A Project Report

on

Plant Leaf Disease Detection Using Deep Learning

submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

19WH1A05B7	Ms. Rangu Dimple Bhavani
20WH5A0511	Ms. Boddu Tejaswi
20WH5A0512	Ms. Avusula Sowmya

under the esteemed guidance of

Mr. K. Bhargav Ram
Assistant Professor

Department of Computer Science and Engineering

BVRIT HYDERABAD
College of Engineering for Women

(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

June, 2023

DECLARATION

We hereby declare that the work presented in this project entitled **"Plant Leaf Disease Detection Using Deep Learning"** submitted towards completion of Project Work in IV year of B.Tech., CSE at 'BVRIT HYDERABAD College of Engineering For Women', Hyderabad is an authentic record of our original work carried out under the guidance of K. Bhargav Ram, Assistant Professor, Department of CSE.

Rangu Dimple Bhavani
(19WH1A05B7)

Boddu Tejaswi
(20WH5A0511)

Avusula Sowmya
(20WH5A0512)

BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "**Plant Leaf Disease Detection Using Deep Learning**" is a bonafide work carried out by Rangu Dimple Bhavani (19WH1A05B7); Boddu Tejaswi (20WH5A0511); Avusula Sowmya(20WH5A0512) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

The results embodied in the project work have not been submitted to any other University or Institute for the award of any degree or diploma.

Head of the Department
Dr. E. Venkateswara Reddy
Professor and HoD,
Department of CSE

Guide
K. Bhargav Ram
Assistant Professor

External Examiner

ACKNOWLEDGEMENT

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. E. Venkateswara Reddy, Professor & HOD**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Mr. K. Bhargav Ram, Assistant Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement, and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. Rangu Dimple Bhavani

(19WH1A05B7)

Ms. Boddu Tejaswi

(20WH5A0511)

Ms. Avusula Sowmya

(20WH5A0512)

Contents

S.No.	Topics	Page No.
	Abstract	i
	List of Figures	ii
1.	INTRODUCTION	1
	1.1 Objective	2
	1.2 Methodology	2
	1.2.1 Dataset	2
	1.2.2 Proposed System	4
	1.3 Organization of Project	4
2.	LITERATURE REVIEW	6
3.	THEORETICAL ANALYSIS OF THE PROPOSED PROJECT	11
	3.1 Requirements Gathering	11
	3.1.1 Software Requirements	11
	3.1.2 Hardware Requirements	11
	3.2 Technological Description	11
4.	DESIGN	13
	4.1 Introduction	13
	4.2 Architecture Design	16
	4.3 Algorithms	16
	4.3.1 CNN	16
	4.3.2 VGG19	21
5.	IMPLEMENTATION	26
	5.1 Coding	26
	5.2 Training Dataset	26
	5.3 Evaluation Metrics	27
6.	CONCLUSION AND FUTURE SCOPE	35
7.	REFERENCES	36
8.	APPENDIX	38

ABSTRACT

Early Plant disease detection and diagnosis are critical tasks in agriculture to prevent crop losses and economic damages. Manual methods are time-consuming and prone to errors, necessitating automated solutions. Deep learning models, including VGG19 and CNN, have gained popularity for automating this process. In this study, we evaluate the performance of VGG19 and CNN using a curated plant leaf diseases dataset from Kaggle. The dataset encompasses a diverse range of plant species, such as apple, blueberry, cherry, corn, grape, orange, peach, pepper, potato, raspberry, soybean, squash, strawberry, and tomato, focusing on the identification of diseases affecting their leaves.

Our aim is to enhance the accuracy and efficiency of plant disease detection through deep learning models. By training and evaluating VGG19 and CNN on this dataset, we seek to determine the model that achieves superior classification performance for different plant leaf diseases. The outcomes of this research contribute to the advancement of precision agriculture, offering a reliable tool for farmers and agricultural experts to swiftly detect and diagnose plant diseases, leading to timely interventions and effective management strategies. The proposed automated approach has the potential to improve crop health, optimize resource utilization, and promote sustainable agricultural practices.

LIST OF FIGURES

S.No.	Description	Page No.
1.	Dataset	3
2.	Workflow design	16
3.	Convolution in CNN	17
4.	Max Pooling In CNN	18
5.	Flattening in CNN	19
6.	Basic CNN Architecture	20
7.	Layers in CNN	21
8.	Layers in VGG19	25
9.	Training Dataset	27
10.	Training and validation accuracy of CNN Model	28
11.	Training and validation accuracy of VGG19 Model	29
12.	Testing Accuracy	30
13.	Training Accuracy	30
14.	Web page	32
15.	Detector Page	33
16.	Upload Image	33
17.	Disease Description and steps	33

LIST OF TABLES

S.No.	Description	Page No.
1	Comparison Table	29

1. INTRODUCTION

Plant leaf disease detection plays a crucial role in ensuring healthy and productive crops in agriculture. Early detection and accurate diagnosis of diseases can help farmers take timely and targeted actions to prevent the spread of diseases and minimize crop losses. Traditional methods of disease detection often rely on manual inspection, which can be time-consuming, labor-intensive, and subject to human errors. With advancements in machine learning and computer vision, deep learning models have emerged as a promising approach for automating and improving the efficiency of plant leaf disease detection.

In this study, we focus on utilizing deep learning techniques for the detection and diagnosis of plant leaf diseases using a new and comprehensive plant leaf diseases dataset. This dataset, specifically curated for this research, consists of a wide range of images capturing various diseases affecting plants such as apple, blueberry, cherry, corn, grape, orange, peach, pepper, potato, raspberry, soybean, squash, strawberry, and tomato. Each image is labeled with the corresponding disease type, providing a valuable resource for training and evaluating deep learning models.

The main objective of this research is to develop an accurate and efficient plant leaf disease detection system using deep learning models. We specifically employ state-of-the-art deep learning architectures, such as convolutional neural networks (CNNs) and VGG19, known for their effectiveness in image recognition tasks. These models have shown remarkable performance in various domains and are well-suited for analyzing complex visual data.

To achieve our goal, we follow a systematic approach. First, we preprocess and augment the dataset to enhance its quality and diversity. Next, we train the deep learning models using a combination of labeled images, applying transfer learning techniques to leverage pre-trained weights from large-scale image datasets. During the training process, the models learn to extract meaningful features from the input images, enabling them to distinguish between healthy leaves and those affected by diseases.

Once the models are trained, we evaluate their performance using various metrics such as accuracy, precision, recall, and F1-score. These metrics provide insights into the models' ability to correctly classify diseased and healthy leaves. We also assess the models' efficiency in terms of computational requirements and inference time to evaluate their practical usability in real-world scenarios.

By leveraging deep learning models for plant leaf disease detection, we aim to significantly improve the accuracy and efficiency of disease diagnosis in agriculture. The outcomes of this research can have far-reaching implications, enabling farmers and agricultural professionals to detect diseases early, implement appropriate control measures, and optimize crop management practices. Ultimately, this can lead to increased crop yields, reduced reliance on pesticides, and improved sustainability in agriculture.

In summary, this study focuses on harnessing the power of deep learning models for plant leaf disease detection using a new and comprehensive dataset. By leveraging the capabilities of CNNs and VGG19, we aim to develop an accurate and efficient system that can aid in the early detection and diagnosis of diseases affecting plants. The findings of this research have the potential to revolutionize disease management practices in agriculture, contributing to the sustainable growth of the farming industry.

1.1 Objective:

The primary goal of this project is to utilize deep learning techniques for early detection of plant leaf diseases. The system will enable users to explore the database, access a collection of leaf samples, capture leaf images, and perform analysis using the deep learning model. By leveraging advanced technology, this project aims to provide a user-friendly platform for efficient and accurate plant leaf disease detection.

1.2 Methodology:

1.2.1 Dataset

The New Plant Diseases Dataset on Kaggle is a comprehensive and extensive collection of RGB images specifically focused on plant leaves. This dataset is publicly available and

consists of a total of 87,900 images. These images include both healthy leaves and leaves affected by diseases or disorders.

The dataset covers a wide range of crop species, including apple, blueberry, cherry, grape, orange, peach, pepper, potato, raspberry, soy, squash, strawberry, and tomato. In total, there are 14 crop species represented in the dataset.

Within the dataset, there is detailed information about disease severity. The dataset encompasses 38 distinct classes, which include the 14 crop species mentioned earlier and 26 specific disorders or diseases affecting these plants.

This New Plant Diseases Dataset provides a valuable resource for researchers, practitioners, and enthusiasts interested in the field of plant pathology. The dataset's large size and diversity enable the development and evaluation of advanced machine learning and deep learning models for accurate and efficient plant disease detection, classification, and diagnosis.

This dataset is freely available at https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset

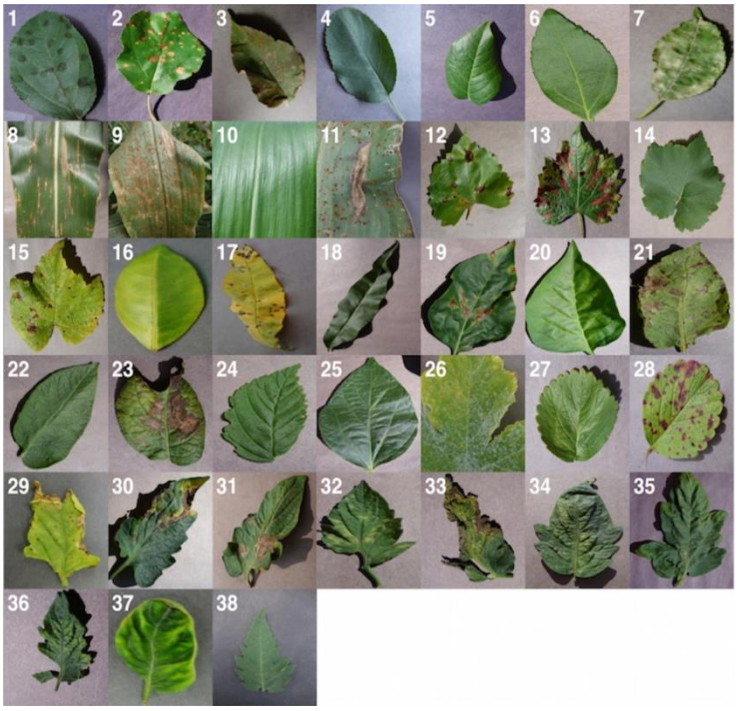

Fig. 1.Dataset

1.2.2 Proposed System:

The proposed system of this project is to leverage deep learning techniques to detect plant leaf diseases at an early stage using plant leaf images. The system will enable users to upload leaf images through a web page, users will receive valuable information regarding the plant name, disease identification, along with a comprehensive description of the disease and recommended steps for mitigation. By automating the disease detection process, this system aims to enable timely interventions and support farmers in effectively managing plant health, leading to improved crop yield and sustainable agricultural practices.

1.3 Organization of project:

Plant Leaf Disease Detection using CNN and VGG19 is a project that aims to develop an automated system for detecting diseases in plant leaves using convolutional neural networks (CNNs) and the VGG19 architecture. This project combines computer vision techniques and deep learning algorithms to accurately identify and classify various diseases affecting plant leaves.

The main objective of the project is to build a robust and accurate model that can detect and classify plant leaf diseases. The project aims to leverage the power of CNNs and specifically the VGG19 architecture to achieve high accuracy in disease identification.

To train and evaluate the model, a large dataset of plant leaf images is required. The project team would need to gather a diverse collection of plant images, including healthy leaves and leaves affected by various diseases. This dataset can be obtained from public repositories, research institutions, or by capturing images of plants in different environments.

Once the dataset is collected, it needs to be preprocessed before feeding it into the CNN model. Preprocessing may involve resizing the images, normalizing pixel values, and augmenting the dataset by applying transformations such as rotation, flipping, and cropping. These steps help in improving the model's generalization and robustness.

The project focuses on using CNNs and specifically employs the VGG19 architecture. VGG19 is a deep convolutional neural network that has shown excellent performance in image classification

tasks. It consists of 19 layers, including convolutional layers, pooling layers, and fully connected layers. The project team would implement and train the VGG19 model using a deep learning framework such as TensorFlow or PyTorch.

The dataset is divided into training and validation sets. The training set is used to train the CNN model by feeding the plant leaf images and their corresponding disease labels. The model learns to extract relevant features from the images and classify them into different disease categories. The validation set is used to evaluate the model's performance during training, helping to prevent overfitting and fine-tuning the hyperparameters.

Once the model is trained, it needs to be evaluated using a separate test dataset. This dataset contains images that the model has never seen before. The performance of the model is assessed by measuring metrics such as accuracy, precision, recall, and F1 score. These metrics provide insights into how well the model can detect and classify plant leaf diseases.

In this project, a comparison is made between the performance of the CNN model and the VGG19 architecture. The team would assess and compare the accuracy, computational efficiency, and overall performance of both models. This evaluation helps determine whether using the VGG19 architecture provides significant improvements over a standard CNN approach for plant leaf disease detection.

Once the model is trained and evaluated, it can be deployed in a practical setting for real-time disease detection. The model can be integrated into a user-friendly application or system where users can input images of plant leaves, and the model will predict the presence of any diseases.

Throughout the project, it is essential to document the entire process, including dataset details, model architecture, training configurations, evaluation results, and any challenges faced. This documentation helps in reproducibility and allows others to understand and build upon the project's work.

2. LITERATURE REVIEW

Title: A Classification Of Plant Leaf Diseases Using Machine Learning And Image Preprocessing Techniques

Authors: P. Sharma, P. Hans, and S. C. Gupta

Summary: This paper presents an artificial intelligence-based approach for automatic plant leaf disease detection and classification to increase agricultural productivity. The approach involves image collection, preprocessing, segmentation, and classification using a convolutional neural network. The model uses a dataset of over 20,000 images with 19 disease classes and can be extended with more data and disease categories. The accuracy can also be improved by tuning the hyperparameters. Remedies for the classified diseases can be included in the model, and it can be deployed on Android and iOS platforms to reach out to farmers. Overall, the system provides quick and easy detection and classification of plant leaf diseases to prevent spread and enhance crop quality and quantity.

Title: Detection of Plant Leaf-based Diseases Using Machine Learning Approach

Authors: P. Chaitanya Reddy, R. M. S. Chandra, P. Vadiraj, M. Ayyappa Reddy, T. R. Mahesh and G. Sindhu Madhuri

Summary: This research paper highlights the importance of using Machine Learning (ML) techniques to detect plant leaf-based diseases for improving agricultural productivity. The traditional approaches for disease detection are time-consuming and costly. ML algorithms such as Support Vector Machine (SVM) and Random Forest are analyzed for detecting leaf-based diseases with the aim of benefiting farmers with low cost and high accuracy results. Performance metrics such as Root Mean Square Error (RMSE), Peak Signal Noise Ratio (PSNR), and disease affected area are used to compare the accuracy of the algorithms. The main objective is to detect leaf diseases in captured images to improve crop yield and quality. The researchers plan to extend their work by considering more datasets and deep learning algorithms for achieving better accuracy results.

Title: Plant Disease Detection Using Machine Learning Techniques

Authors: D. Varshney, B. Babukhanwala, J. Khan, D. Saxena and A. K. Singh

Summary: Plant diseases can cause a significant decrease in agricultural productivity and food safety. Traditional machine-learning approaches have been used to detect plant diseases from digital images. In this paper, a transfer learning approach is proposed that utilizes a convolutional neural network (CNN) as a feature extractor and support vector machine (SVM) for classification. The model is evaluated on a benchmark dataset called PlantVillage and achieves an 88.77% training accuracy, outperforming previous methods. The proposed approach can identify different crops and disease severity of 38 distinct classes using a dataset of 54,306 pictures of healthy and damaged plant leaves. The paper provides a complete description of the entire technique, from image acquisition to deep CNN learning and classification. The study shows that deep learning can be used to enable autonomous plant disease detection through image categorization.

Title: A Novel Method of Plant Leaf Disease Detection Based on Deep Learning and Convolutional Neural Network

Authors: X. Guan

Summary: This study proposes a new plant disease detection approach by combining four CNN models and using a stacking method to achieve higher accuracy. The experiment used an open-source database of 36258 images of 10 plant species and 61 classes of healthy and diseased plant leaves. The accuracy rate achieved by the stacking method was 87%, which is a significant improvement compared to using a single CNN model. The use of CNN models, in combination with a voting mechanism, detected more plant diseases (61) compared to previous research. The trained machine has the potential to assist the economy in agriculture by preventing plant diseases in their early stages. Future research should focus on decreasing the number of parameters while maintaining accuracy, expanding the diversity of species that can be diagnosed and despite the exceptional performance achieved on the validation set, the model is not yet ready for practical use due to its high demand for computational power.

Title: Plant Disease Detection Using Machine Learning

Authors: Shima Ramesh, Ramachandra Hebbar, Niveditha M, Pooja R., Prasad Bhat N, Shashank N, Vinod P.V.

Summary: Crop diseases are a significant threat to food security, but their quick detection remains difficult in many parts of the world due to the absence of necessary infrastructure. This paper proposes using Random Forest to identify healthy and diseased leaves from datasets created through various phases of implementation, including dataset creation, feature extraction, training the classifier, and classification. The Histogram of an Oriented Gradient (HOG) is used to extract features from images. The algorithm was compared with other machine learning models for accuracy and achieved approximately 70% accuracy when trained using 160 images of papaya leaves. The accuracy can be increased by training the model with a vast number of images and using other local features together with global features such as SIFT, SURF, and DENSE along with BOVW. Overall, using machine learning to train large publicly available datasets can help detect plant diseases on a large scale.

Title: Detection of Plant Diseases Using Image Processing with Machine Learning

Authors: R. M. Alyas and A. S. Mohammed

Summary: This article focuses on using image processing and machine learning techniques to detect and categorize plant diseases from images. The paper reviews various techniques used in this area, including dataset size, preprocessing, segmentation, and classification techniques. The study used SVM and k-mean clustering techniques to extract color and texture information from plant disease images. The SVM classifier was found to be an effective tool for detecting and identifying plant diseases. The paper emphasizes the importance of saving plants from diseases since human life is dependent on nature and plants. The study aimed to locate the afflicted areas of the leaf and offer remedies to protect crops from contracting these diseases. Overall, the paper suggests that image processing and machine learning techniques can simulate human decision-making in detecting and categorizing various plant diseases.

Title: Plant Disease Detection using Convolutional Neural Networks

Authors: P. K. V, E. G. Rao, G. Anitha and G. K. Kumar

Summary: The loss of food due to contaminated crops is a significant problem in agriculture. This research introduces a novel approach to identify plant diseases using large convolution networks based on leaf image classification. The developed model can distinguish thirteen different types of plant diseases from healthy leaves. The approach is projected for the first time, and all necessary steps were taken by agricultural consultants to incorporate this disease recognition model. The deep CNN process was implemented using Python and PyCharm. The final overall accuracy of the model was 96.3%, and fine-tuning did not significantly improve precision, but augmentation had a stronger impact on producing acceptable results. This approach using deep learning has no overlap with related findings in the field of disease detection. However, there is still a lack of commercial approaches for plant disease identification.

Title: Early Detection of Plant Leaf Disease Using Convolutional Neural Networks

Authors: S. Pratapagiri, R. Gangula, R. G, B. Srinivasulu, B. Sowjanya and L. Thirupathi

Summary: Plant disease identification and prevention is a major challenge for smallholder farmers. The process of identifying and diagnosing the disease is time-consuming and requires expertise. Computer vision applications have shown potential in this area. A pre-trained CNN model was fine-tuned using pictures of different fruit plant leaves, covering various diseases and safe samples. The accuracy of the model was reported to be 98.7% in a controlled setting. The precision obtained depended on variables such as disease stage, type, context data, and object structure. Augmentation and transfer learning helped the model generalize with greater reliability. Overall, this research shows how CNNs can help smallholder farmers combat plant disease.

Title: Machine Learning Based Recognition of Crops Diseases By CNN

Authors: K. Naresh, G. Naga Satish, K. Bhargav Ram, Ch. Srinivasulu

Summary: This paper discusses the use of machine learning in identifying crop diseases and providing remedies through an automated vision system. The Agro Consultant application aids users in spotting plant species through their leaves, identifying crop species, and their diseases, and also providing remedies for diseased crops, weeds, and damaged pests. The system allows users to search a database, browse a list of collected leaf samples, and take pictures of leaves for analysis. Crop diseases are a major hazard to food security, but their swift discovery remains challenging due to the lack of necessary communication in many parts of the world. The authors trained a deep convolutional neural network using a public dataset of 12,673 leaf images to identify diseased plants. The application helps farmers identify diseased plants and improve accuracy and therapy dynamically in the future.

3. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

3.1 Requirements Gathering:

3.1.1Software requirements:

- Language Used: Python 3.8
- Operating System: Windows
- Software: Jupyter Notebook, Google Colab
- Packages: Matplotlib, Seaborn, TensorFlow, Flask, Scikit Learn

3.1.2Hardware requirements:

- Processor: Intel Core i3
- RAM: 4GB

3.2 Technologies Description:

1. Programming Language:

- Python: Python is the chosen programming language for this project due to its extensive libraries and frameworks for data analysis, machine learning, and deep learning. It offers a user-friendly syntax and a rich ecosystem of tools that facilitate the implementation of complex algorithms.

2. Image Processing and Data Analysis:

- OpenCV: OpenCV (Open Source Computer Vision Library) is a widely used library for image processing and computer vision tasks. It provides a range of functions for image loading, resizing, color space conversions, and other preprocessing tasks required for plant leaf image analysis.

- NumPy: NumPy is a fundamental library for numerical operations in Python. It offers efficient data structures and mathematical functions, particularly useful for handling and manipulating arrays of pixel values in plant leaf images.

- Pandas: Pandas is a powerful library for data manipulation and analysis. It provides data structures like DataFrames, which can be used to organize and process the dataset used in the project. Pandas enables easy data preprocessing and exploration.

3. Deep Learning Framework:

- TensorFlow: TensorFlow is a popular deep learning framework that provides a comprehensive ecosystem for building and training neural network models. It offers a high-level API called Keras, which simplifies the implementation of CNN and VGG19 architectures used in the project. TensorFlow provides

efficient computation on both CPUs and GPUs.

4. CNN and VGG19 Model Development:

- Keras: Keras is a high-level neural network library that runs on top of TensorFlow. It offers a user-friendly and intuitive API for building, training, and evaluating neural network models. Keras allows easy construction of CNN models with various layers, including convolutional layers, pooling layers, and fully connected layers. It also provides pre-trained VGG19 models that can be fine-tuned for plant leaf disease detection.

5. Data Visualization:

- Matplotlib: Matplotlib is a widely used plotting library in Python. It provides a variety of functions for creating static, animated, and interactive visualizations. Matplotlib can be used to visualize the plant leaf images, display model training curves, and generate informative graphs and charts for performance analysis.

- Seaborn: Seaborn is a statistical data visualization library built on top of Matplotlib. It offers high-level abstractions and aesthetically pleasing visualizations for exploring and presenting data. Seaborn can be used to create informative plots, such as bar plots or heatmaps, to visualize the distribution of plant leaf diseases or compare the performance of different models.

6. Web Application Framework:

- Flask: Flask is a lightweight web framework for Python. It can be used to develop a web-based interface for the plant leaf disease detection system. Flask allows easy integration of the trained CNN and VGG19 models, enabling users to upload plant leaf images and receive predictions through a user-friendly web interface.

7. Development Environment:

- Jupyter Notebook: Jupyter Notebook provides an interactive environment for prototyping and experimenting with code

4. DESIGN

4.1 Introduction:

By offering a comparative review of several deep learning techniques, the suggested methodology seeks to advance the discipline of Plant Leaf Disease Detection. The system's results will help identify the productive algorithm in locating misleading papers and provide insights for further research in this area.

The system consists of the following components:

1. Data Collection:

The first step involves collecting a diverse dataset of plant leaf images. The dataset should be organized into separate subdirectories, where each subdirectory represents a specific class or category of plant disease. This hierarchical directory structure facilitates the training and validation processes.

2. Pre-processing:

Data preprocessing is an essential step before feeding the images into the model. To perform this task, the code utilizes the Image Data Generator class from TensorFlow. The Image Data Generator class provides a range of functions for data augmentation, normalization, and resizing.

- Rescaling: The pixel values of the images are rescaled by dividing them by 255. This normalization step ensures that the pixel values fall within the range of 0 to 1, which is a standard practice for neural network inputs.

- Data Augmentation: To increase the diversity of the training data and improve the model's ability to generalize, various data augmentation techniques are applied. These include shearing, zooming, flipping, rotation, and shifting of the images. These transformations introduce variations and simulate different scenarios, enabling the model to learn robust representations.

- Target Size: All the images in the dataset are resized to a specified target size using the flow from directory method. This ensures that the input images have consistent dimensions, which is necessary for feeding them into the model.

- Batching: The training and validation generators are configured to generate images in batches of a specified size. Batching allows for efficient model training by processing a set number of images at a time, reducing memory usage, and enabling parallel processing.

- Class Encoding: The flow from directory method automatically assigns class labels to the images based on the directory structure. These class labels are encoded as categorical variables, enabling multi-class classification.

These internal steps are performed by the Image Data Generator and flow from directory methods to preprocess the images before training the convolutional neural network (CNN) model. The generated data batches are then fed into the model for training and validation.

3. Feature Extraction:

The core of the methodology involves defining the architecture of the deep learning model. In this case, a sequential model is created using the Sequential class from TensorFlow. The model comprises several convolutional layers (Conv2D) followed by activation layers, which apply non-linearity to the extracted features. Max pooling layers (MaxPooling2D) are incorporated to downsample the feature maps and reduce computational complexity. The flattened feature maps are then connected to dense layers (Dense) for classification. The final dense layer employs softmax activation to produce a probability distribution over the different classes, indicating the model's confidence for each class.

4. Algorithm Selection:

The system will identify the disease using different deep learning models like CNN(Convolutional Neural Network) and VGG19(Visual Geometry Group19).

5. Model Training:

The model is trained using the fit function, which takes the training generator, validation generator, and other parameters. The steps per epoch parameter specifies the number of batches to be processed per epoch. It is calculated as the total number of training samples divided by the batch size. The epochs parameter determines the number of times the model will iterate over the entire training dataset.

The validation data parameter is set to the validation generator to evaluate the model's performance on the validation set during training. The validation steps parameter specifies the number of batches to be processed for validation. It is calculated similarly to steps per epoch. During training, the model adjusts its weights based on the gradients computed from the forward and backward passes through the network, gradually improving its performance.

6. Model Evaluation:

After training, the model is evaluated on the validation set using the evaluate function. It returns the loss value and the accuracy of the model on the validation data. The accuracy metric represents the percentage of correctly classified samples from the validation set. It is a measure of how well the model generalizes to unseen data.

7. Comparative Analysis:

To compare the performance of CNN and VGG19, the model is trained and evaluated with varying architectures or hyperparameters. By comparing their accuracy metrics on the validation set, the VGG19 model performs better compared to the CNN model.

Result Visualization: The system provides a user-friendly interface, a web page where the user can upload an image. The uploaded image is then processed by the trained model to predict the plant disease. The predicted disease, along with its description and recommended steps, can be displayed to the user. This can be achieved using web development frameworks like Flask along with HTML, CSS, and JavaScript for the front end. The predicted result can be communicated back to the user using dynamic web pages. Visualizations, such as displaying the uploaded image and the predicted disease, can be implemented using image processing libraries and web technologies.

4.2 Architecture Diagram:

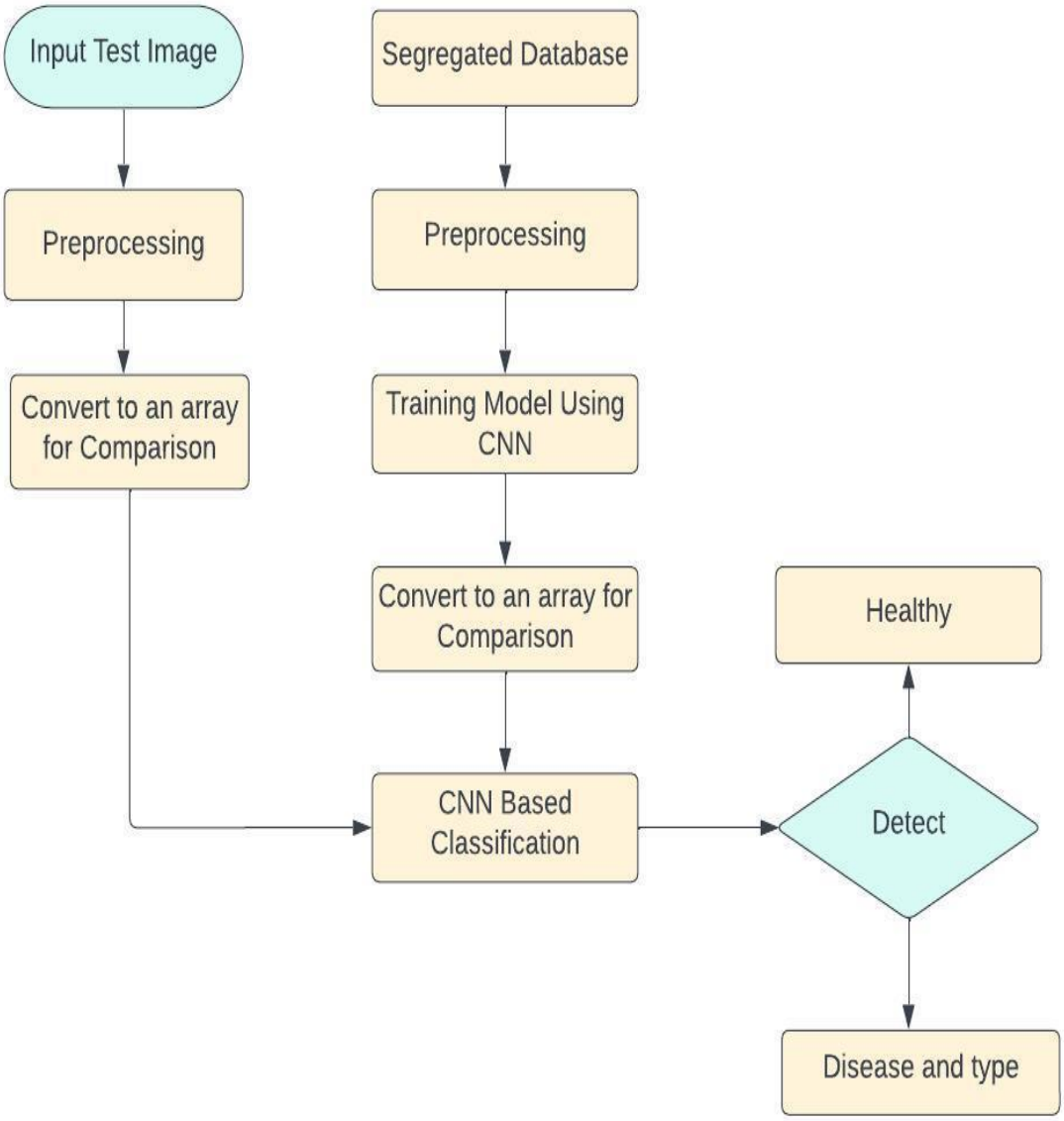

Fig. 2. Workflow Design

4.3 Algorithms:

4.3.1: CNN

Convolutional Neural Networks (CNNs) are a class of deep learning models that have been highly successful in various computer vision tasks, including image classification, object detection, and image segmentation. CNNs are specifically designed to automatically learn and extract hierarchical representations from visual data.

A CNN consists of multiple layers that are stacked sequentially to form the network architecture. The main types of layers in a CNN include convolutional layers, pooling layers, and fully connected layers.

1. Input Layers:

It's the layer in which we give input to our model. In CNN, Generally, the input will be an image or a sequence of images. This layer holds the raw input of the image with width 224, height 224, and depth 3.

2. Convolutional Layers:

Convolutional layers consist of multiple filters or kernels, each with learnable weights. These filters are small, spatially arranged matrices that slide or convolve across the input image. As they slide, they perform element-wise multiplications with the image pixels and accumulate the results to produce feature maps. The convolution operation captures local patterns and spatial relationships between pixels.

Convolutional layers have three main hyperparameters:

Number of Filters: Determines the number of features to extract. Each filter learns different aspects of the input data.

Filter Size: Specifies the spatial dimensions of the filters. Common filter sizes are 3x3, 5x5, or 7x7.

Stride: Controls the step size of the filter movement. A stride of 1 moves the filter pixel by pixel, while a larger stride skips pixels.

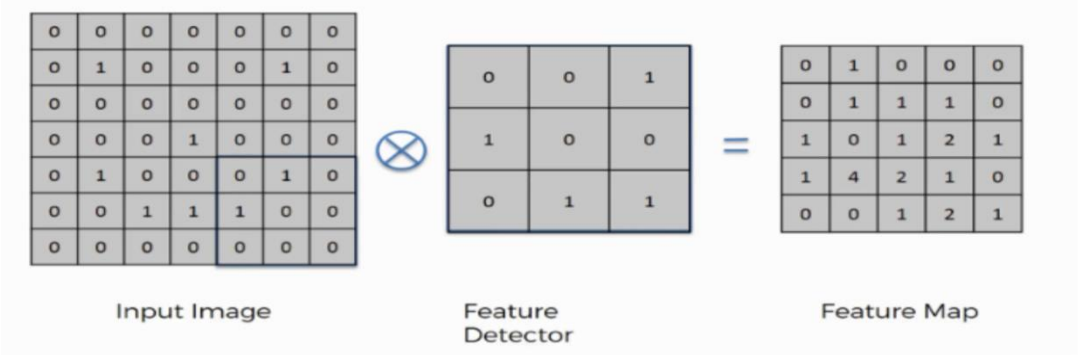

Fig. 3.Convolution in CNN

3. Pooling Layers:

Pooling layers are inserted between successive convolutional layers to reduce the spatial dimensions of the feature maps. The most common pooling operation is max pooling, which partitions the feature map into non-overlapping regions and keeps the maximum value within each region. Pooling helps to down sample the feature maps, reduce computational complexity, and increase the network's translational invariance.

Pooling layers have two primary hyperparameters:

Pool Size: Specifies the size of the pooling regions. Common pool sizes are 2x2 or 3x3.

Stride: Controls the step size of the pooling regions. A stride of 2 moves the pooling regions two units at a time.

Fig. 4.Max Pooling in CNN

4. Activation Functions:

Activation functions introduce non-linearity to the network, allowing it to learn complex relationships between the input data and the desired output. Commonly used activation functions in CNNs include Rectified Linear Unit (ReLU), sigmoid, and hyperbolic tangent (tanh). ReLU is widely used due to its simplicity and effectiveness in mitigating the vanishing gradient problem.

5. Flattening:

The resulting feature maps are flattened into a one-dimensional vector after the convolution and pooling layers so they can be passed into a completely linked layer for categorization or regression

Fig. 5.Flattening in CNN

6. Dropout Layers:

The Dropout layer is a mask that nullifies some neurons' contributions to the following layer while leaving all others unchanged. A Dropout layer can be applied to the input vector, nullifying some of its properties; however, it can also be applied to a hidden layer, nullifying some hidden neurons. Dropout layers are critical in CNN training because they prevent the training data from overfitting. If they aren't there, the first batch of training data has a disproportionately large impact on learning.

7. Fully Connected Layers:

After several convolutional and pooling layers, the output is flattened and fed into fully connected layers. These layers resemble a traditional neural network, where each neuron is connected to every neuron in the previous and subsequent layers. Fully connected layers capture global relationships in the data and produce the final output probabilities or predictions.

8. Training CNN:

CNNs are trained using a variant of the stochastic gradient descent (SGD) algorithm called backpropagation. The process involves forward propagation to compute the network's output, comparison of the predicted output with the ground truth labels, and backpropagation of the error to adjust the network's weights and biases. This iterative process continues for multiple epochs until the network converges to optimal weights.

9. Loss Functions:

Loss functions quantify the dissimilarity between the predicted output and the ground truth labels. For multi-class classification tasks, the cross-entropy loss is commonly used with the softmax activation function in the output layer.

10. Regularization Techniques:

To prevent overfitting and improve generalization, CNNs often incorporate regularization techniques. Dropout is a widely used regularization technique that randomly sets a fraction of the neurons' outputs to zero during training. It helps to reduce the co-adaptation of neurons and improves the model's robustness.

11. Transfer Learning:

Transfer learning is a powerful technique in deep learning that allows us to reuse the knowledge gained from pre-training on large-scale datasets. Instead of training a model from scratch, we initialize the model with pre-trained weights, which capture general image representations. This initialization provides a head start and reduces the training time for the new task or dataset. By leveraging the learned features, the model can generalize well to the new data, even when the new dataset is small or dissimilar to the original pre-training data. Transfer learning is widely used to achieve better performance, improve convergence, and overcome data scarcity in various computer vision applications.

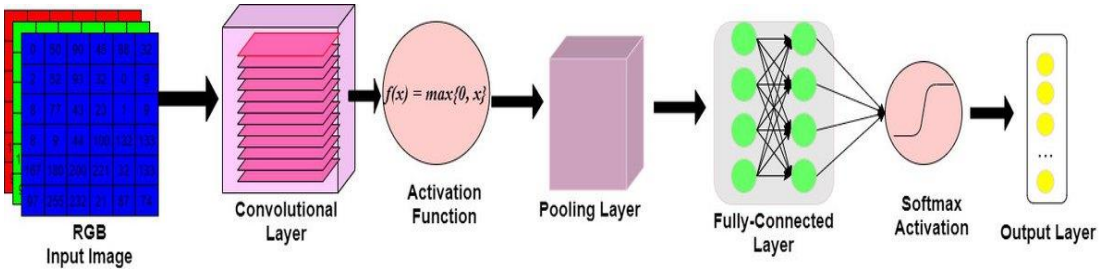

Fig. 6.Basic CNN Architecture

CNNs with convolutional and pooling layers have revolutionized computer vision tasks. They capture local patterns, spatial relationships, and hierarchical features in input images, making them powerful tools for image analysis. The careful design of architectures, activation functions, and pooling strategies, along with techniques like deepening layers and parameter sharing, contributes to their success.

CNNs find applications in various domains, including image classification, object detection, image segmentation, facial recognition, medical imaging, and autonomous driving. Their ability to automatically learn features from raw image data without relying on handcrafted features has made CNNs a cornerstone of modern computer vision systems.

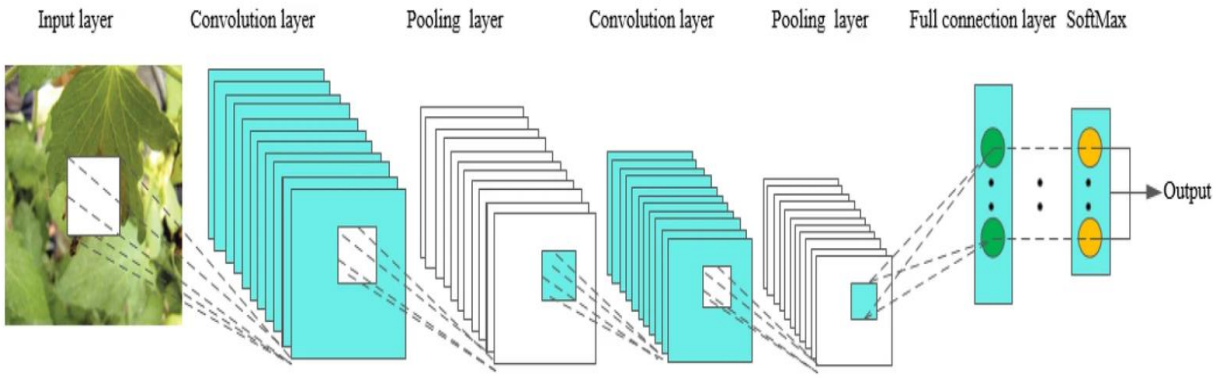

Fig. 7.Layers in CNN

In a CNN, the convolutional layers apply filters to extract local patterns and features from the input. Activation layers introduce non-linearities, enabling the network to learn complex relationships between features. Pooling layers reduce spatial dimensions, preserving important information while improving computational efficiency. Fully connected layers connect neurons, enabling the network to perform classification or regression tasks based on learned features. These layers collectively enable the CNN to understand and make predictions from input data, capturing meaningful features such as edges, textures, shapes, and higher-level representations. Each layer plays a crucial role in the feature extraction and decision-making process of the network.

4.3.2 VGG19

Plant diseases have a significant impact on crop yield and quality. Early detection and accurate diagnosis of these diseases are crucial for effective disease management. In recent years, deep learning models have shown promising results in various image classification tasks, including plant disease detection. One widely used deep learning model for image classification is the VGG19 architecture. This article aims to provide an overview of the VGG19 layers and their role in plant leaf disease detection.

Plant leaf disease detection is an important task in agriculture to ensure the health and productivity of crops. Deep learning techniques have shown promising results in automating this process. One such deep learning architecture commonly used for image classification tasks is VGG19, which stands for Visual Geometry Group 19.

1. VGG19 Architecture:

VGG19 is a convolutional neural network (CNN) architecture that was proposed by the Visual Geometry Group at the University of Oxford. It has 19 layers, including 16 convolutional layers and 3 fully connected layers. The architecture of VGG19 is known for its simplicity and uniformity, which allows for easier implementation and understanding.

2. Convolutional Layers and Max pooling layers:

The convolutional layers and max pooling layers in VGG19 work in tandem to extract hierarchical features from input images. These layers play a crucial role in capturing patterns and structures at different scales, enabling the network to learn discriminative features for plant leaf disease detection.

The convolutional layers in VGG19 perform convolutions on the input image using a set of learnable filters. Each filter slides across the input image, computing element-wise multiplications and accumulating the results to produce a feature map. These feature maps represent different visual patterns detected in the input.

In VGG19, there are a total of 16 convolutional layers, denoted as Conv1_1 to Conv4_4. The initial layers capture low-level features such as edges, corners, and textures, while the deeper layers extract more complex and abstract features. The number of filters in each convolutional layer gradually increases to learn more discriminative representations.

The max pooling layers in VGG19 reduce the spatial dimensions of the feature maps while retaining the most salient features. Max pooling is applied by dividing the feature map into non-overlapping regions and selecting the maximum value within each region. This operation effectively downsamples the feature maps, enabling the network to focus on the most informative features.

In VGG19, there are a total of 5 max pooling layers, denoted as Pool1 to Pool5. These layers follow specific convolutional layers to reduce the spatial resolution of the feature maps. Each max

pooling layer applies a 2x2 window with a stride of 2, halving the width and height of the feature maps. The pooling operation helps to achieve translation invariance and spatial robustness in feature extraction.

The combination of convolutional and max pooling layers in VGG19 allows the network to progressively learn and capture features of increasing complexity. The initial convolutional layers detect basic visual patterns, such as edges and textures, while the subsequent layers build upon these patterns to recognize more intricate structures and meaningful features.

The max pooling layers play a crucial role in reducing the spatial dimensions of the feature maps, which helps to control the computational complexity and the number of parameters in the network. By downsampling the feature maps, the network focuses on the most informative features while discarding redundant spatial information.

Overall, the convolutional and max pooling layers in VGG19 work synergistically to extract hierarchical features from input images. The convolutional layers capture patterns and structures at various scales, while the max pooling layers downsample the feature maps to retain the most salient information. This combined feature extraction process enables VGG19 to effectively learn discriminative representations for plant leaf disease detection, contributing to accurate classification and diagnosis of plant diseases.

Information about the working of convolutional layers:

1. Conv1_1: This is the first convolutional layer with 64 filters of size 3x3. It applies these filters to the input image to detect low-level features such as edges, corners, and textures.

2. Conv1_2: Similar to Conv1_1, this layer also has 64 filters of size 3x3. It further extracts low-level features from the input.

3. Pool1: This layer performs max pooling with a 2x2 window and a stride of 2. Max pooling reduces the spatial dimensions of the feature maps while preserving the most salient features.

4. Conv2_1: This layer has 128 filters of size 3x3 and extracts more complex features compared to the previous layers.

5. Conv2_2: Similar to Conv2_1, this layer also has 128 filters of size 3x3 and captures additional high-level features.

6. Pool2: This layer performs max pooling similar to Pool1.

7. Conv3_1: This layer has 256 filters of size 3x3 and continues to extract more complex features.

8. Conv3_2: Similar to Conv3_1, this layer also has 256 filters of size 3x3 and captures additional high-level features.

9. Conv3_3: This layer has 256 filters of size 3x3 and further deepens the network's feature extraction capabilities.

10. Conv3_4: Similar to Conv3_3, this layer also has 256 filters of size 3x3 and captures more high-level features.

11. Pool3: This layer performs max pooling similar to Pool1 and Pool2.

12. Conv4_1: This layer has 512 filters of size 3x3 and continues to extract more complex features.

13. Conv4_2: Similar to Conv4_1, this layer also has 512 filters of size 3x3 and captures additional high-level features.

14. Conv4_3: This layer has 512 filters of size 3x3 and further deepens the network's feature extraction capabilities.

15. Conv4_4: Similar to Conv4_3, this layer also has 512 filters of size 3x3 and captures more high-level features.

16. Pool4: This layer performs max pooling similar to Pool1, Pool2, and Pool3.

3. Fully Connected Layers:

After the convolutional layers, the feature maps are flattened and passed through the fully connected layers for classification. Let's explore the fully connected layers in VGG19:

17. FC1: This layer is fully connected with 4,096 neurons and uses the rectified linear unit (ReLU) activation function to introduce non-linearity.

18. FC2: Similar to FC1, this layer is also fully connected with 4,096 neurons and applies ReLU activation.

19. FC3: This is the final fully connected layer with 1,000 neurons, which corresponds to the number of output classes in the original VGG19 architecture. However, for plant leaf disease detection, this layer is typically modified to have the desired number of classes for disease classification.

VGG19 is a powerful CNN architecture for plant leaf disease detection. Its 19 layers, including 16 convolutional layers and 3 fully connected layers, enable the extraction of hierarchical features from input images. The convolutional layers capture low-level to high-level features, while the fully connected layers perform classification based on the learned features. By training VGG19 on a large dataset of plant leaf images, it can effectively classify and detect various diseases, contributing to more efficient and accurate agricultural practices.

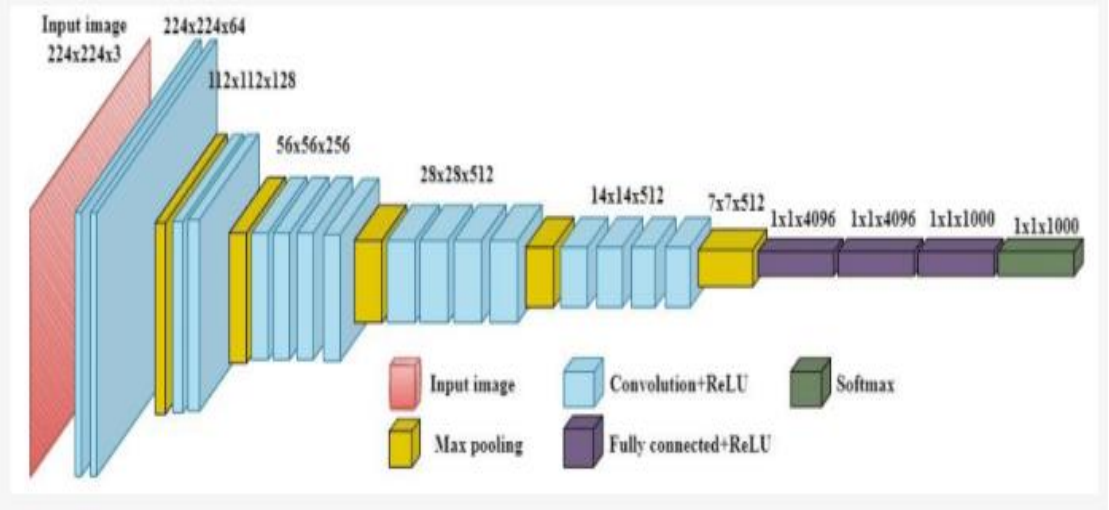

Fig. 8.Layers in VGG19

VGG19, like other CNN architectures, has been successfully applied to various computer vision tasks beyond image classification. Some common applications include object detection, semantic segmentation, and feature extraction. Its deep architecture allows VGG19 to capture rich features, making it particularly useful in tasks that require detailed visual understanding, such as medical image analysis, fine-grained categorization, and plant leaf disease detection.

5. IMPLEMENTATION

5.1 Coding:

Git link: https://github.com/19WH1A05B7/Plant_Leaf_Disease_Detector

5.2 Training Dataset :

The extensive collection of RGB images in the New Plant Diseases Dataset on Kaggle focuses specifically on plant leaves. With a total of 87,900 images, the dataset includes healthy leaves as well as leaves affected by various diseases or disorders.

The dataset encompasses a wide range of crop species such as apple, blueberry, cherry, grape, orange, peach, pepper, potato, raspberry, soy, squash, strawberry, and tomato, totaling 14 different crop species.

Notably, the dataset provides detailed information on disease severity, consisting of 38 distinct classes. These classes include the aforementioned 14 crop species and 26 specific disorders or diseases affecting these plants.

Researchers, practitioners, and enthusiasts interested in plant pathology can benefit greatly from this comprehensive New Plant Diseases Dataset. Its large size and diversity make it an invaluable resource for developing and evaluating advanced machine learning and deep learning models, specifically for the accurate and efficient detection, classification, and diagnosis of plant diseases.

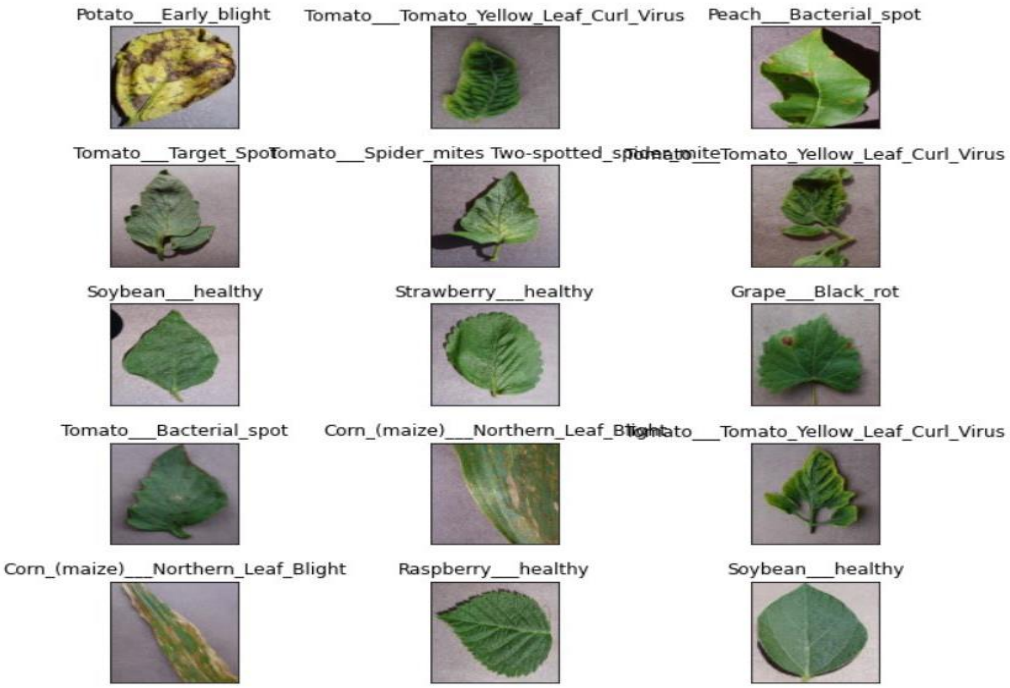

Fig 9: Training Dataset

5.3 Evaluation Metrics:

The effectiveness of our approaches was measured using several evaluation metrics, which are: Accuracy (ACC).

- **Accuracy:**(ACC)

  It is calculated by dividing the number of accurate predictions by the total number of things under consideration.

$$ACC = \frac{TP+TN}{TP+TN+FP+FN}$$

Fig. 10.Training and validation accuracy of CNN Model

Fig. 11.Training and Validation accuracy of VGG19 Model

Table 1. Comparison Table

S.NO	ALGORITHM	TRAINING ACCURACY	TESTING ACCURACY
1	CNN	95.23	91.91
2	VGG19	94.63	93.90

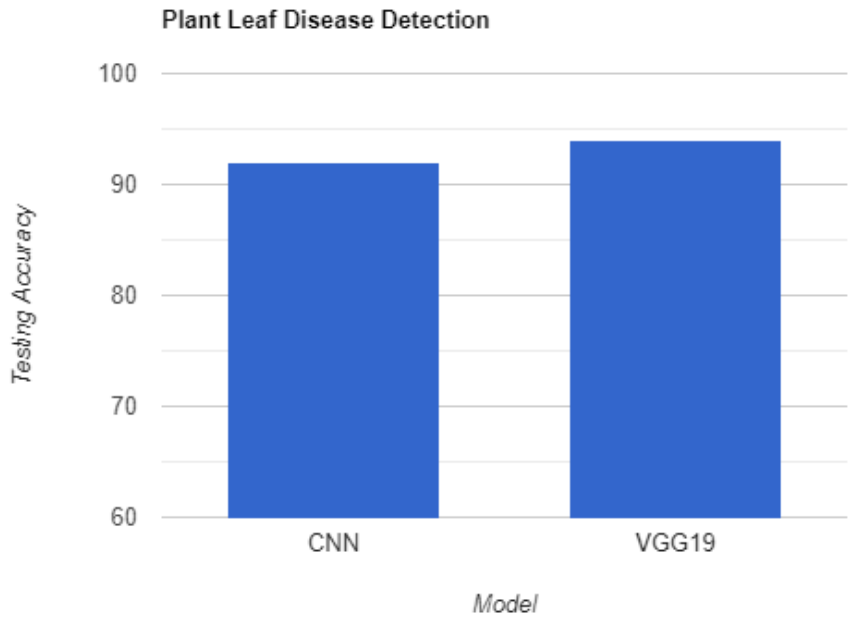

Fig 12. Testing Accuracy

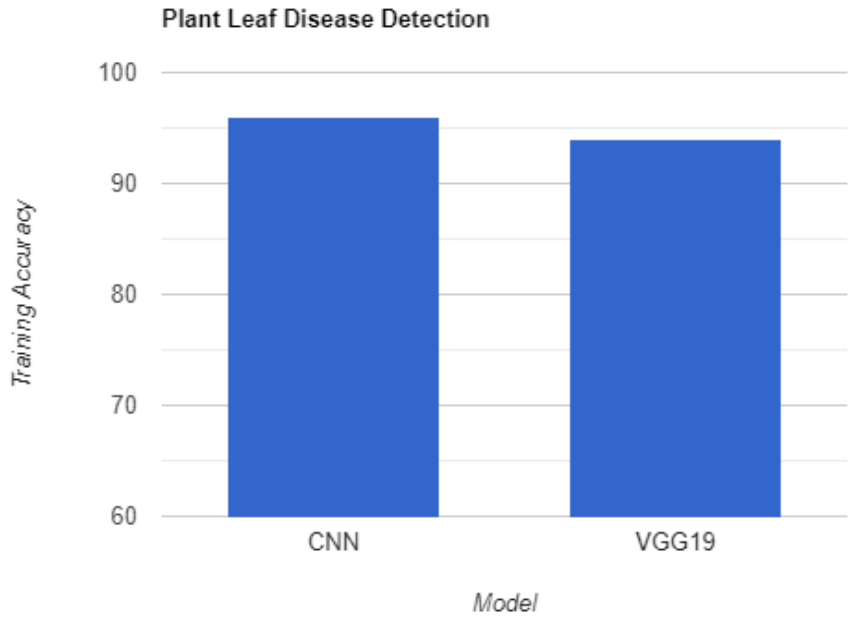

Fig. 13. Training Accuracy

Among the evaluated algorithms, VGG19 demonstrated superior performance in terms of testing accuracy compared to the CNN model. The VGG19 architecture achieved a testing accuracy of 93.90%, outperforming the CNN model's testing accuracy of 91.91%. This suggests that VGG19 has a higher capability to accurately classify plant leaf diseases, resulting in more reliable predictions.

Furthermore, even in terms of training accuracy, VGG19 achieved a commendable accuracy of 94.63%, which is slightly higher than the CNN model's training accuracy of 95.23%. This indicates that VGG19 effectively learned the underlying patterns and features within the training data, enabling it to generalize well and make accurate predictions on unseen test data.

These results highlight the strength of the VGG19 architecture in plant leaf disease detection, as it outperformed the CNN model in terms of both training and testing accuracy. With its deep architecture and ability to capture intricate features, VGG19 showcases its potential to effectively classify and detect various diseases in plant leaves. These findings make a strong case for the utilization of VGG19 as a reliable and accurate model for plant leaf disease detection tasks.

The improved performance of VGG19 can be attributed to its deep architecture and the utilization of smaller-sized filters (3x3) throughout the convolutional layers. The deep architecture allows VGG19 to capture intricate and hierarchical features from the input images, enabling it to better differentiate between various types of plant leaf diseases. The use of smaller-sized filters helps in capturing finer details and localized patterns, contributing to the model's overall effectiveness.

The superior performance of VGG19 in accurately classifying plant leaf diseases makes it a highly promising and reliable model for this task. Its ability to learn and discriminate complex features, as evidenced by the higher testing accuracy, demonstrates its potential for practical applications in the field of agriculture. Researchers and practitioners can confidently leverage the capabilities of VGG19 for plant leaf disease detection, knowing that it has achieved better accuracy compared to the CNN model in this specific evaluation.

The web page provides a user-friendly interface where users can easily upload an image for analysis. Once uploaded, the trained model processes the image to predict the presence of plant diseases. The predicted disease, accompanied by its description and recommended steps, is then displayed to the user. To achieve this functionality, web development frameworks like Flask, along with HTML, CSS, and JavaScript, can be utilized for the front-end implementation.

Dynamic web pages can be employed to effectively communicate the predicted results back to the user. Visualizations, such as displaying the uploaded image alongside the predicted disease, can be implemented using image processing libraries and web technologies. This approach enhances user experience and facilitates the interpretation and understanding of the detection results.

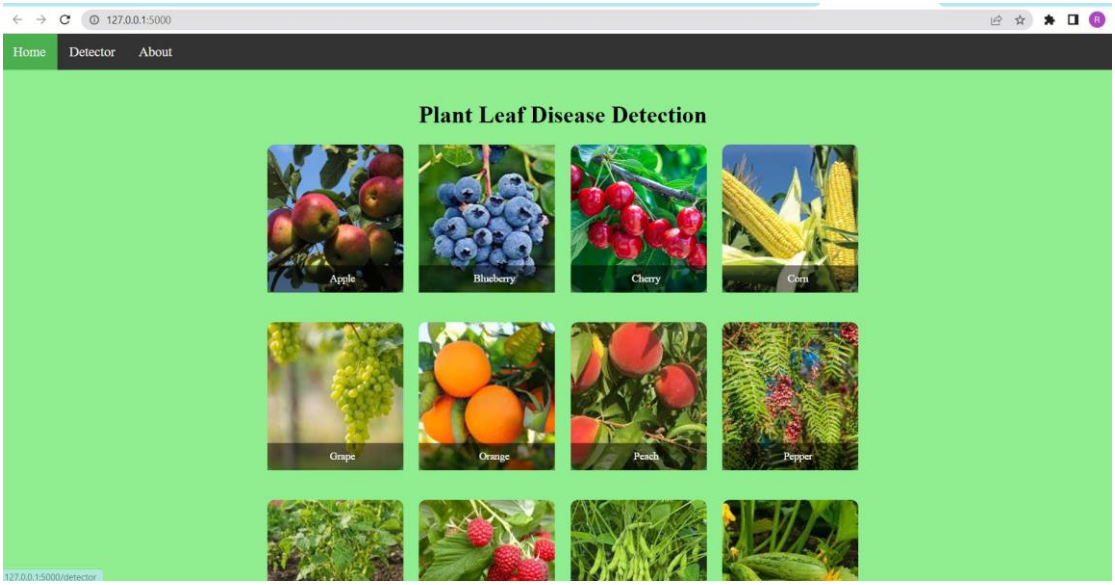

Fig 14: Web page

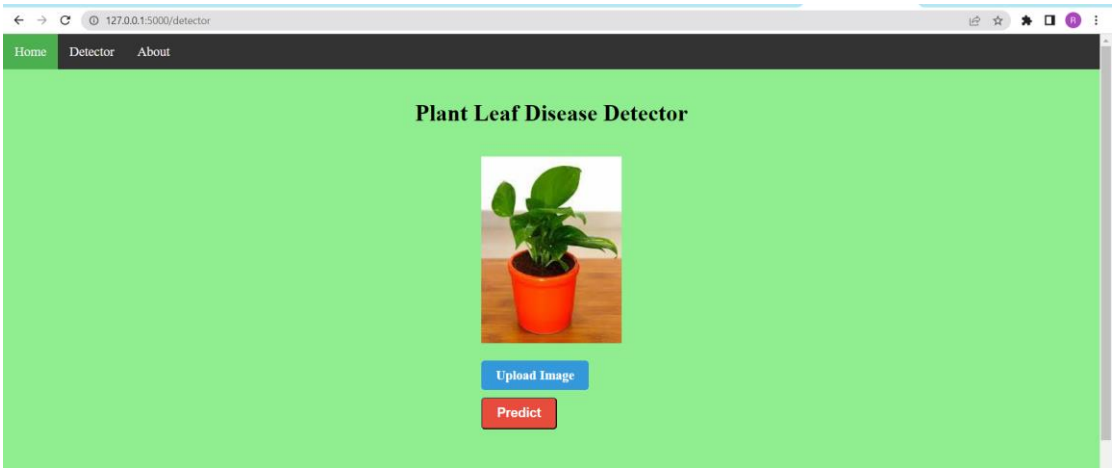

Fig. 15: Detector page

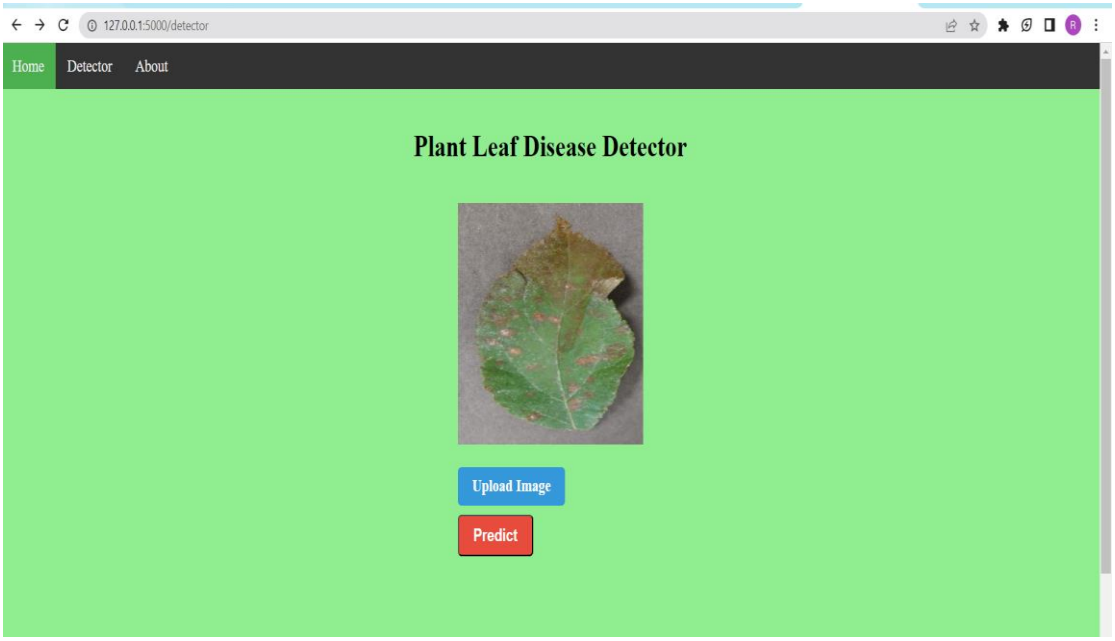

Fig.16 Upload Image

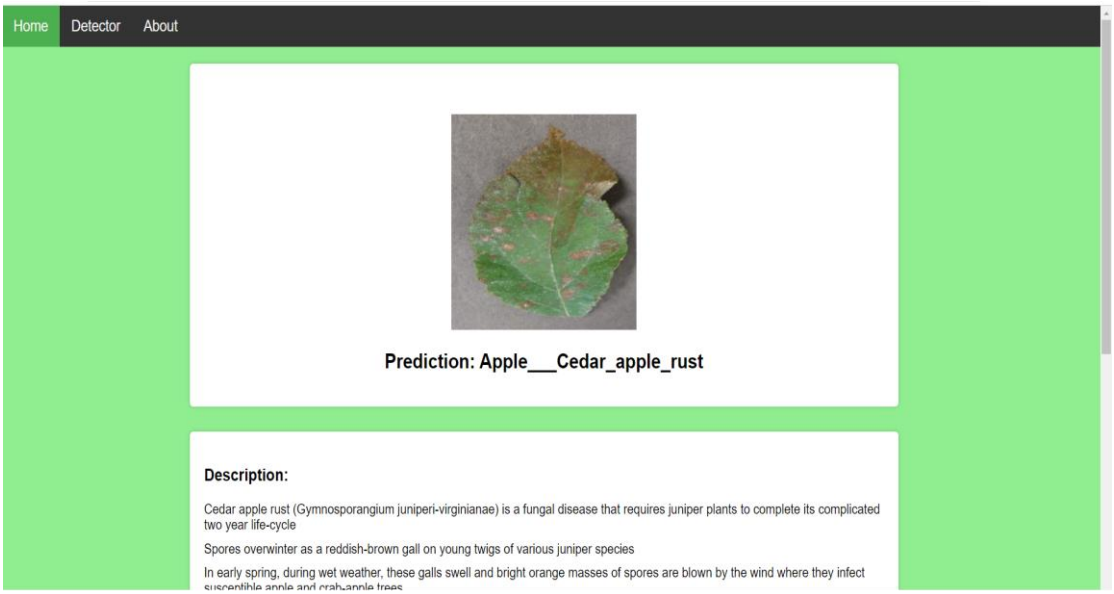

Fig.17 .Disease Description and steps

6. CONCLUSION AND FUTURE SCOPE

The proposed methodology for plant leaf disease detection utilizing deep learning techniques, such as CNN and VGG19, has shown promising results. Through comparative analysis, it was determined that VGG19 achieved superior accuracy in classifying plant diseases compared to the CNN model. The implemented system demonstrated the importance of proper data collection, pre-processing, and feature extraction for accurate disease detection. The user-friendly interface allowed for easy interaction and provided insightful predictions for users.

For future work, there are several potential directions to explore. Firstly, the development of a mobile application or device for scanning plant leaves and capturing images in real time would enhance the system's usability and accessibility. Additionally, integrating probabilistic disease predictions could provide users with a more comprehensive understanding of the likelihood and severity of the detected diseases. Furthermore, incorporating fertilizer recommendation algorithms based on disease analysis and plant health could offer valuable insights for agricultural practices, aiding in the prevention and management of diseases. Finally, expanding the system's capabilities to include a wider range of plant diseases and incorporating continuous learning techniques could ensure the system stays up-to-date with emerging diseases and evolving patterns. These future enhancements would contribute to the practicality and effectiveness of the plant leaf disease detection system, benefiting farmers, researchers, and agricultural communities worldwide.

7. REFERENCES

[1] P. Sharma, P. Hans and S. C. Gupta, "Classification Of Plant Leaf Diseases Using Machine Learning And Image Preprocessing Techniques," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2020, pp. 480-484, doi: 10.1109/Confluence47617.2020.9057889.

[2] P. Chaitanya Reddy, R. M. S. Chandra, P. Vadiraj, M. Ayyappa Reddy, T. R. Mahesh and G. Sindhu Madhuri, "Detection of Plant Leaf-based Diseases Using Machine Learning Approach," 2021 IEEE International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS), Bangalore, India, 2021, pp. 1-4, doi: 10.1109/CSITSS54238.2021.9683020.

[3] D. Varshney, B. Babukhanwala, J. Khan, D. Saxena and A. K. Singh, "Plant Disease Detection Using Machine Learning Techniques," 2022 3rd International Conference for Emerging Technology (INCET), Belgaum, India, 2022, pp. 1-5, doi: 10.1109/INCET54531.2022.9824653.

[4] X. Guan, "A Novel Method of Plant Leaf Disease Detection Based on Deep Learning and Convolutional Neural Network," 2021 6th International Conference on Intelligent Computing and Signal Processing (ICSP), Xi'an, China, 2021, pp. 816-819, doi: 10.1109/ICSP51882.2021.9408806. [5] D. Varshney, B. Babukhanwala, J. Khan, D. Saxena and A. K. Singh, "Plant Disease Detection Using Machine Learning Techniques," 2022 3rd International Conference for Emerging Technology (INCET), Belgaum, India, 2022, pp. 1-5, doi: 10.1109/INCET54531.2022.9824653.

[6] R. M. Alyas and A. S. Mohammed, "Detection of Plant Diseases using Image Processing with Machine Learning," 2022 2nd International Conference on Computing and Machine Intelligence (ICMI), Istanbul, Turkey, 2022, pp. 1-6, doi: 10.1109/ICMI55296.2022.9873793.

[7] R. Moyazzoma, M. A. A. Hossain, M. H. Anuz and A. Sattar, "Transfer Learning Approach for Plant Leaf Disease Detection Using CNN with Pre-Trained Feature Extraction Method

Mobilnetv2," 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), DHAKA, Bangladesh, 2021, pp. 526-529, doi: 10.1109/ICREST51555.2021.9331214.

[8] P. K. V, E. G. Rao, G. Anitha and G. K. Kumar, "Plant Disease Detection using Convolutional Neural Networks," 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2021, pp. 1473-1476, doi: 10.1109/ICOEI51242.2021.9453045.

[9] U. P. Singh, S. S. Chouhan, S. Jain and S. Jain, "Multilayer Convolution Neural Network for the Classification of Mango Leaves Infected by Anthracnose Disease," in IEEE Access, vol. 7, pp. 43721-43729, 2019, doi: 10.1109/ACCESS.2019.2907383.

[10] S. Pratapagiri, R. Gangula, R. G, B. Srinivasulu, B. Sowjanya and L. Thirupathi, "Early Detection of Plant Leaf Disease Using Convolutional Neural Networks," 2021 3rd International Conference on Electronics Representation and Algorithm (ICERA), Yogyakarta, Indonesia, 2021, pp. 77-82, doi: 10.1109/ICERA53111.2021.9538659.

[11] K. Naresh, G. Naga Satish, K.BhargavRam & Ch. Srinivasulu (2019). Machine Learning Based Recognition of Crops Diseases By CNN. In International Journal of Innovative Technology and Exploring Engineering (Vol. 8, Issue 9, pp. 1264–1268).

[12] M. A. Jasim and J. M. AL-Tuwaijari, "Plant Leaf Diseases Detection and Classification Using Image Processing and Deep Learning Techniques," 2020 International Conference on Computer Science and Software Engineering (CSASE), Duhok, Iraq, 2020, pp. 259-265, doi: 10.1109/CSASE48920.2020.9142097.

[13] R. Verma, R. Mishra, P. Gupta, Pooja and S. Trivedi, "CNN based Leaves Disease Detection in Potato Plant," 2023 6th International Conference on Information Systems and Computer Networks (ISCON), Mathura, India, 2023, pp. 1-5, doi: 10.1109/ISCON57294.2023.10112080.

8. APPENDIX

IMPORTING DATASET

```
# Setting up the dataset directory and file paths
traindir = '../project/new plant diseases dataset(augmented)/New Plant Diseases
Dataset(Augmented)/train'
testdir = '../project/test/'
validdir = '../project/new plant diseases dataset(augmented)/New Plant Diseases
Dataset(Augmented)/valid'
```

DATA PREPROCESSING

Image Augumentation

```
img_height = 224
img_width = 224
batch_size = 32

train_datagen = ImageDataGenerator(rescale=1./255,
                          shear_range=0.2,
                          zoom_range=0.2,
                          horizontal_flip=True,
                          fill_mode='nearest')

valid_datagen = ImageDataGenerator(rescale=1./255)
training_set = train_datagen.flow_from_directory(traindir,
                                    target_size=(img_height,img_width),
                                    batch_size=batch_size,
                                    class_mode='categorical')

valid_set = valid_datagen.flow_from_directory(validdir,
                                  target_size=(img_height,img_width),
                                  batch_size=batch_size,
                                  class_mode='categorical')
```

Classes visual representation

```
fig, axes = plt.subplots(nrows=5,
                  ncols=3,
                  figsize=(10,8),
                  subplot_kw={'xticks':[],'yticks':[]})
for i, ax in enumerate(axes.flat):
   ax.imshow(plt.imread(data.Filepath[i]))
   ax.set_title(data.Label[i])
plt.tight_layout()
plt.show()
```

Images count in files

```
counts = data.Label.value_counts()
sns.barplot(x=counts.index, y=counts)
plt.xlabel('Type')
plt.xticks(rotation=90);
```

CNN MODEL

```
# build CNN model
model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(512, activation='relu'))
model.add(Dense(38, activation='softmax'))

# compile model
model.compile(optimizer=Adam(learning_rate=0.001),
        loss='categorical_crossentropy',
        metrics=['accuracy'])
```

VGG19 MODEL

```
# Define the VGG19 model
model = Sequential()

# Convolutional layers
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', input_shape=(224,
224, 3)))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
```

```
model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

model.add(Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(Conv2D(512, (3, 3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))

# Fully connected layers
model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dense(4096, activation='relu'))
model.add(Dense(1000, activation='softmax'))  # Replace 1000 with the number of
classes in your PLDD dataset

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Print the model summary
model.summary()
```

MODEL TRAINING

```
history = model.fit(training_set,
        steps_per_epoch=train_num//batch_size,
        epochs=5,
        validation_data=valid_set,
        validation_steps=valid_num//batch_size)
```

DATA VISUALIZATION

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
```

```
epochs = range(1, len(loss) + 1)

#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()
#loss plot
plt.plot(epochs, loss, color='pink', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

MODEL EVALUATION

```
from keras.models import load_model
model = load_model("../project/vgg19_project.h5")
acc = model.evaluate_generator(valid_set)[1]
print(f"The accuracy = {acc*100}%")
```

MODEL DEPLOYMENT IN FLASK CODE

Detector.py

```
import csv
from flask import Flask, render_template, request
import numpy as np
from tensorflow import keras
from PIL import Image
import json

app = Flask(__name__)

# Load the saved model
model = keras.models.load_model("C:/Users/HP/cnn_model.h5")


@app.route('/')
def home():
   return render_template('home.html')
```

```
def fetch_disease_info(disease):
    with open('disease_info.csv', 'r') as file:
        reader = csv.DictReader(file)
        for row in reader:
            if row['disease_name'] == disease:
                description = row['description'].split('.')
                steps = row['steps'].split('. ')
                return description, steps

    # Return empty strings if the disease is not found in the CSV file
    return '', []

@app.route('/detector', methods=['GET', 'POST'])
def detector():
    if request.method == 'POST':
        if 'image' not in request.files:
            return render_template('detector.html', error='No image file uploaded.')

        # Get the uploaded image file from the request
        image_file = request.files['image']

        # Check the file extension
        if image_file.filename == '':
            return render_template('detector.html', error='No image file selected.')

        if not allowed_file(image_file.filename):
            return render_template('detector.html', error='Invalid file extension. Only
JPG, JPEG, and PNG files are allowed.')

        imagepath = "static/uploaded_image.jpg"  # Set the path where you want to save
the image
        image_file.save(imagepath)

        # Check if the uploaded image is a leaf image
        # if not check_leaf_image(imagepath):
        #     return render_template('detector.html', error='Uploaded image does not
contain predominantly leaves.')

        # Load and preprocess the image
        image = Image.open(imagepath)
        image = image.resize((224, 224))  # Adjust the image size as needed
        image = np.array(image) / 255.0  # Normalize the image pixel values

        # Reshape the image array to match the model's input shape
        image = np.expand_dims(image, axis=0)

        # Perform the prediction using the loaded model
        pred = np.argmax(model.predict(image))

        # Process the prediction (if required)
```

```
        # For example, convert the prediction to a human-readable format
        plant_diseases = ['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']
        disease = plant_diseases[pred]
        description, steps = fetch_disease_info(disease)

        return render_template('result.html', image_path=imagepath,
prediction=disease,description=description, steps=steps)

    return render_template('detector.html')


@app.route('/about')
def about():
    author_details = {
        'name': 'John Doe',
        'email': 'johndoe@example.com',
        'bio': 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam sodales
lacus ac turpis iaculis sodales. Nulla facilisi. In hac habitasse platea dictumst.'
    }
    return render_template('about.html', author_details=author_details)


# def fetch_suggestions(disease):
#     # Read suggestions from JSON file
```

```
#     with open('suggestions.json') as file:
#         suggestions_data = json.load(file)

#     # Get suggestions for the selected disease
#     suggestions = suggestions_data.get(disease, [])

#     return suggestions



def check_leaf_image(image_file):
    # Load and preprocess the image
    image = Image.open(image_file)
    image = image.convert("RGB")
    image = image.resize((224, 224))  # Adjust the image size as needed
    image = np.array(image) / 255.0  # Normalize the image pixel values

    # Calculate the percentage of green pixels in the image
    green_pixels = np.sum(image[:, :, 1])  # Green channel
    total_pixels = image.shape[0] * image.shape[1]
    green_percentage = green_pixels / total_pixels

    # Set a threshold for green pixel percentage
    threshold = 0.5  # Adjust the threshold as needed

    # Return True if the image contains predominantly leaves (green pixels above the
threshold)
    return green_percentage > threshold


ALLOWED_EXTENSIONS = {'jpg', 'jpeg', 'png'}

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


if __name__ == '__main__':
    app.run(debug=True)
```