DETECTION OF FUNGAL DISEASES USING LEAF IMAGES THROUGH CNN

A PROJECT REPORT

Submitted by

RAHUL J **(953120104039)**

THANGAPPAN N **(953120104054)**

SRIMAN BALA GANESH M **(953120104304)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

THAMIRABHARANI ENGINEERING COLLEGE, TIRUNELVELI

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2024

DETECTION OF FUNGAL DISEASES USING LEAF IMAGES THROUGH CNN

A PROJECT REPORT

Submitted by

RAHUL J **(953120104039)**

THANGAPPAN N **(953120104054)**

SRIMAN BALA GANESH M **(953120104304)**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

THAMIRABHARANI ENGINEERING COLLEGE, TIRUNELVELI

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2024

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **"DETECTION OF FUNGAL DISEASES USING LEAF IMAGES THROUGH CNN"** is the bonafide work of "**RAHUL J (953102014039), THANGAPPAN N (953120104054), SRIMAN BALA GANESH M (953120104304)"** who carried out the project work under my supervision.

SIGNATURE

Dr. J. ARMSTRONG JOSEPH, Ph. D.,

HEAD OF THE DEPARTMENT

Department of Computer Science and Engineering

Thamirabharani Engineering College, Tirunelveli - 627 358

SIGNATURE

Dr. J. ARMSTRONG JOSEPH, Ph. D.,

SUPERVISOR

Head of the Department

Department of Computer Science and Engineering

Thamirabharani Engineering College, Tirunelveli - 627 358

Submitted to Anna University Viva – Voce held on _____

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We wish our heartful thanks to **Mr. M. R. Paulraj**, **Chairman** and **Mr. Senthil Kumar Palraj**, **Secretary** for the blessing and constant support over our project period.

We wish to express our thanks to **Dr. G. Ranganathan, M. E., Ph. D., Principal**, **Mr. A. Balajikrishnabharathi, M. E., (Ph. D.), Director-Academics,** and **Mr. T. Kumaran, M. E., (Ph. D.), Director-Administration,** Thamirabharani Engineering College.

Our special thanks to our Head of the Department and Project Guide **Dr. J. Armstrong Joseph M. E., Ph. D.**, for his valuable guidance and support to complete our project successfully.

We wish to express our indebtedness to our Project Coordinator **Dr. P. Revathy M. E., Ph. D.,** Associate Professor, Department of Computer Science and Engineering for her valuable guidance, advice and encouragement in successfully completing our work.

We also express our sincere thanks to entire faculty and lab assistants in Computer Science and Engineering Department for their co-operation and support.

We are very grateful to our well-wishers as without their encouragement we would not able to make this project work as success.

ABSTRACT

Plant diseases are a significant threat to agricultural productivity, causing reduced crop yields and economic losses. Detecting these diseases early is crucial for effective management and mitigation. Among the various types of plant diseases, fungal infections like rust and powdery mildew are particularly problematic due to their widespread occurrence and destructive nature. To address this challenge, researchers are leveraging advanced technologies such as Convolutional Neural Networks (CNNs) to develop efficient disease identification methods. In this study, a Conv-9 CNN architecture is employed, specifically tailored for analyzing images, to accurately identify fungal diseases based on leaf images. The Conv-9 CNN operates by processing the visual features of leaf images through multiple layers of convolutional and pooling operations. Through the training process with large datasets containing diverse examples of healthy and diseased plants, the CNN learns to recognize patterns and distinguish between different types of fungal infections. This learning process enables the CNN to accurately classify images as either healthy or infected with specific fungal diseases such as rust or powdery mildew. The implementation of this technology empowers farmers with a rapid and reliable tool for disease detection in their crops. By capturing images of plant leaves and feeding them into the trained Conv-9 CNN model, farmers can quickly receive feedback on the presence of fungal infections.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

Adam	Adaptive Moment Estimation
API	Application Program Interface
CNNs	Convolutional Neural Networks
DCNNs	Deep Convolutional Neural Networks
DFD	Data Flow Diagram
GNNs	Graph Neural Networks
IoT	Internet of Things
ML	Machine Learning
OOP	Object Oriented Programming
OpenCV	Opensource Computer Vision
RAM	Random Access Memory
ResNet	Residual Networks
RNNs	Recurrent Neural Networks
UML	Unified Modeling Language
VGG	Visual Geometry Group

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

Detecting fungal diseases in plant leaves through Convolutional Neural Networks (CNNs) requires a systematic approach, starting with meticulous data collection. A diverse dataset is indispensable, encompassing images of plant leaves from various species, growth stages, and environmentalconditions. This dataset must include both healthy leaves and those infected with different fungal diseases, capturing a range of infection severities. Ideally, each image should be annotated to indicate whether it depicts ahealthy leaf or one afflicted by a specific fungal disease. Data preprocessingis vital to prepare the raw image data for training. This involves resizing images to a uniform size, normalizing pixel values to a standard scale, and applying augmentation techniques like rotation, flipping, and scaling to enhance dataset diversity. Noise and irrelevant background elements are removed to focus the model's attention on the leaf itself. The design of an appropriate CNN architecture is critical for effective feature extraction and classification. Common architectures such as VGG, ResNet, or Inception networks are typically employed. These architectures consist of convolutional layers for feature extraction followed by pooling layers, and fully connected layers for classification. Transfer learning, utilizing pre-trained CNN models fine-tuned for fungal disease detection, can also be beneficial. Training the CNN model involves adjusting its internal parameters through backpropagation and optimization algorithms like stochastic gradient descent (SGD) or Adam. The model is presented with batches of preprocessed images iteratively, with the aim of minimizing prediction errors (loss). The training process may require multiple epochs to converge, depending on the dataset's complexity and size.

Validation is essential to evaluate the model's performance on a separate dataset not used during training. This helps assess its generalization abilityand identify potential issues like overfitting. Hyperparameters such as learning rate, batch size, and network architecture may be adjusted based on validation performance. Once trained and validated, the model is tested on an unseen dataset to assess its accuracy and effectiveness in detecting fungal diseases in plant leaves. This provides an unbiased measure of the model's real-world applicability. Finally, upon demonstrating satisfactory performance, the model can be deployed for practical applications such as crop monitoring in agricultural fields or aiding farmers in early disease detection. Deployment may involve integration into user-friendly interfaces orembedding within automated systems for continuous monitoring.

1.2 INTRODUCTION TO FUNGAL DISEASES

Fungal diseases represent a significant threat to plant health and agricultural productivity, posing challenges to farmers and ecosystems worldwide. These diseases are caused by various fungal pathogens that can infect a wide range of plant species, including crops, ornamentals, and trees. Fungal pathogens can attack plant leaves, stems, roots, and fruits, leading to symptoms such as leaf spots, wilting, necrosis, and fruit rot. The impact of fungal diseases extends beyond crop yield losses, affecting food security, economic livelihoods, and environmental sustainability. Fungal diseases spread through multiple mechanisms, including airborne spores, contaminated soil, infected seeds, and vectors such as insects and water. Favorable environmental conditions, such as high humidity, warm temperatures, and poor air circulation, often contribute to disease development and spread. Additionally, factors such as monoculture farming, globalization of trade, and climate change can exacerbate the prevalence and severity of fungal diseases

by altering ecological balances and creating new opportunities for pathogen dissemination. Management of fungal diseases typically involves a combination of cultural practices, chemical treatments, and biological control methods. Cultural practices such as crop rotation, sanitation, and planting disease-resistant varieties help reduce pathogen pressure and minimize disease outbreaks. Chemical treatments, including fungicides, are commonly used to control fungal infections, although their efficacy can be limited by factors such as resistance development and environmental concerns. Biological control strategies, utilizing beneficial microbes or antagonistic organisms, offer sustainable alternatives to chemical control methods. Early detection and accurate diagnosis of fungal diseases are essential for effective disease management and mitigation of economic losses. Advances in technology, such as remote sensing, molecular diagnostics, and artificial intelligence, are revolutionizing the detection and monitoring of fungal diseases in plant populations. These tools enable rapid and non-destructive assessment of disease incidence, allowing for timely interventions and informed decision-making by farmers and plant health professionals.

1.3 INTRODUCTION TO MACHINE LEARNING

Machine learning holds significant promise for revolutionizing the detection, diagnosis, and management of fungal diseases in plants. Leveraging computational algorithms and statistical techniques, machine learning models can analyze complex datasets, identify patterns, and make predictions with remarkable accuracy. In the context of fungal diseases, machine learning techniques offer innovative solutions for early detection, rapid diagnosis, and targeted intervention, ultimately enhancing crop resilience and agricultural sustainability. One of the primary applications of machine learning in fungal disease management is in the development of predictive models for disease

risk assessment. By analyzing environmental factors, crop characteristics, and historical disease incidence data, machine learning algorithms can forecast the likelihood of fungal disease outbreaks and provide early warnings to farmers. These predictive models enable proactive decision-making, allowing farmers to implement preventive measures and minimize crop losses. Machine learning also plays a crucial role in automated image-based detection and diagnosis of fungal diseases in plant tissues. Convolutional Neural Networks (CNNs) and other deep learning architectures excel in analyzing digital images of plant leaves, identifying characteristic symptoms of fungal infections, and distinguishing between healthy and diseased plants. This automated image analysis accelerates disease identification, facilitates large- scale screening efforts, and enables precision agriculture practices tailored to specific disease pressures. Moreover, machine learning techniques contribute to personalized disease management strategies by integrating diverse data sources and optimizing treatment recommendations. By considering factors such as soil health, weather conditions, and crop genetics, machine learning algorithms can tailor fungicide applications and cultural practices toindividual farm contexts, maximizing efficacy while minimizing environmental impacts and input costs. But also separate non-linearly scattereddata by converting the classification plane to higher dimensions. Machine learning techniques provided a wealth of benefits for classifying traffic signs, but they were unable to handle the features, such as various sizes of images and aspect ratios, which must be completed manually. Thus, the feature generated process was always time-consuming and error-prone. Machine learning has the potential to revolutionize not only fungal disease management but also various other fields by leveraging its ability to process vast amounts of data and extract meaningful insights. As technology advances and algorithms improve, the applications of machine learning in agriculture.

1.4 INTRODUCTION TO CNN

Convolutional Neural Networks (CNNs) have emerged as a transformative tool in the domain of fungal disease detection, offering unprecedented capabilities for automated analysis of plant leaf images to identify signs of infection. CNNs are a class of deep learning models specifically designed for image recognition tasks, inspired by the biological processes of visual perception in animals. Within the realm of fungal diseases, CNNs play a pivotal role in scrutinizing digital images of plant leaves, discerning subtle patterns and features indicative of fungal infection. Their architecture is structured to automatically learn hierarchical representations of features from raw pixel data, enabling them to effectively capture the nuanced characteristics of fungal lesions, discolorations, or other symptomatic manifestations across diverse plant species and environmental conditions. By leveraging convolutional layers for feature extraction and pooling layers for spatial down sampling, CNNs excel at identifying relevant patterns at various scales, orientations, and contexts within the image. This capability empowers CNNs to distinguish between healthy and infected leaves with remarkable accuracy, facilitating early detection and proactive management of fungal diseases in agricultural settings. As CNNs continue to evolve and adapt, they hold the promise of revolutionizing fungal disease diagnosis, contributing to improved crop health, and sustainable agriculture practices. CNN, or Convolutional Neural Networks, are a class of neural networks designed for visual processing tasks like image recognition. They use convolutional layers to efficiently capture patterns and features from images.

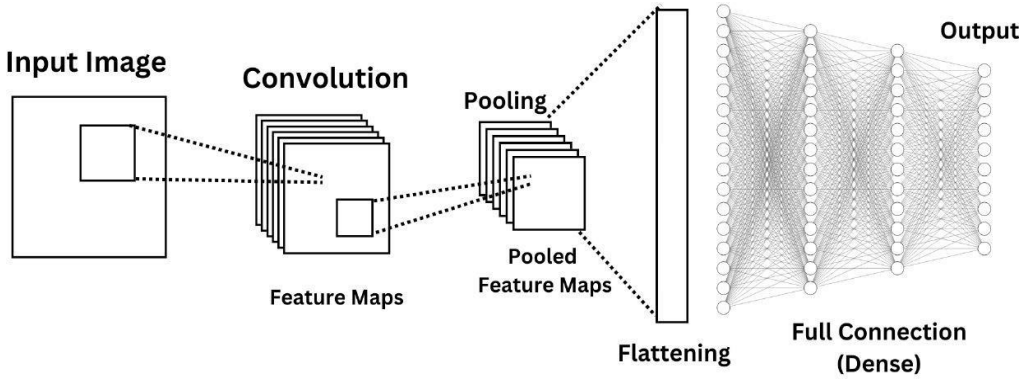

Figure1.1 Convolutional Neural Network

1.4.1 Deep Neural Networks

Detecting fungal diseases in plant leaves through Convolutional Neural Networks (CNNs) using leaf images involves leveraging deep learning techniques to automate the identification and classification of fungal infections. Deep Convolutional Neural Networks (DCNNs) are a specific typeof CNN architecture optimized for image analysis tasks, making them well- suited for this application. The process typically begins with data collection, where a diverse dataset of plant leaf images is compiled, covering various species, growth stages, and fungal infections. Each image is annotated to indicate whether it depicts a healthy leaf or one affected by a specific fungal disease. Next, the dataset undergoes preprocessing to enhance its quality and diversity. This may involve resizing images, normalizing pixel values, and applying augmentation techniques to increase the dataset's robustness. The DCNN architecture is then designed, typically comprising multiple convolutional layers for feature extraction, followed by pooling layers to

reduce spatial dimensions, and fully connected layers for classification. Transfer learning may also be employed, where a pre-trained DCNN model (e.g., VGG, ResNet) is fine-tuned on the specific task of fungal disease detection in plant leaves. The DCNN model is trained using the preprocessed dataset, where it learns to differentiate between healthy and infected leaves by adjusting its internal parameters through backpropagation and optimization algorithms. After training, the model's performance is evaluated on a separate validation dataset to assess its generalization ability and identify potential issues like overfitting. Once validated, the model is tested on unseen leaf images to evaluate its accuracy and effectiveness in detecting fungal diseases. Finally, the trained DCNN model can be deployed for real-world applications, such as monitoring crops in agricultural fields or assisting farmers in early disease detection, thereby contributing to improved crop health and yields.

1.4.2 Neural Network

The detection of fungal diseases in leaf images through Convolutional Neural Networks (CNNs) using leaf involves constructing a neural network architecture optimized for image analysis tasks. This architecture is specifically tailored to process digital representations of plant leaves, aiming to automatically identify signs of fungal infection. Initially, a diverse dataset of leaf images is gathered, encompassing both healthy leaves and leaves affected by various fungal diseases. Each image is meticulously annotated to denote its disease status, providing crucial training data for the neural network. The neural network is structured to consist of multiple layers, including convolutional layers responsible for extracting features from the input leaf images, pooling layers for spatial down sampling, and fully connected layers for classification. Through the process of training, the network learns to differentiate between healthy and infected leaves by iteratively adjusting its internal parameters based on the provided training

data. Training involves presenting batches of preprocessed leaf images to the neural network, computing prediction errors, and updating the network's parameters to minimize these errors. The network's performance is continuously evaluated during training to ensure its effectiveness in accurately detecting fungal diseases. Once trained, the neural network can efficiently analyze unseen leaf images and classify them as healthy or infectedwith fungal diseases based on learned features. This automated detection capability facilitates early disease diagnosis and intervention, contributing to improved plant health and agricultural productivity.

1.5 DEFINITION OF IMAGE

A picture is spoken to as a two-dimensional capacity f (x, y) where x and y are spatial co-ordinates and the adequacy of "T" at any match of directions (x, y) is known as the power of the picture by then.

1.5.1 Processing On Image

Processing on image can be of three types They are low-level, mid-level, high level.

Low-level Processing:

- Preprocessing To Remove Noise
- Contrast Enhancement
- Image Sharpening

Medium Level Processing:

- Segmentation
- Edge Detection
- Object Extraction

High Level Processing

- Image Analysis
- Scene Interpretation

1.5.2 Image Processing

Since the digital image is invisible, it must be prepared for viewing on one or more output device (laser printer, monitor at). The digital image can be optimized for the application by enhancing the appearance of the structures within it. There are three of image processing used. They are

- Image to Image transformation
- Image to Information transformations
- Information to Image transformations

1.5.3 Pixel

Pixel is the smallest element of an image. Each pixel corresponds to any one value. In an 8-bit gray scale image, the value of the pixel between 0 and 255.Each pixel store a value proportional to the light intensity at that particular location. It is indicated in either Pixels per inch or Dots.

1.5.4 Resolution

The resolution can be defined in many ways. Such as pixel resolution, spatial resolution, temporal resolution, spectral resolution. In pixel resolution, the term resolution refers to the total number of count of pixels in a digital image. For example, if an image has M rows and N columns, then its resolution can be defined as MX N. Higher is the pixel resolution, the higheris the quality of the image.

Resolution of an image is of generally two types.

- Low Resolution Image

- High Resolution

Since high resolution is not a cost-effective process It is not always possible to achieve high resolution images with low cost. Hence it is desirable Imaging. In Super Resolution imaging, with the help of certain methods and algorithms we can be able to produce high resolution images from the low-resolution image from the low-resolution images.

1.5.5 Gray Scale Image

A gray scale picture is a capacity I (xylem) of the two spatial directions of the picture plane. I (x, y) am the force of the picture force of picture at the point (x, y) on the picture plane. I (xylem) take non- negative expect the picture is limited by a rectangle

1.5.6 Color Image

It can be spoken to by three capacities, R (xylem) for red, G (xylem) for green and B (xylem) for blue. A picture might be persistent as for the x and y facilitates and furthermore in adequacy. Changing over such a picture to advanced shape requires that the directions and the adequacy to be digitized. Digitizing the facilitates esteems is called inspecting. Digitizing the adequacy esteems is called quantization.

1.6 RELATED TECHNOLOGY

1.6.1 DCNN

DCNN, or Deep Convolutional Neural Networks, are a class of neural networks designed for visual processing tasks like image recognition. They use convolutional layers to efficiently capture patterns and features from images.

1.7 APPLICATION OF FUNGAL LEAF IMAGES

1.7.1 Crop Management:

In agriculture, fungal disease leaf images are utilized for early

detection and monitoring of plant diseases. By analyzing these images, farmers and agricultural experts can identify signs of fungal infections in crops, allowing for timely intervention measures such as targeted pesticide application or crop rotation.

1.7.2 Research & Disease Prediction:

Fungal disease leaf images serve as valuable resources for research purposes, aiding in the understanding of fungal pathogens' behavior, host-pathogen interactions, and disease progression.

1.7.3 Plant Breeding:

Plant breeders utilize fungal disease leaf images to evaluate the resistance or susceptibility of different plant varieties to fungal infections.

1.7.4 Precision Agriculture:

Fungal disease leaf images are integrated into precision agriculture systems and decision support tools to provide farmers with real-time insights and recommendations for disease management.

1.8 SYSTEM REQUIREMENTS

A system requirement is a specification that describes the necessary conditions, capabilities, or characteristics that a software application, hardware device, or system component must possess in order to function correctly.

1.8.1 Hardware Requirements

RAM	**:**	Minimum 4GB
Processor	**:**	Intel core i3 or later
Hard Disk	**:**	Minimum 40GB

1.8.2 Software Requirements

Operating system	**:**	Windows 10 or later
Language	**:**	Python3.10
Platform	**:**	Jupyter Notebook

1.9 IMPLEMENTATION OVERVIEW

The implementation overview serves as a roadmap for executing the project or plan efficiently and effectively. This overview typically outlines the key steps, processes, and resources required. This overview may includedetails such as the technologies or tools that will be used for the implementation.

1.9.1 Introduction to Python

Python is a versatile and high-level programming language known for its simplicity and readability. Developed in the late 1980s by Guido van Rossum, Python has gained immense popularity due to its ease of learning, extensive libraries, and strong community support. It is widely used in variousdomains such as web development, data science, artificial intelligence, scientific computing, and more. Python's clean syntax and dynamic typing make it ideal for rapid prototyping and development, while its robust standard library provides tools for a wide range of tasks. With its emphasis on simplicity and productivity, Python continues to be a go-to choose for both beginners and experienced developers.

1.9.2 OpenCV

OpenCV (Open-Source Computer Vision Library) is a powerful open-source computer vision and machine learning software library. It provides a wide range of functionalities for image and video processing, including real-

time face detection, object recognition, and image segmentation. With support for multiple programming languages, platforms, and hardware accelerators, OpenCV is widely used in various domains such as robotics, augmented reality, and autonomous vehicles.

1.9.3 Features

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by meta programming and meta objects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The language's core philosophy is summarized in the document. The Zen of Python (PEP 20) which includes aphorisms such as:

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach. Python strivesfor a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one and preferably only one obvious way to do it" design philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C andmakes direct C-level API calls into the Python interpreter.

A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic.

CHAPTER 2
LITERATURE SURVEY

1. Title : Detection of Apple Plant Diseases Using Leaf Images Through Convolutional Neural Network

Author : Vibhor Kumar Vishnoi, Krishan Kumar, Brajesh Kumar, Shashank Mohan, Arfat Ahmad Khan

Year : 2023

In this study, we propose a novel approach for the detection of apple plant diseases using leaf images through Convolutional Neural Networks (CNNs). Our method leverages deep learning techniques to automatically analyze digital images of apple leaves and identify signs of common diseases such as apple scab, powdery mildew, and cedar apple rust. Through the utilization ofa CNN architecture optimized for image classification tasks, we achieve accurate and efficient disease detection, enabling early intervention and precise management strategies. The CNN model is trained on a large datasetof annotated apple leaf images, encompassing various disease states and environmental conditions. Experimental results demonstrate the effectiveness of our approach in accurately identifying apple plant diseases, showcasing its potential for practical applications in agricultural settings. This research contributes to the advancement of automated disease diagnosis in fruit crops, paving the way for improved crop health and yield.

2. Title **:** Plant Disease Detection and Classification by Deep Learning

Author : Lili Li, Shujuan Zhang, Bin Wang.

Year : 2023

This study presents a comprehensive approach for the detection and classification of plant diseases using deep learning techniques. Leveraging Convolutional Neural Networks (CNNs), we develop a robust model capable of analyzing leaf images and accurately identifying various diseases affecting plants. Through extensive experimentation and validation on diverse datasets encompassing multiple plant species and disease types, our proposed method demonstrates high accuracy and reliability in disease detection and classification tasks. The utilization of deep learning enables automated and efficient disease diagnosis, facilitating timely interventions and effective disease management strategies in agricultural settings. This research contributes to the advancement of precision agriculture and sustainable crop production practices, offering a valuable tool for farmers, researchers, and agricultural practitioners worldwide.

3. Title : Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks

Author : Peng Jiang, Yuehan Chen, Bin Liu, Dongjian He, Chunquan Liang.

Year : 2023

This research introduces a real-time approach for detecting apple leaf diseases using an enhanced Convolutional Neural Network (CNN) framework. By integrating deep learning techniques, we develop a robust model capable of accurately identifying various diseases affecting apple trees directly from leaf images. Through innovative enhancements to the CNN architecture, our method achieves high-speed processing while maintaining superior accuracy in disease detection. Extensive experimentation and validation on real-world datasets demonstrate the effectiveness of our approach in enabling rapid and reliable diagnosis of apple leaf diseases. This study represents a significant advancement in precision agriculture, providing farmers with a practical tool for early disease detection and proactive management, ultimately contributing to improved crop health and productivity.

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

3.1 EXISTING SYSTEM

The existing technique employs a Conv-3 DCNN architecture, featuring three convolutional layers, two fully connected layers, and a dropout layer. It utilizes the Adam optimization algorithm for efficient parameter optimization. This method swiftly identifies apple diseases like scab, black rot, and cedar rust. However, its suitability is confined to apple plants, necessitating tailored models for other plant species.

Table 3.1 The Previous Works

Sl. No.	TITLE	METHOD
1	Detection of Apple Plant Diseases Using Leaf Images Through Convolutional Neural Network	Conv-3 DCNN architecture, Adam optimization algorithm. .
2	Plant Disease Detection and Classification by Deep Learning	Deep Learning
3	Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks	Improved Neural Network with Rainbow Concatenation and Single Shot Detector (INAR-SSD)

3.2 PROBLEM DEFINITION

Pathogenic fungi cause plant diseases like rust and powdery mildew. Effective detection and classification of these diseases from leaf images are crucial for agricultural management. Leveraging CNN for automated identification faces challenges such as diverse environmental conditions and leaf variations. This project aims to develop a robust CNN-based system for accurate fungal disease detection and classification in plants.

3.3 PROPOSED SYSTEM

This system aims to detect healthy plants and those with rust and powdery mildew. Focusing on normal green leaves, it utilizes image analysis and machine learning to develop a robust model. Using a Conv-9 DCNN architecture and the Adam optimization algorithm, it accurately identifies plant health based on leaf color and texture features.

3.4 UML DIAGRAM

Unified Modeling Language (UML) is a general-purpose modeling language. The main aim of UML is to define a standard way to visualize the way a system has been designed. UML is not a programming language; it is rather a visual language. We use UML diagrams to portray the behavior and structure of a system. UML helps software engineers, businessmen and system architects with modeling, design and analysis.

There are various types of UML diagrams, some are represented below,

i) Usecase Diagram

ii) Activity Diagram

iii) Dataflow Diagram

iv) Class Diagram

3.4.1 Usecase Diagram

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The actors, usually individuals involved with the system defined according to their roles. Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actor).

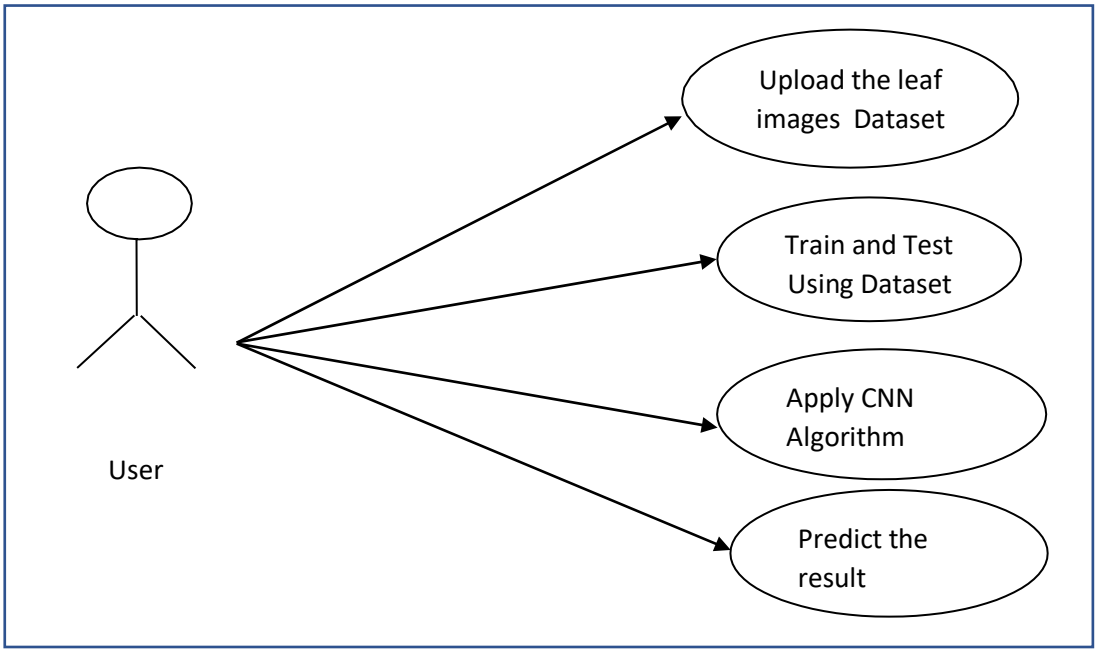

Figure 3.1 Usecase Diagram

3.4.2 Activity Diagram

Activity diagrams are graphical representations of work flows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

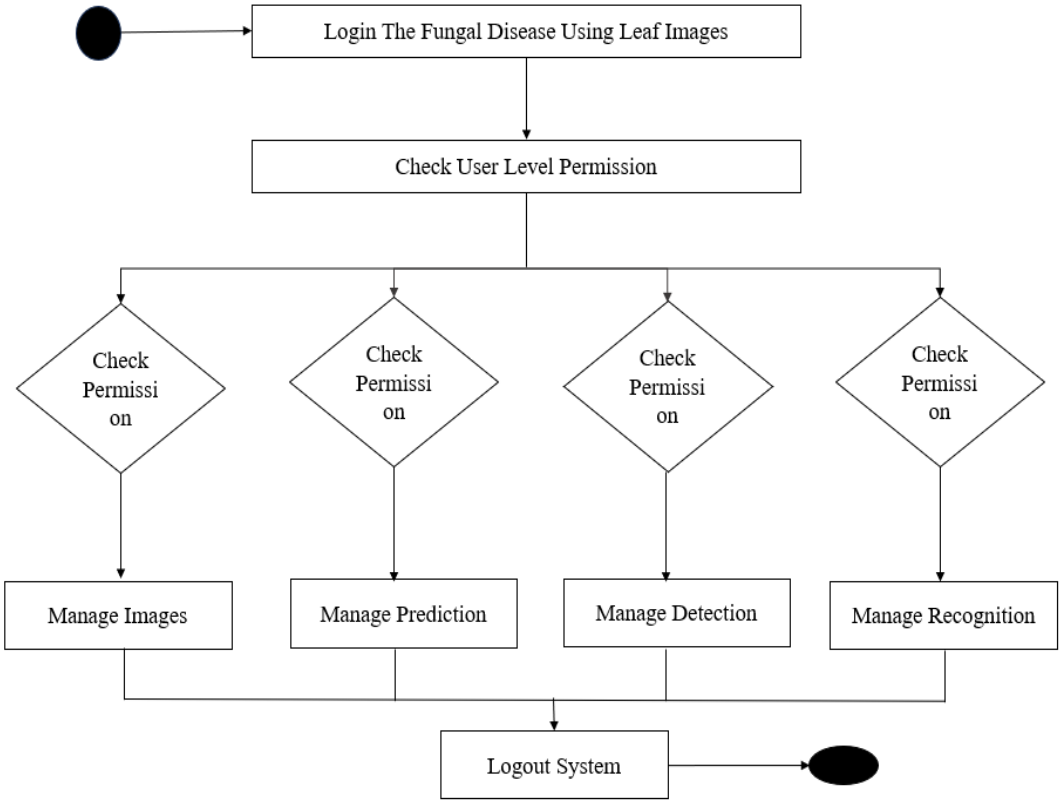

Figure 3.2 Activity Diagram

3.4.3 Dataflow Diagram

DFD is the abbreviation for Data Flow Diagram. The flow of data of a system or a process is represented by DFD. It also gives insight into the inputs and outputs of each entity and the process itself. DFD does not have control flow and no loops or decision rules are present. Specific operations depending on the type of data can be explained by a flowchart. It is a graphicaltool, useful for communicating with users, managers and other personnel. It is useful for analyzing existing as well as proposed system.

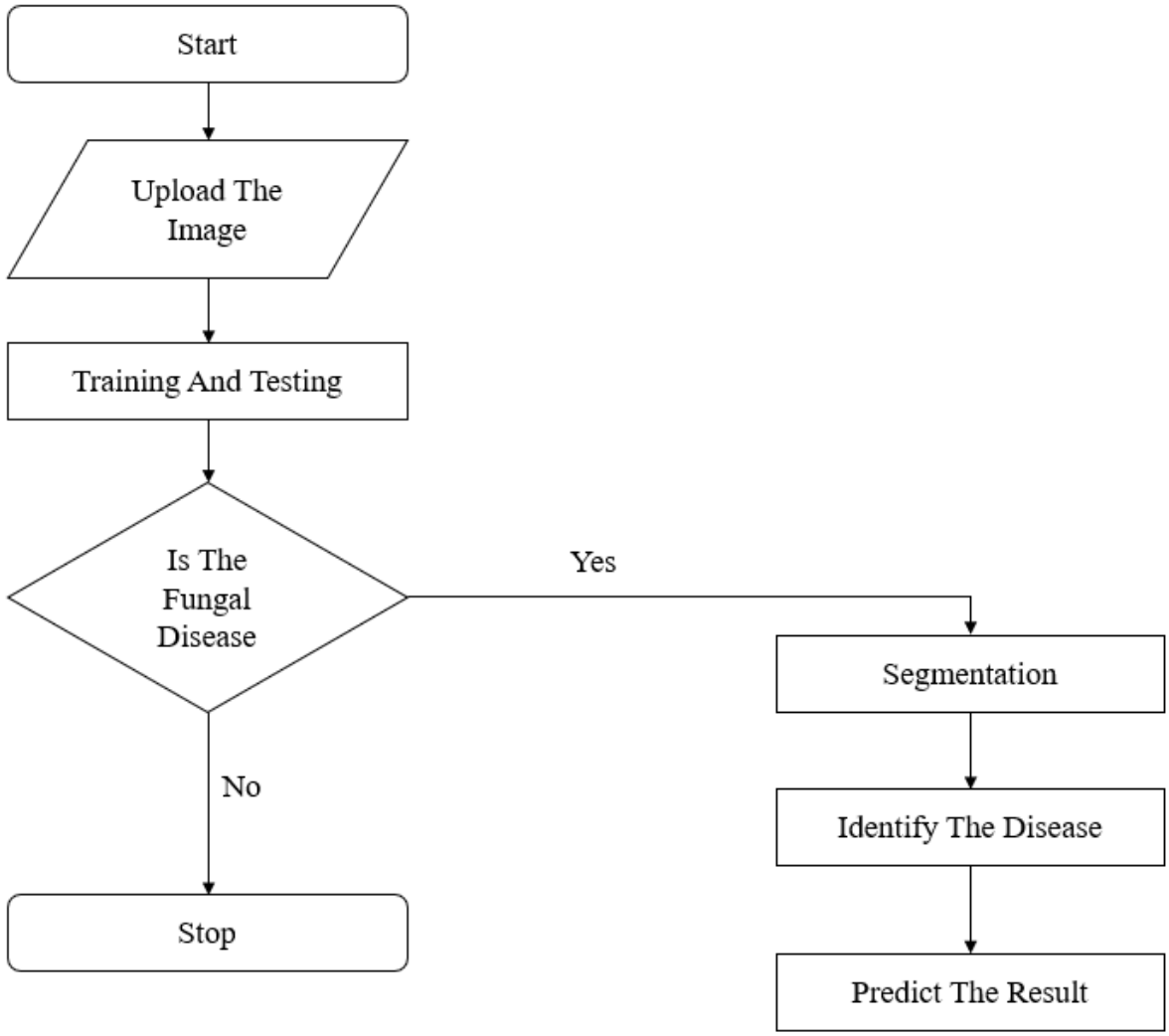

Figure 3.3 Dataflow Diagram

3.4.4 Class Diagram

A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modelling Language (UML). In this context, a class defines the methods and variables in an object, which isa specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of Object-Oriented Programming (OOP). The concept is several years old but has been refined as OOP modelling paradigms have evolved. In a class diagram, the classes are arranged in groups that share common characteristics. A class diagram resembles a flowchart in which classes are portrayed as boxes, each box having three rectangles inside. The toprectangle contains the name of the class; the middle rectangle containsthe attributes of the class; the lower rectangle contains the methods, also called operations, of the class.

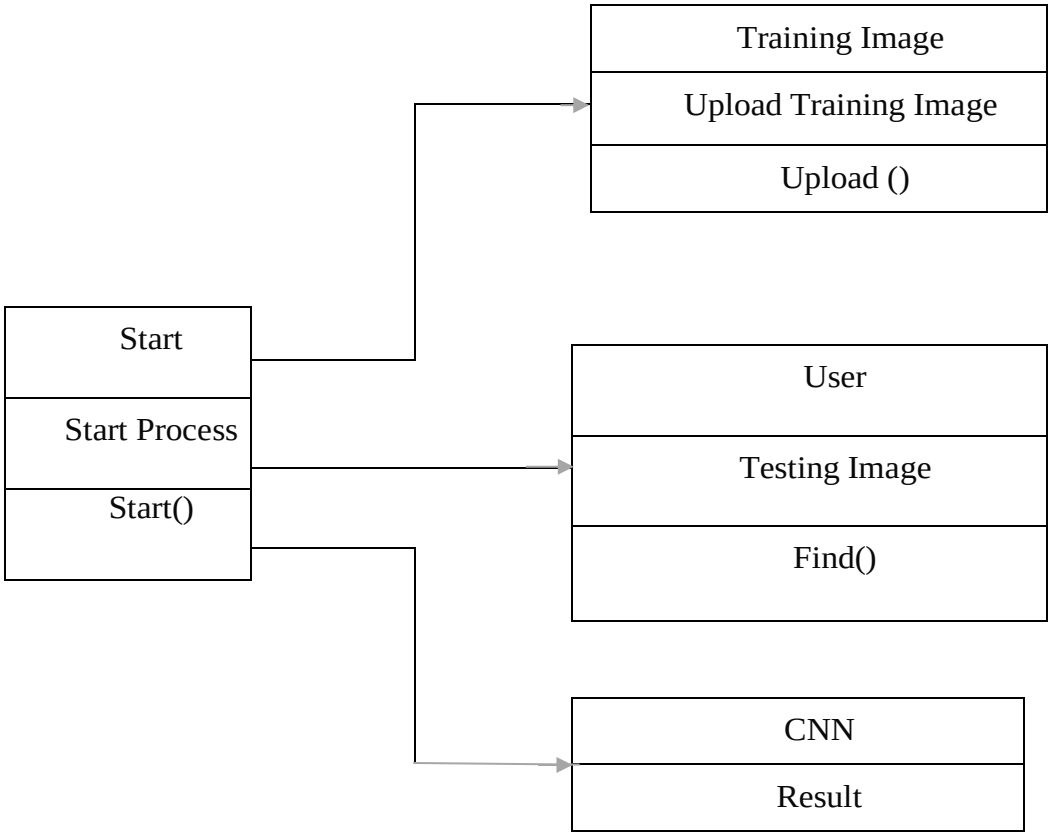

Figure 3.4 Class diagram

3.5 SYSTEM ARCHITECTURE

A system architecture or system design is the model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organizedin a way that supports reasoning about the structures and behaviors of the system. A system architecture can comprise system components, the expand systems developed, that will work together to implement the overall system.

=

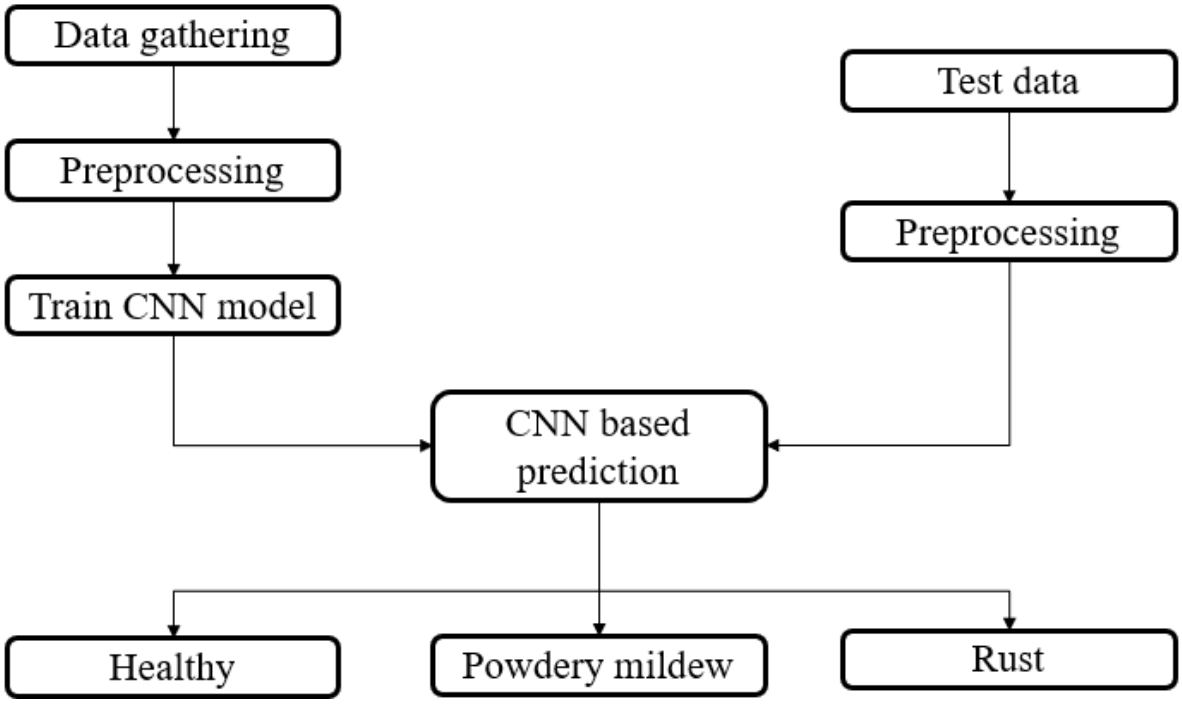

Figure 3.5 System Architecture

CHARPTER 4

MODULES

The Proposed System has four modules. The modules are:

4.1 Data Collection

4.2 Preprocessing

4.3 Model Training

4.4 Evaluation

4.5 Deployment

4.6 Testing and Validation

4.1 DATA COLLECTION

Gather a diverse set of plant leaf images encompassing healthy leaves, leaves with rust symptoms, and leaves with powdery mildew symptoms.Ensure the dataset includes various plant species, leaf angles, lighting conditions, and backgrounds to enhance model robustness. Obtain labeled data where each image is annotated with its corresponding health status (healthy, rust, or powdery mildew).

4.2 PREPROCESSING

Resize all images to a uniform dimension suitable for input into the CNN model. Normalize pixel values to a common scale to standardize data distribution and improve model convergence. Augment the dataset through techniques like rotation, flipping, and brightness adjustments to increase dataset diversity and prevent overfitting

4.3 MODEL TRAINING

Design a Conv-9 DCNN architecture suitable for plant leaf disease classification. Train the Conv-9 DCNN model using the preprocessed dataset, optimizing for classification accuracy through the Adam (Adaptive Moment Estimation) optimization algorithm.

4.4 EVALUATION

Evaluate the trained model's performance on a separate test dataset, calculating metrics such as accuracy for each class (healthy, rust, powdery mildew). Also, this evaluation process metrics such as recall, precision, and F1 score are calculated alongside a confusion matrix and confidence scores.

4.5 DEPLOYMENT

Integrate the trained DCNN model into a user-friendly application or web interface for easy accessibility by farmers or researchers. Optimize the model for inference speed and resource efficiency to enable real-time or near-real-time diagnosis of plant leaf health. Provide clear and interpretable results, including confidence scores or probability estimates for each disease class, to aid decision-making by end-users.

4.6 TESTING AND VALIDATION

Conduct thorough testing of the deployed model under various scenarios, including different plant species, environmental conditions, and disease severities. Validate the model's performance through field trials or real-world deployments, gathering feedback from users to refine and improve the system. Continuously monitor and update the model to adapt to emerging plant diseases or changes in leaf characteristics over time.

CHAPTER 5

CONCLUSION

5.1 CONCLUSION

In conclusion, the utilization of Convolutional Neural Networks (CNNs) for the detection of fungal diseases in plant leaves using leaf images marks a significant advancement in agricultural technology. Through deep learning techniques, CNNs have demonstrated remarkable accuracy and efficiency in identifying various fungal infections, enabling prompt intervention and improved crop health. This approach offers several key benefits, including automation of disease diagnosis, early detection of infections, and precise management strategies. By leveraging CNNs, farmersand agricultural experts can make informed decisions to mitigate the impact of fungal diseases on crop yields and quality. Looking ahead, further research and development are essential for enhancing the robustness and scalability of CNN models. This includes the creation of larger and more diverse datasets, exploration of advanced CNN architectures, and integration with emerging technologies for remote monitoring and decision support. Overall, the application of CNNs for fungal disease detection in leaf images holds great promise for revolutionizing agricultural practices. By harnessing the power of deep learning, we can pave the way for sustainable and resilientcrop production, ensuring food security for generations to come.

5.2 FUTURE ENHANCEMENT

In the future, enhancing the detection of fungal diseases in leaf images through Convolutional Neural Networks (CNNs) will involve several key

areas of development. One critical aspect is the expansion of datasets, encompassing images of diverse plant species, disease severities, and environmental conditions, to improve model performance and generalization capabilities. Additionally, exploring and implementing advanced CNN architectures, such as attention mechanisms, recurrent neural networks (RNNs), or graph neural networks (GNNs), can potentially enhance the model's ability to capture complex patterns within leaf images, leading tomore accurate disease detection. Leveraging transfer learning techniques, where pre-trained CNN models are fine-tuned on specific fungal diseasedetection tasks, will accelerate model training and improve performance, particularly in scenarios with limited training data. Integrating information from multiple modalities, such as spectral imaging or thermal imaging, alongside leaf images, can provide complementary data sources for more comprehensive disease detection and characterization. Developing CNN models optimized for real-time processing will enable on-the-fly disease detection in agricultural settings, facilitating immediate decision-making and intervention strategies. Enhancing the interpretability and explainability of CNN models through techniques such as attention maps or saliency maps will improve user trust and facilitate understanding of model decisions. Finally, integrating CNN-based disease detection models into smart farming systems and agricultural IoT platforms will enable automated and continuous monitoring of plant health, leading to more proactive and efficient disease management practices. By addressing these areas of enhancement, the future of fungal disease detection in leaf images through CNNs holds great potential for revolutionizing agricultural practices and contributing to global food security.

CHAPTER 6

APPENDIX - 1

6.1 SAMPLE CODE

```
<h1>Leaf Disease Classification - Keras CNN .. With 96% Accuracy</h1>

<img

style="width : 100%;margin: auto;"

src="https://www.google.com/search?q=leaf+disease&sxsrf=ALiCzsb-Fr5LvT82eoxXJt3fDQmP8Qo_cA:1654161628719&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiGpZ-iuI74AhVKBaYKHYEJBIMQ_AUoAXoECAIQAw&biw=1707&bih=806&dpr=1.13#imgrc=BJHS-ZYd7zmuTM&imgdii=O8NuCBPDZa8ipM>

import warnings

warnings.filterwarnings("ignore")

import tensorflow as tf

import matplotlib.pyplot as plt

tf.compat.v1.set_random_seed(0)

from tensorflow import keras

import numpy as np

np.random.seed(0)

import itertools

from keras.preprocessing.image import image_dataset_from_directory
```

```
from tensorflow.keras.layers.experimental.preprocessing import Rescaling

from sklearn.metrics import precision_score, accuracy_score, recall_score, confusion_matrix, ConfusionMatrixDisplay

<h1>Data Loading</h1>

<p>Setting up Image Data Generators<p>

#train_gen = image_dataset_from_directory(directory="../input/new-plant-diseases-dataset/train",image_size=(256, 256))

#test_gen = image_dataset_from_directory(directory="../input/new-plant-diseases-dataset/valid",image_size=(256, 256))


train_gen = image_dataset_from_directory(directory="/kaggle/input/powdery-and-rust/Plant Data/Train/Train",
                                         image_size=(256, 256))

test_gen = image_dataset_from_directory(directory="/kaggle/input/powdery-and-rust/Plant Data/Test/Test",
                                        image_size=(256, 256))


rescale = Rescaling(scale=1.0/255)

train_gen = train_gen.map(lambda image,label:(rescale(image),label))

test_gen = test_gen.map(lambda image,label:(rescale(image),label))

<h1>Data Engineering</h1>
```

Since the data is already augmented, there is no requirement of data engineering. Feature scaling is automatically done by image generators

<h1>Modelling</h1>

model = keras.Sequential()

model.add(keras.layers.Conv2D(32,(3,3),activation="relu",padding="same", input_shape=(256,256,3)))

model.add(keras.layers.Conv2D(32,(3,3),activation="relu",padding="same"))

model.add(keras.layers.MaxPooling2D(3,3))

model.add(keras.layers.Conv2D(64,(3,3),activation="relu",padding="same"))

model.add(keras.layers.Conv2D(64,(3,3),activation="relu",padding="same"))

model.add(keras.layers.MaxPooling2D(3,3))

model.add(keras.layers.Conv2D(128,(3,3),activation="relu",padding="same"))

model.add(keras.layers.Conv2D(128,(3,3),activation="relu",padding="same"))

model.add(keras.layers.MaxPooling2D(3,3))

```
model.add(keras.layers.Conv2D(256,(3,3),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(256,(3,3),activation="relu",padding="same"))

model.add(keras.layers.Conv2D(512,(5,5),activation="relu",padding="same"))
model.add(keras.layers.Conv2D(512,(5,5),activation="relu",padding="same"))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(1568,activation="relu"))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(38,activation="softmax"))

opt = keras.optimizers.Adam(learning_rate=0.0001)
model.compile(optimizer=opt,loss="sparse_categorical_crossentropy",metrics=['accuracy'])
model.summary()
ep = 32
```

```
history = model.fit_generator(train_gen,
        validation_data=test_gen,
        epochs = ep)
model.save('/kaggle/working/model.h5')
model.save_weights('/kaggle/working/weights.h5')
<h1>Metrics</h1>
plt.figure(figsize = (20,5))
plt.subplot(1,2,1)
plt.title("Train and Validation Loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.plot(history.history['loss'],label="Train Loss")
plt.plot(history.history['val_loss'], label="Validation Loss")
plt.xlim(0, 10)
plt.ylim(0.0,1.0)
plt.legend()

plt.subplot(1,2,2)
plt.title("Train and Validation Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
```

```
plt.plot(history.history['accuracy'], label="Train Accuracy")

plt.plot(history.history['val_accuracy'], label="Validation Accuracy")

plt.xlim(0, 9.25)

plt.ylim(0.75,1.0)

plt.legend()

plt.tight_layout()

labels = []

predictions = []

for x,y in test_gen:

    labels.append(list(y.numpy()))

    predictions.append(tf.argmax(model.predict(x),1).numpy())

predictions = list(itertools.chain.from_iterable(predictions))

labels = list(itertools.chain.from_iterable(labels))

print("Train Accuracy : {:.2f} %".format(history.history['accuracy'][-
1]*100))

print("Test Accuracy  : {:.2f} %".format(accuracy_score(labels,
predictions) * 100))

print("Precision Score : {:.2f} %".format(precision_score(labels,
predictions, average='micro') * 100))

print("Recall Score    : {:.2f} %".format(recall_score(labels, predictions,
average='micro') * 100))

<h3>Confusion Matrix</h3>
```

```
plt.figure(figsize= (20,5))
cm = confusion_matrix(labels, predictions)
disp = ConfusionMatrixDisplay(confusion_matrix=cm,
                              display_labels=list(range(1,4)))
fig, ax = plt.subplots(figsize=(15,15))
disp.plot(ax=ax,colorbar= False,cmap = 'YlGnBu')
plt.title("Confusion Matrix")
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.show()
# Define the class names
class_names = ['Healthy', 'Powdery Mildew', 'Rust']

# Choose one image from the training dataset
for images, labels in train_gen.take(1):
    sample_image = images[0] # Take the first image
    sample_label = labels[0] # Corresponding label

# Create an intermediate model to visualize activations
activation_model = tf.keras.Model(inputs=model.input,
outputs=[layer.output for layer in model.layers])
```

```
# Get activations for the sample image
activations = activation_model.predict(sample_image[np.newaxis, ...])

# Visualize original image
plt.figure(figsize=(6, 6))
plt.imshow(sample_image.numpy())
plt.title(f'Original Image - Class: {class_names[sample_label]}')
plt.axis('off')
plt.show()

# Visualize activations for each layer
layer_names = [layer.name for layer in model.layers]

for layer_name, activation in zip(layer_names, activations):
    if len(activation.shape) == 4: # Check if activation is 4-dimensional (convolutional layer)
        num_features = activation.shape[-1]

        plt.figure(figsize=(15, 6))
        plt.suptitle(f'Activations for Layer: {layer_name}', fontsize=16)
```

```
for j in range(num_features):
plt.subplot(2, num_features//2, j + 1)
plt.title(f'Feature {j + 1}')
                plt.imshow(activation[0, :, :, j], cmap='viridis')
                plt.axis('off')
            plt.tight_layout()
            plt.show()
        elif 'flatten' in layer_name: # Flatten layer
            plt.figure(figsize=(6, 6))
            plt.imshow(activation, cmap='viridis')
            plt.title(f'Activations for Flatten Layer: {layer_name}')
            plt.axis('off')
            plt.show()
        elif 'dense' in layer_name or 'dropout' in layer_name: # Dense and
    Dropout layers
            plt.figure(figsize=(6, 6))
            plt.plot(activation[0], 'bo')
            plt.title(f'Activations for {layer_name}')
            plt.xlabel('Neuron Index')
            plt.ylabel('Activation Value')
            plt.show()
```

6.2 SCREENSHOTS

```
# Model Training
history = model.fit(
    train_augmented_gen,
    epochs=10,
    validation_data=val_gen
)
```

```
Epoch 1/10
42/42 [==============================] - 226s 5s/step - loss: 1.4374 - accuracy: 0.3245 - val_loss: 1.1182 - val_accuracy: 0.3333
Epoch 2/10
42/42 [==============================] - 222s 5s/step - loss: 1.1247 - accuracy: 0.3782 - val_loss: 1.1431 - val_accuracy: 0.3333
Epoch 3/10
42/42 [==============================] - 220s 5s/step - loss: 0.9329 - accuracy: 0.5159 - val_loss: 0.7830 - val_accuracy: 0.5833
Epoch 4/10
42/42 [==============================] - 220s 5s/step - loss: 0.6278 - accuracy: 0.7012 - val_loss: 0.6949 - val_accuracy: 0.6500
Epoch 5/10
42/42 [==============================] - 220s 5s/step - loss: 0.5698 - accuracy: 0.7489 - val_loss: 0.6752 - val_accuracy: 0.6833
Epoch 6/10
42/42 [==============================] - 221s 5s/step - loss: 0.4190 - accuracy: 0.8298 - val_loss: 0.6907 - val_accuracy: 0.7500
Epoch 7/10
42/42 [==============================] - 221s 5s/step - loss: 0.2962 - accuracy: 0.8850 - val_loss: 0.4226 - val_accuracy: 0.8667
Epoch 8/10
42/42 [==============================] - 222s 5s/step - loss: 0.1861 - accuracy: 0.9357 - val_loss: 0.3937 - val_accuracy: 0.8667
Epoch 9/10
42/42 [==============================] - 221s 5s/step - loss: 0.1624 - accuracy: 0.9410 - val_loss: 0.4184 - val_accuracy: 0.8500
Epoch 10/10
42/42 [==============================] - 220s 5s/step - loss: 0.1447 - accuracy: 0.9516 - val_loss: 0.2062 - val_accuracy: 0.9000
```

Figure 6.1 Model Trained Image

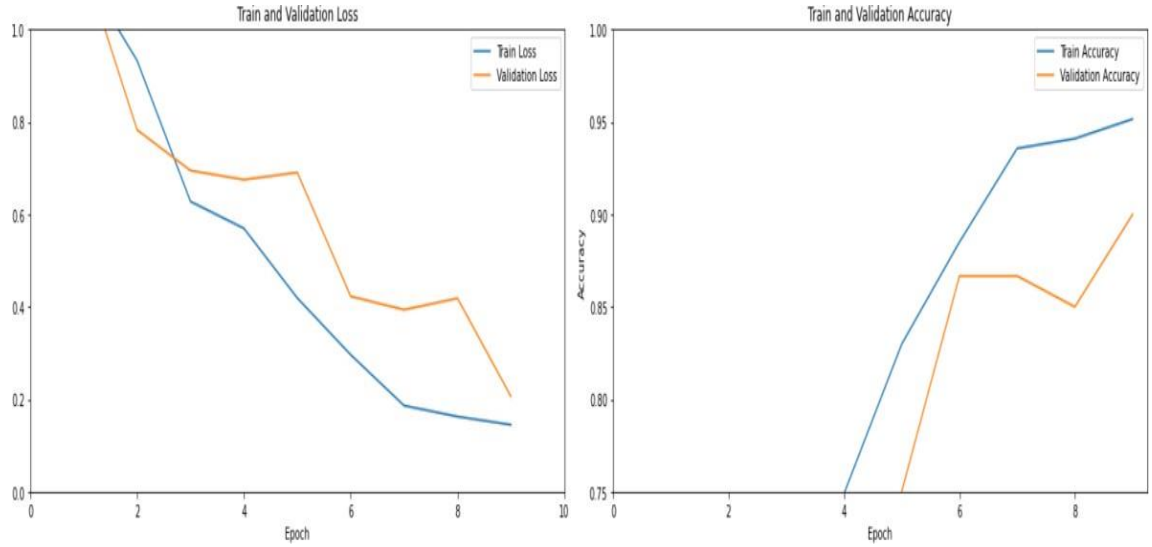

Figure 6.2 Model's Accuracy During Training.

```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

# Assuming labels and predictions are already defined

print("Train Accuracy  : {:.2f} %".format(history.history['accuracy'][-1]*100))
print("Test Accuracy   : {:.2f} %".format(accuracy_score(labels, predictions) * 100))
print("Precision Score : {:.2f} %".format(precision_score(labels, predictions, average='micro') * 100))
print("Recall Score    : {:.2f} %".format(recall_score(labels, predictions, average='micro') * 100))
print("F1 Score        : {:.2f} %".format(f1_score(labels, predictions, average='micro') * 100))
```

```
Train Accuracy  : 95.16 %
Test Accuracy   : 91.33 %
Precision Score : 91.33 %
Recall Score    : 91.33 %
F1 Score        : 91.33 %
```

Figure 6.3 The Performance Metrices Output

REFERENCES

1. A. M. Mostafa, S. A. Kumar, T. Meraj, H. T. Rauf, A. A. Alnuaim, and M. A. Alkhayyal (2021) 'Guava disease detection using deep convolutional neural networks: A case study of guava plants,' Appl. Sci., vol. 12, no. 1, p. 239.

2. Gharge and P. Singh (2016) 'Image processing for soybean disease classification and severity estimation' Emerging Research in Computing, Information, Communication and Applications, N. R. Shetty, N. H. Prasad, and N. Nalini, Eds. New Delhi, India: Springer, pp. 493–500.

3. I. Kandel and M. Castelli (2018) 'The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset' ICT Exp., vol. 6, no. 4, pp. 312–315.

4. J. W. Orillo, J. Dela Cruz, L. Agapito, P. J. Satimbre, and I. Valenzuela (2014) 'Identification of diseases in Rice plant (oryza sativa) using back propagation artificial neural network' in Proc. Int. Conf. Humanoid, Nanotech nol., Inf. Technol., Commun. Control, Environ. Manage. (HNICEM), pp. 1–6.

5. K. P. Ferentinos (2018) 'Deep learning models for plant disease detection and diagnosis' Comput. Electron. Agricult., vol. 145, pp. 311–318.

6. Lili Li, Shujuan Zhang, Bin Wang (2021) 'Plant Disease Detection

and Classification by Deep Learning - A Review', IEEE Access Vol. 9, pp. 56683- 56698.

7. M. A. Khan, M. I. U. Lali, M. Sharif, K. Javed, K. Aurangzeb, S. I. Haider, A. S. Altamrah, and T. Akram (2019) 'An optimized method for segmentation and classification of apple diseases based on strong correlation and genetic algorithm based feature selection,'' IEEE Access, vol. 7, pp. 46261–46277.

8. M. Sharif, M. A. Khan, Z. Iqbal, M. F. Azam, M. I. U. Lali, and M. Y. Javed (2018) 'Detection and classification of citrus diseases in agriculture based on optimized weighted segmentation and feature selection' Comput. Electron. Agricult., vol. 150, pp. 220–234.

9. Peng Jiang, Yuehan Chen, Bin Liu, Dongjian He, Chunquan Liang (2019) 'Real-Time Detection of Apple Leaf Diseases Using Deep Learning Approach Based on Improved Convolutional Neural Networks', IEEE Access Vol.7, pp. 59069- 59080.

\

10. R. Thangaraj, S. Anandamurugan, and V. K. Kaliappan (2021) 'Automated tomato leaf disease classification using transfer learning-based deep convolution neural network' J. Plant Diseases Protection, vol. 128, no. 1, pp. 73–86.

11. S. Coulibaly, B. Kamsu-Foguem, D. Kamissoko, and D. Traore (2019) 'Deep neural networks with transfer learning in millet crop images,' Comput. Ind., vol. 108, pp. 115–120.

12. S. P. Mohanty, D. P. Hughes, and M. Salathe (2016)‘‘Using deep learning for image-based plant disease detection,’’ Frontiers Plant Sci., vol. 7, pp. 1–10.

13. S. Theodoridis (2020) ‘Chapter 18 - neural networks and deep learning in Machine Learning’, 2nd ed., S. Theodoridis, Ed. New York, NY, USA: Academic Press, pp. 901–1038.

14. Z. Dong, X. Chen, W. Jia, S. Du, K. Muhammad, and S.-H. Wang (2019) ‘Image based fruit category classification by 13-layer deep convolutional neural network and data augmentation’ Multimedia Tools Appl., vol. 78, no. 3, pp. 3613–3632.

15. Z. Zhang, Y. Li, F. Wang, and X. He (2014) ‘A particle swarm optimization algorithm for neural networks in recognition of maize leaf diseases,’ Sensors Transducers, vol. 166, no. 3, pp. 181–189.