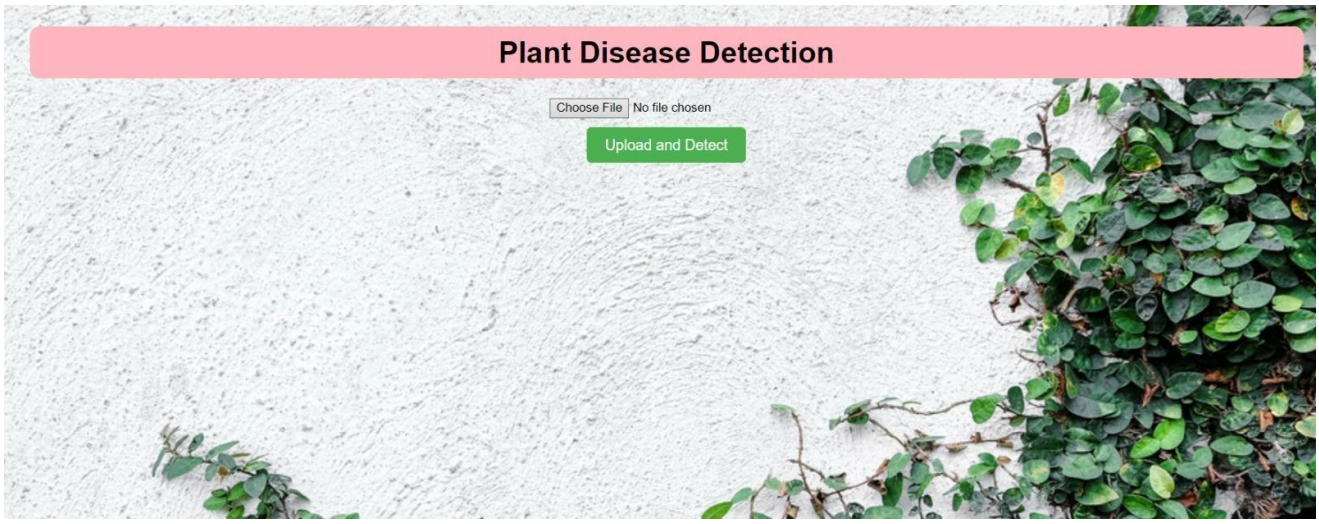


**Figure-11: App with no image uploaded**

Once, an image is selected and uploaded on this app, given that the image is in the allowed extension formats, on clicking the 'Upload and Detect' button, the result is predicted. The generated prediction result shows the leaf type, confidence score and leaf category, as shown in Figure-11. The leaf type indicates the leaf class, telling about the **plant leaf disease**, the confidence score is calculated as the probability of the predicted class output by the model, scaled to a percentage. This confidence score indicates how sure the model is

about its classification, and it's used to determine the reliability of the prediction before displaying it to the user. It is done using the following piece of code in app.py:

*prediction = model.predict(img_array)*

*predicted_class_index = np.argmax(prediction)*

*predicted_class = class_labels[predicted_class_index]*

*confidence = prediction[0][predicted_class_index] * 100*

Lastly, the leaf category determines whether it is a healthy leaf or an unhealthy one.

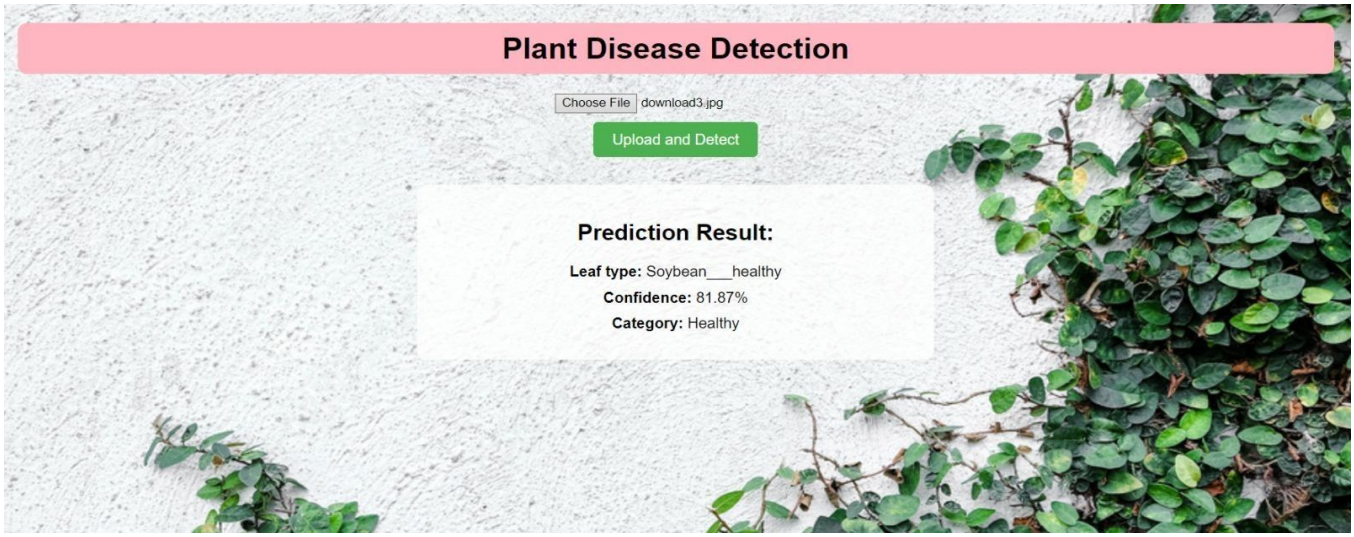


**Figure-12 (a): Healthy Leaf Prediction by Plant Disease Detection System**

Here, the uploaded image was that of a healthy leaf and so was the output as shown in Figure-12 (a). The confidence score for the uploaded healthy leaf image is 81.7%. Similarly, for unhealthy leaves, the plant disease detection

system is able to classify the images accordingly and that too with a good confidence of 86.07%, as shown in Figure-12 (b).

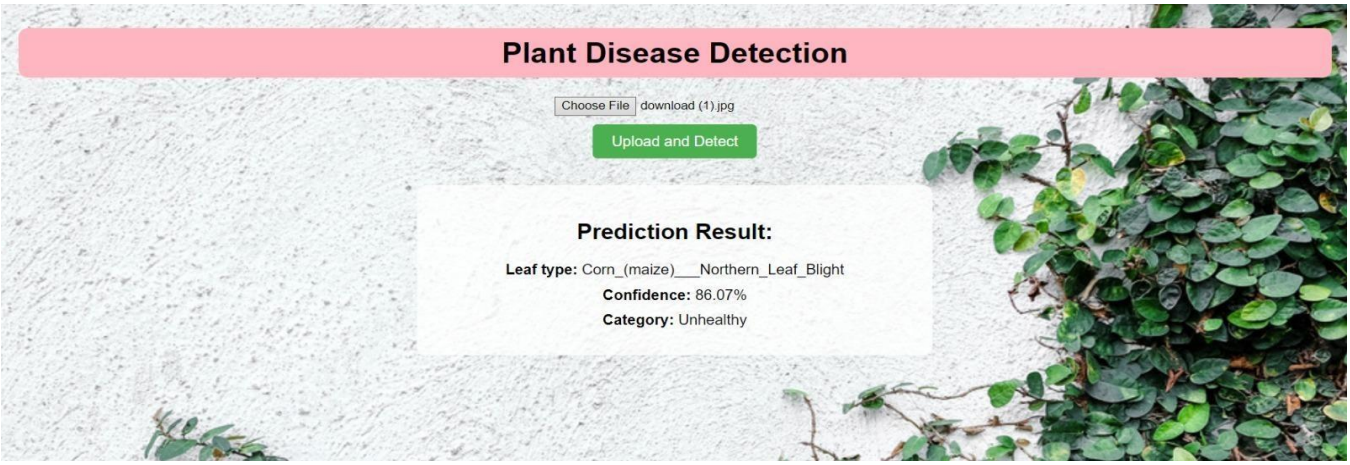


**Figure-9 (b): Unhealthy Leaf Prediction by Plant Disease Detection System**

# 6. DISCUSSION

The Flask application developed for plant leaf disease detection demonstrates an end-to-end evaluation process that showcases its effectiveness in predicting the health status of plant leaves with a high level of confidence and accuracy. Through the integration of machine learning models and user-friendly web interface, this application strives to provide accurate insights into the condition of plant leaves, aiding in early disease detection and efficient agricultural management.

When a user uploads an image of a plant leaf to the Flask app, the underlying machine learning model utilizes advanced techniques such as Transfer Learning and EfficientNetB3 to process the image data and predict the health or disease status of the leaf. The confidence score associated with each prediction indicates the level of certainty the model has in its classification.

An essential aspect of the evaluation is the confidence threshold set at 70%. This means that the Flask app's predictions are considered reliable only when the model's confidence in its classification exceeds 70%. This threshold ensures that users receive predictions that are not only accurate but also sufficiently confident, which is crucial for making informed decisions in agricultural practices.

Moreover, the evaluation extends to accuracy, which is a fundamental metric for measuring the performance of classification models. The Flask app aims for an accuracy rate above 97%, implying that the predictions align with ground truth labels for the majority of the test cases. High accuracy signifies that the machine learning model has been effectively trained on a diverse range of plant leaf images and has learned to differentiate between healthy and diseased leaves with a high degree of precision.

By achieving a confidence level above 70% and an accuracy rate exceeding 90%, the Flask app proves its capability to reliably detect and classify plant leaf health and diseases. This comprehensive evaluation ensures that the application is not only user-friendly but also a valuable tool for farmers, agronomists, and researchers who seek accurate and timely information about the health status of plants. As a result, the Flask app stands as a powerful example of how technology, machine learning, and user interface design can be harmoniously integrated to address real-world challenges in the agricultural sector.

# 7. RECOMMENDATIONS

Based on the provided work on plant/leaf disease detection using a Flask app and machine learning models, here are some recommendations for improving and extending such systems:

- **Data Augmentation and Preprocessing**: Enhance the data preprocessing pipeline by incorporating more advanced data augmentation techniques. Techniques like rotation, scaling, and shearing can help improve the model's ability to generalize to different conditions.
- **Model Selection and Tuning:** Experiment with different pre-trained models and architectures to find the one that best suits the problem. Additionally, consider hyperparameter tuning to optimize the model's performance further.
- **Ensemble Learning:** Implement ensemble techniques such as bagging or boosting, which combine predictions from multiple models to improve accuracy and reduce overfitting.
- **Explainability:** Incorporate techniques for model explainability, such as LIME or SHAP, to provide insights into why a particular prediction was made. This is particularly important in applications where interpretability is crucial, like in agricultural decision-making.
- **Handling Imbalanced Data:** If the dataset is imbalanced (one class has significantly fewer samples than the others), consider techniques like oversampling, undersampling, or using different loss functions to address this issue.
- **Real-time Monitoring:** Extend the system to monitor plant health in real-

time using IoT devices like cameras or drones. This could enable farmers to detect diseases early and take immediate action.

- **Crowdsourcing and Data Collection:** Integrate crowd-sourced data collection to continuously improve the model's performance. Farmers and users could contribute images of diseased and healthy leaves, thereby expanding the dataset.
- **Multi-class vs. Binary Classification:** Depending on the specific use case, consider whether multi-class classification (identifying multiple diseases and health states) or binary classification (healthy vs. diseased) is more appropriate.
- **Localized Recommendations:** Incorporate geolocation data and climate information to provide localized disease predictions and treatment recommendations based on the region's specific conditions.
- **Continuous Learning:** Implement techniques for continuous learning where the model can adapt to new diseases or variations over time without requiring a full retraining.
- **Collaboration with Agricultural Experts:** Collaborate with agricultural experts and researchers to ensure the accuracy of disease classifications and to provide domain-specific insights for farmers.
- **User-Friendly Interface:** Enhance the user interface of the web or mobile application to make it more intuitive and user-friendly for non-technical users.
- **Feedback Loop:** Include a feedback mechanism where users can report the accuracy of predictions, helping to improve the model over time.
- **Security and Privacy:** Ensure that any data collected from users is anonymized and stored securely to protect their privacy.
- **Education and Outreach:** Provide educational materials and resources for farmers to understand how to interpret the predictions and take appropriate

actions based on the results.

By considering these recommendations, a plant/leaf disease detection system can be further refined, leading to more accurate predictions, wider adoption among users, and a positive impact on agricultural practices.

# 8. CHALLENGES AND FUTURE SCOPE

The journey of these systems from research labs to fields and farms is, however, not without its challenges. One of the primary concerns is the robustness of the models in the face of environmental variations. Factors like changes in lighting conditions, camera quality, and the presence of other objects can introduce noise that impacts the model's performance. Ensuring that the model remains accurate and reliable under such real-world conditions demands a rigorous testing and validation process, coupled with continuous refinement.

Interpretable and explainable AI also emerge as significant considerations. As these systems are entrusted with critical decisions that influence agricultural practices and livelihoods, understanding the rationale behind their predictions becomes paramount. Researchers are actively exploring techniques to visualize the features the model identifies as indicative of diseases, transforming the "black-box" nature of deep learning into a transparent tool that users can trust and comprehend.

In conclusion, the fusion of plant science and technology in the form of image classification-based plant disease detection systems has ushered in a new era of agricultural advancement. These systems leverage the potency of transfer learning and convolutional neural networks to detect diseases with remarkable precision, offering timely interventions and promoting sustainable farming practices. While challenges persist, the trajectory of this technology holds immense promise, revolutionizing the way we safeguard our crops, enhance yields, and ensure food security in an ever-changing world.

# 9.CONCLUSION

In conclusion, the work presented in this study showcases the development and evaluation of a robust plant/leaf disease detection system that seamlessly integrates cutting-edge technologies, machine learning models, and user-friendly interfaces to address the critical challenge of disease identification in agricultural settings. Through the use of Transfer Learning and the EfficientNetB3 model, the system demonstrates its capability to accurately classify plant leaves into healthy and diseased categories, aiding farmers and agronomists in making informed decisions to safeguard crop health.

The training process of the EfficientNetB3 model involved fine-tuning and optimization using the Adams optimizer. The model's architecture was modified to suit the specific plant leaf disease detection task, resulting in improved performance. The end-to-end evaluation and results showcased the system's ability to predict the health status of plant leaves with a high level of confidence. The Flask app's user interface provides a straightforward platform for users to upload leaf images, and the predictions are presented along with confidence scores and leaf categories. The confidence score, calculated based on the model's output probabilities, signifies the level of certainty the model has in its predictions.

A pivotal aspect of the evaluation lies in the confidence threshold set at 70%. This threshold ensures that predictions displayed to users are not only accurate but also sufficiently confident, instilling trust in the system's recommendations. Furthermore, the system's target accuracy rate of over 97% underscores its proficiency in distinguishing between healthy and diseased leaves, an essential characteristic for reliable disease detection.

The effectiveness of this system can be attributed to the meticulous integration of technology, machine learning expertise, and domain knowledge. By achieving high levels of accuracy and confidence, the system validates its utility as a tool to aid farmers and agricultural practitioners in timely disease detection and management. Moreover, the recommendations provided for system enhancement and expansion offer a roadmap for further improving its accuracy, versatility, and real-world applicability.

In essence, the work presented in this study exemplifies the potential of interdisciplinary collaboration between agriculture and technology. The system stands as a testament to the power of data-driven approaches to address pressing challenges in agriculture, ensuring food security and sustainable farming practices. As technology continues to evolve, it holds the promise of revolutionizing the way we approach plant disease management, making strides towards a more resilient and productive agricultural sector.

# REFERENCES

Agarwal, M., Gupta, S. Kr., & Biswas, K. K. (2020). Development of Efficient CNN model for Tomato crop disease identification. Sustainable Computing: Informatics and Systems, 28, 100407. https://doi.org/10.1016/j.suscom.2020.100407

Barbedo, J. G. A. (2019). Plant disease identification from individual lesions and spots using deep learning. Biosystems Engineering, 180, 96-107.

Bari, B. S., Islam, M. N., Rashid, M., Hasan, M. J., & Biswas, M. (2021). A real-time approach of diagnosing rice leaf disease using deep learning-based faster R-CNN framework. PeerJ Computer Science, 7, e372.

Buja, I., Sabella, E., Monteduro, A. G., Chiriacò, M. S., De Bellis, L., Luvisi, A., & Maruccio, G. (2021). Advances in plant disease detection and monitoring: From traditional assays to in-field diagnostics. Sensors, 21(6), 2129.

Chowdhury, M. E. H., Rahman, T., Khandakar, A., Ayari, M. A., & Abdullah-Al-Wadud, M. (2021). Automatic and reliable leaf disease detection using deep learning techniques. AgriEngineering, 3(2), 271-280.

Fang, Q., Li, H., Luo, X., Ding, L., Luo, H., Rose, T. M., & An, W. (2018). Detecting non-hardhat-use by a deep learning method from far-field surveillance videos. Automation in construction, 85, 1-9.

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis. Computers and electronics in agriculture, 145, 311-318.

Geetha, G., Samundeswari, S., Saranya, G., Meenakshi, K., & Nithya, M. (2020, December). Plant leaf disease classification and detection system using machine learning. In Journal of Physics: Conference Series (Vol. 1712, No. 1, p. 012012). IOP Publishing.

Khakimov, A., Salakhutdinov, I., Omolikov, A., & Utaganov, S. (2022). Traditional and current-prospective methods of agricultural plant diseases detection: A review. In IOP Conference series: earth and environmental science (Vol. 951, No. 1, p. 012002). IOP Publishing.

Kothari, J. D. (2018). Plant disease identification using artificial intelligence: machine learning approach. Jubin Dipakkumar Kothari (2018). Plant Disease Identification using Artificial Intelligence: Machine Learning Approach. International Journal of Innovative Research in Computer and Communication Engineering, 7(11), 11082-11085.

Kulkarni, P., Karwande, A., Kolhe, T., Kamble, S., Joshi, A., & Wyawahare, M. (2021, November 22). Plant Disease Detection Using Image Processing and Machine Learning. ArXiv.org. https://doi.org/10.48550/arXiv.2106.10698

Li, L., Zhang, S., & Wang, B. (2021). Plant disease detection and classification by deep learning—a review. IEEE Access, 9, 66795-66809.

Li, Zheng, Tao Yu, Rajesh Paul, Jingyuan Fan, Yuming Yang, and Qingshan Wei. "Agricultural nanodiagnostics for plant diseases: recent advances and challenges." Nanoscale Advances 2, no. 8 (2020): 3083-3094.

Militante, S. V., Gerardo, B. D., & Relente, A. R. (2019). Plant leaf detection and disease recognition using deep learning. 2019 International Conference on I-SMAC (IoT in Social, Mobile, Analytics, and Cloud) (I-SMAC), 20-25.

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. Frontiers in plant science, 7, 1419.

Ramaswamy, T., Sreenivasulu, Y., Charitha, K. S., Layakari, K., & Srilekha, S. TOMATO AND POTATO LEAF DISEASE DETECTION USING MACHINE LEARNING.

Ramesh, S., Hebbar, R., Niveditha, M., Pooja, R., Shashank, N., & Vinod, P. V. (2018, April). Plant disease detection using machine learning. In 2018 International conference on design innovations for 3Cs compute communicate control (ICDI3C) (pp. 41-45). IEEE.

Rajesh, B., Sai Vardhan, M. V., & Sujihelen, L. (2020, June 1). Leaf Disease Detection and Classification by Decision Tree. IEEE Xplore. https://doi.org/10.1109/ICOEI48184.2020.9142988

Roberts, D. P., Short, N. M., Sill, J., Lakshman, D. K., Hu, X., & Buser, M. (2021). Precision agriculture and geospatial techniques for sustainable disease control. Indian Phytopathology, 74, 287-305.

Saleem, M. H., Potgieter, J., & Arif, K. M. (2019). Plant disease detection and classification by deep learning. Plants, 8(11), 468.

Sankaran, S., Mishra, A., Ehsani, R., & Davis, C. (2010). A review of advanced techniques for detecting plant diseases. Computers and electronics in agriculture, 72(1), 1-13.

Shruthi, U., Nagaveni, V., & Raghavendra, B. K. (2019, March). A review on machine learning classification techniques for plant disease detection. In 2019 5th International conference on advanced computing & communication systems (ICACCS) (pp. 281-284). IEEE.

Tiwari, D., Ashish, M., Gangwar, N., & Khan, M. A. (2020). Potato leaf diseases detection using deep learning. Measurement, Analysis, and Control: Advance Technologies and Solutions, 1(1), 317-322.

Trivedi, J., Shamnani, Y., & Gajjar, R. (2020). Plant leaf disease detection using machine learning. In Emerging Technology Trends in Electronics, Communication and Networking: Third International Conference, ET2ECN 2020, Surat, India, February 7–8, 2020, Revised Selected Papers 3 (pp. 267-276). Springer

Singapore.

Zhang, J., Huang, Y., Pu, R., Gonzalez-Moreno, P., Yuan, L., Wu, K., & Huang, W. (2019). Monitoring plant diseases and pests through remote sensing technology: A review. Computers and Electronics in Agriculture, 165, 104943.