An Efficient Deep Learning-Based System for Accurate Plant Leaf Disease Detection and Remedial Measures

A PROJECT REPORT
Submitted by

PRADEEP KUMAR S	**(811719104064)**
SHRI VISHNU P	**(811719104092)**
SOMASUNDARAM S	**(811719104095)**

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY
(An Autonomous Institution, affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621112

MAY-2023

BONAFIDE CERTIFICATE

Certified that this project report titled **"AN EFFICIENT DEEP LEARNING-BASED SYSTEM FOR ACCURATE PLANT LEAF DISEASE DETECTION AND REMEDIAL MEASURES"** is the bonafide work of **PRADEEP KUMAR S (811719104064), SHRI VISHNU P (811719104092) ,** and **SOMASUNDARAM S (811719104095)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

-

SIGNATURE

Mr. R. Rajavarman M.E., (PhD)

HEAD OF THE DEPARTMENT

Department of Computer Science and Engineering

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram – 621 112

SIGNATURE

Mrs. K. Vallipriadharshini M.E., (PhD)

SUPERVISOR

(ASSISTANT PROFESSOR)

Department of Computer Science and Engineering

K.Ramakrishnan College of Technology (Autonomous)

Samayapuram – 621 112

Submitted for the viva-voice examination held on ……

Internal Examiner **External Examiner**

DECLARATION

We jointly declare that the project report on **"AN EFFICIENT DEEP LEARNING-BASED SYSTEM FOR ACCURATE PLANT LEAF DISEASE DETECTION AND REMEDIAL MEASURES"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of Bachelor of Engineering in Computer Science and Engineering. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of Bachelor of Engineering.

Signature

PRADEEP KUMAR S (811719104064)

SHRI VISHNU P (811719104092)

SOMASUNDARAM S (811719104095)

Place: Samayapuram

Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K. Ramakrishnan College of Technology**", for providing us with the opportunity to do this project.

We are glad to credit honorable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

We would like to thank **Dr. N. Vasudevan, M.E., Ph.D.,** Dean, who gave opportunity to frame the project the full satisfaction.

We whole heartily thanks to **Mr. R. Rajavarman M.E.**, **(PhD)** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing his encourage pursuing this project.

We express my deep and sincere gratitude to my project guide **Mrs. K. Vallipriadharshini M.E.**, **(PhD)** Assistant professor of **COMPUTER SCIENCE AND ENGINEERING** Department, for her incalculable suggestions, creativity, assistance and patience which motivated me to carry out this project.

We render my sincere thanks to Course Coordinator and other staff members for providing valuable information during the course. We wish to express my special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

Plant diseases can significantly reduce agricultural output and result in severe financial losses. For disease control to be successful, plant illnesses must be accurately and quickly detected. In this project, we provide a deep learning-based method for identifying plant illnesses and offer cures for those diseases that are found. Convolutional neural networks (CNNs) are used in our suggested method to classify plant diseases and to suggest appropriate treatments depending on the findings of the diagnostic. We collected a dataset of plant images, consisting of healthy leaves and leaves with five common types of diseases, and pre-processed the dataset to train our deep learning models. Using a variety of assessment measures, we assessed the efficacy of our suggested strategy and contrasted it with more established techniques for identifying plant diseases. In this study, we used a publically accessible dataset of 2152 photos of healthy and sick tomato leaves gathered under controlled settings to train a deep convolutional neural network to recognise 2 illnesses. The trained model's outstanding accuracy of 99.91% on a held-out test set proved the viability and promise of this method for widespread, worldwide smartphone-assisted crop disease diagnosis. The proposed approach can also provide remedial measures for the detected diseases, including spraying of fungicides, proper irrigation, and pruning, among others. Our work demonstrates the potential of deep learning methods in the field of managing and detecting plant diseases and offers a viable option for efficient and timely disease diagnosis.

Keywords – Convolutional neural network, plant diseases, deep learning, Image processing, Categorization of leaves.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Plant illnesses have the potential to severely lower agricultural output and quality, which would result in substantial financial losses for farmers and food shortages for populations. Plant diseases can be contained and the damage reduced with prompt identification and diagnosis. Visual examination by farmers or agricultural professionals is a traditional technique of disease identification, although it can be time-consuming, subjective, and prone to mistakes. In recent years, there has been an increase in interest in creating automated systems for plant disease identification and diagnosis utilizing deep learning techniques.

Convolutional neural networks (CNNs), a type of deep learning model, have demonstrated encouraging results in effectively identifying and diagnosing plant illnesses from photographs of afflicted leaves. These models may be taught to recognize patterns and characteristics linked to particular illnesses by subjecting them to extensive datasets of labelled pictures during training. Once a disease is detected, remedial measures can be recommended to farmers based on the type and severity of the disease.

In this project, we investigate the application of deep learning methods for plant leaf disease diagnosis and offer corrective actions. We offer a method for identifying and categorizing typical plant diseases from leaf photos that is CNN-based. We also propose a system that recommends appropriate remedial measures for each detected disease based on expert knowledge and existing scientific literature. Our suggested approach might enable farmers take quick and efficient action to stop the spread of illnesses and reduce crop losses by dramatically increasing the efficiency and accuracy of plant disease diagnostics.

1.1 INTRODUCTION TO DEEP LEARNING

First Generation of convolutional neural network (CNN) was composed of perceptron's in neural layers, which were limited in computations. The second-generation calculated the error rate and backpropagated the error. Restricted Boltzmann machine overcame the limitation of backpropagation, which made the learning easier. Then other networks are evolved eventually. The performance of classifiers using deep learning improves on a large scale with an increased quantity of data when compared to traditional learning methods. The performance of traditional machine learning algorithms becomes stable when it reaches the threshold of training data whereas the deep learning upturns it's performance with increased amount of data. Now a day's deep learning is used in a lot many applications such as Google's voice and image recognition, Netflix and Amazon's recommendation engines, Apple's Siri, automatic email and text replies, chatbots etc.

Deep learning is a type of machine learning and artificial intelligence (AI) that imitates the way humans gain certain types of knowledge. Deep learning is an important element of data science, which includes statistics and predictive modeling. It is extremely beneficial to data scientists who are tasked with collecting, analyzing and interpreting large amounts of data; deep learning makes this process faster and easier. At its simplest, deep learning can be thought of as a way to automate predictive analytics. While traditional machine learning algorithms are linear, deep learning algorithms are stacked in a hierarchy of increasing complexity and abstraction.

A type of advanced machine learning algorithm, known as an artificial neural network, underpins most deep learning models. As a result, deep learning may sometimes be referred to as deep neural learning or deep neural networking. Neural networks come in several different forms, including

convolutional neural networks, artificial neural networks and feedforward neural networks, and each has benefits for specific use cases. However, they all function in somewhat similar ways by feeding data in and letting the model figure out for itself whether it has made the right interpretation or decision about a given data element.

Neural networks involve a trial-and-error process, so they need massive amounts of data on which to train. It's no coincidence neural networks became popular only after most enterprises embraced big data analytics and accumulated large stores of data. Because the model's first few iterations involve somewhat educated guesses on the contents of an image or parts of speech, the data used during the training stage must be labelled so the model can see if its guess was accurate. This means, though many enterprises that use big data have large amounts of data, unstructured data is less helpful. Unstructured data can only be analyzed by a deep learning model once it has been trained and reaches an acceptable level of accuracy, but deep learning models can't train on unstructured data.

1.2 Types of Deep Learning

There are several types of deep learning architectures used in the field of artificial intelligence. Some of the most commonly used types are:

1. Convolutional Neural Networks (CNNs): CNNs are primarily used for image and video recognition. They work by using multiple layers of convolutional filters to extract features from the input data, which are then used to classify the data.

2. Recurrent Neural Networks (RNNs): RNNs are used for processing sequential data such as text, speech, and time series data. They work by using recurrent connections to maintain a memory of the past inputs, which enables them to better understand the context of the current input.

3. Generative Adversarial Networks (GANs): GANs are used for generating new data that is similar to the input data. They work by training two neural networks: a generator network that generates new data, and a discriminator network that evaluates the authenticity of the generated data.

4. Autoencoders: Autoencoders are used for data compression and feature extraction. They work by encoding the input data into a compressed representation, which is then decoded back into the original data. The compressed representation can be used for further analysis or classification.

5. Reinforcement Learning: Reinforcement learning is used for training agents to learn from their environment. It works by using a reward system to incentivize the agent to learn optimal actions based on its current state.

Each of these deep learning architectures has its own strengths and weaknesses, and they are often used in combination to achieve more complex tasks.

1.3 Advantages of Deep Learning

Deep learning is a subset of machine learning that utilizes artificial neural networks to model and solve complex problems. Here are some of the advantages of deep learning:

1. High Accuracy: One of the primary advantages of deep learning is its high accuracy. Neural networks can be trained to recognize patterns and features in large datasets, which makes them ideal for tasks such as image and speech recognition, natural language processing, and fraud detection.

2. Automated Feature Extraction: In traditional machine learning algorithms, feature extraction is a manual and time-consuming process. However, deep learning algorithms can automatically extract features from raw data, which saves a significant amount of time and effort.

3. Scalability: Deep learning models can handle large and complex datasets, making them suitable for big data applications. Neural networks can be trained on clusters of computers, which makes it possible to scale up the training process and reduce the training time.

4. Versatility: Deep learning algorithms can be used in a wide range of applications, from image and speech recognition to language translation and self-driving cars. As a result, deep learning has the potential to revolutionize many industries and drive innovation in various fields.

5. Adaptability: Deep learning models can adapt and learn from new data, making them ideal for tasks where the data distribution changes over time. This makes deep learning particularly useful in applications such as stock market prediction, weather forecasting, and customer behavior analysis.

Overall, the advantages of deep learning make it a powerful tool for solving complex problems in a wide range of applications. As deep learning algorithms continue to evolve and improve, we can expect to see even more innovative and exciting applications of this technology in the future.

1.4 Challenges of Deep Learning

Deep learning, despite its many advantages, also poses several challenges. Some of the most significant challenges of deep learning include:

1. Data requirements: Deep learning algorithms require a massive amount of training data to perform well. This can be a challenge in many real-world applications where data may be scarce, or the collection and annotation of data may be expensive and time-consuming.

2. Computational resources: Deep learning models are computationally intensive and require a significant amount of processing power and memory. This can be a challenge for organizations that do not have access to specialized

hardware or cloud computing resources.

3. Interpretability: Deep learning models are often black boxes, making it difficult to understand how they arrive at their predictions. This lack of interpretability can be a significant challenge in many domains where the ability to explain and understand model predictions is essential.

4. Overfitting: Deep learning models can be prone to overfitting, which occurs when a model is too complex and learns to memorize the training data rather than generalize to new data. Overfitting can result in poor performance on new data and can be challenging to detect and prevent.

5. Algorithmic biases: Deep learning models can also be prone to algorithmic biases, which can lead to discriminatory or unfair outcomes. This can be a significant challenge in many domains, including healthcare and criminal justice, where decisions based on machine learning models can have real-world consequences.

Despite these challenges, ongoing research and development in the field of deep learning are addressing many of these issues, and the potential benefits of deep learning continue to drive significant interest and investment in the field.

1.5 Application of Deep Learning

Deep learning has found a wide range of applications in various fields, including computer vision, speech recognition, natural language processing, and robotics. Some of the notable applications of deep learning are:

1. Computer Vision: Deep learning is widely used in computer vision applications such as object detection, image classification, and segmentation. It has been used to develop systems that can identify objects in images, recognize faces, and even drive autonomous vehicles.

2. Speech Recognition: Deep learning has been applied to improve the accuracy of speech recognition systems. It has been used to develop voice assistants such as Siri and Alexa, which can recognize and respond to voice commands.

3. Natural Language Processing: Deep learning has been used to improve the accuracy of natural language processing systems. It has been used to develop chatbots, which can interact with humans in a natural language.

4. Robotics: Deep learning is used to develop intelligent robots that can learn from their environment and adapt to new situations. It has been used to develop robots that can perform tasks such as object recognition, grasping, and manipulation.

Overall, deep learning has the potential to revolutionize many fields by providing powerful tools for data analysis, pattern recognition, and decision-making.

CHAPTER 2

LITRATURE SURVEY

2.1. TITLE: A Systematic Analysis of Machine Learning and Deep Learning Based Approaches for Plant Leaf Disease Classification

AUTHOR: Raj Kumar, Anuradha Chug, Amit Prakash Singh and Dinesh Singh

YEAR: 2022

DESCRIPTION:

Crops' production and quality of yields are heavily affected by crop diseases which cause adverse impacts on food security as well as economic losses. In India, agriculture is a prime source of income in most rural areas. In this study, we systematically reviewed recent research studies undertaken by a variety of scholars and researchers of fungal and bacterial plant disease detection and classification and summarized them based on vital parameters like type of crop utilized, deep learning/machine learning architecture used, dataset utilized for experiments, performance matrices, types of disease detected and classified, and highest accuracy achieved by the model. As per the analysis carried out, in the category of machine learning-based approaches, 70% of studies utilized real-field plant leaf images and 30% utilized laboratory condition plant leaf images for disease classification while in the case of deep learning-based approaches, 55% studied employed laboratory-conditioned images from the Plant Village dataset, 25% utilized real-field images, and 20% utilized open image datasets. The average accuracy attained with deep learning-based approaches is quite higher at 98.8% as compared to machine learning-based approaches at 92.2%. In the case of deep learning-based methods, we also analyzed the performances of pretrained and training from scratch models that have been utilized in various studies for plant leaf disease classification. Pretrained models perform better with 99.64% classification accuracy compared to training from scratch models which achieved 98.64% average accuracy.

2.2. TITLE: IoT and Deep Learning based Smart Greenhouse Disease Prediction

AUTHOR: Karunakar Pothuganti, Balne Sridevi, and Phaneendra Seshabattar

YEAR: 2021

DESCRIPTION:

Globally, rapid industrialization and urbanization have resulted in a reduction in agricultural acreage and production. As a result of this, and the growing desire for chemical-free organic veggies among educated urban families, greenhouses are rapidly gaining popularity for their specific benefits, particularly in regions with severe weather. They provide the perfect conditions for longer and more productive growth seasons, as well as lucrative harvests. The current article proposes and shows a full IoT -based Smart Greenhouse system that combines monitoring, alerting, cloud storage, automation, and disease prediction into a single, easily deployed package. It constantly monitors environmental variables like as temperature, humidity, and soil moisture to guarantee a better crop production and quick correction in the event of aberrant circumstances. A built-in automated irrigation management system is also included. Finally, for disease detection using leaf pictures, it uses the most efficient deep learning model. Furthermore, using cloud storage to optimise memory and storage, a city dweller may construct a greenhouse and watch it from his home, allowing him to take corrective action as needed.

2.3. TITLE: A real-time approach of diagnosing rice leaf disease using deep learning-based faster R-CNN framework

AUTHOR: Bifta Sama Bari, Md Nahidul Islam, and Mamunur Rashid

YEAR: 2021

DESCRIPTION:

The rice leaves related diseases often pose threats to the sustainable production of rice affecting many farmers around the world. Early diagnosis and appropriate remedy of the rice leaf infection is crucial in facilitating healthy growth of the rice plants to ensure adequate supply and food security to the rapidly increasing population. Therefore, machine-driven disease diagnosis systems could mitigate the limitations of the conventional methods for leaf disease diagnosis techniques that is often time-consuming, inaccurate, and expensive. Nowadays, computer-assisted rice leaf disease diagnosis systems are becoming very popular. However, several limitations ranging from strong image backgrounds, vague symptoms' edge, dissimilarity in the image capturing weather, lack of real field rice leaf image data, variation in symptoms from the same infection, multiple infections producing similar symptoms, and lack of efficient real-time system mar the efficacy of the system and its usage. To mitigate the aforesaid problems, a faster region-based convolutional neural network (Faster R-CNN) was employed for the real-time detection of rice leaf diseases in the present research. The Faster R-CNN algorithm introduces advanced RPN architecture that addresses the object location very precisely to generate candidate regions. The robustness of the Faster R-CNN model is enhanced by training the model with publicly available online and own real-field rice leaf datasets. The proposed deep-learning-based approach was observed to be effective in the automatic diagnosis of three discriminative rice leaf diseases including rice blast, brown spot, and hispa with an accuracy of 98.09%, 98.85%, and 99.17% respectively. Moreover, the model was able to identify a healthy rice leaf with an accuracy of 99.25%.

2.4. TITLE: Deep Learning-Based Image Processing for Cotton Leaf Disease and Pest Diagnosis

AUTHOR: Azath M, Melese Zekiwos, and Abey Bruck

YEAR: 2021

DESCRIPTION:

Cotton is one of the economically significant agricultural products in Ethiopia, but it is exposed to different constraints in the leaf area. Mostly, these constraints are identified as diseases and pests that are hard to detect with bare eyes. This study focused to develop a model to boost the detection of cotton leaf disease and pests using the deep learning technique, CNN. To do so, the researchers have used common cotton leaf disease and pests such as bacterial blight, spider mite, and leaf miner. K-fold cross-validation strategy was worn to dataset splitting and boosted generalization of the CNN model. For this research, nearly 2400 specimens (600 images in each class) were accessed for training purposes. This developed model is implemented using python version 3.7.3 and the model is equipped on the deep learning package called Keras, TensorFlow backed, and Jupyter which are used as the developmental environment. This model achieved an accuracy of 96.4% for identifying classes of leaf disease and pests in cotton plants. This revealed the feasibility of its usage in real-time applications and the potential need for IT-based solutions to support traditional or manual disease and pest's identification.

2.5. TITLE: Binary class and multi-class plant disease detection using ensemble deep learning-based approach

AUTHOR: C.K. Sunil, C.D. Jaidhar and Nagamma Patil

YEAR: 2022

DESCRIPTION:

Providing food for the exponentially growing global population is a highly challenging task. Owing to the demand and supply gap may diminish food production due to diseases in plants, such as bacterial disease, viral disease, and fungal diseases. Early recognition of such diseases and applying an appropriate pesticide or fertiliser can improve crop yield. Accordingly, early plant disease detection necessitates continuous crop monitoring from its initial stages. Recently some research works have been proposed as remedial measures. However, such methodologies utilise costly equipment that is infeasible for small-scale farmers. Thus, there is a need for a cost-effective plant-disease-detection approach. This study embellishes the challenges and opportunities in plant disease detection. Correspondingly, this research proposes an ensemble deep learning-based plant disease diagnosis approach using a combination of AlexNet, ResNet50, and VGG16 deep learning-based models. It effectively ascertains plant diseases by analysing the plant leaf images. A broad set of experiments were conducted using different plant leaf image datasets such as cherry, grape, maize, pepper, potato, strawberry, and cardamom to evaluate the robustness of the proposed approach. Experiential results demonstrated that the proposed approach attained a maximum detection accuracy of 100% for binary and 99.53% for multi-class datasets.

2.6. TITLE: Plant Disease Identification and Suggestion of Remedial Measures using Machine Learning

AUTHOR: Shyam Chand G, Hari R

YEAR: 2021

DESCRIPTION:

Plants are an important source of energy for all organisms on earth. But plant diseases act as a hindrance for effective consumption of plant products and also adversely affect the life of crops. When the farmers diagnose diseases manually, lot of difficulties arise due of the lack of knowledge and unavailability of professionals. It also requires much time in manually identifying and classifying crop diseases. In this context, a model is proposed for identifying plant diseases and to suggest remedial measures. Here a transfer learning-based CNN model is implemented using VGG16 and ResNet50. The dataset used consists of 34824 training images and 8767 testing images of thirty-eight output classifications including 26 crop diseases found in fourteen crops. The VGG16 model shown 99.1 percentage accuracy and ResNet50 exhibited 99.3 percentage accuracy with considerable reduction of computation time than VGG16.

2.7. TITLE: Machine Learning Techniques for Plant Disease Detection

AUTHOR: Divyanshu Varshney, Burhanuddin Babukhanwala, and Javed Khan

YEAR: 2022

DESCRIPTION:

Undoubtedly, agriculture is an essential source of livelihood, which stands as a backbone of Indian economy. The plant production is severely affected due to various kinds of diseases, which if accurately and timely detected, could raise health standards and economic growth significantly. The traditional approaches of disease detection and classification involves an immense amount of time, an intense amount of labor and constant monitoring of the farm. By using disease detection methods, diseases caused by bacteria, viruses and fungi are often avoided. Within the upkeep of agricultural goods, crop protection plays a critical role. Techniques of Machine Learning are often used to identify the affected leaf images. The various machine learning algorithms used to determine whether a plant is infected or not with a disease, are discussed in this study. It was done in various steps, such as image acquisition, feature extraction, categorization of the illness and result display. This paper also needs to carry out an accurate study of various techniques for the identification of diseases of plants. The aim is to identify the plant diseases using image analysis. It also, after detection of the illness, says the name of fertilizer to be used. The pests and insects accountable for the pandemic are also described.

2.8. TITLE: Real-time detection of maize crop disease via a deep learning-based smartphone app

AUTHOR: Blake Richey, Sharmin Majumder, and Mukul Shirvaikar.

YEAR: 2020

DESCRIPTION:

Diseases in plants are substantially problematic for agricultural yield management. Compounded with insufficient information to correctly diagnose crops, they can lead to significant economic loss and yield inefficiencies. Due to the success of deep learning in many image processing applications, the first part of this paper involves designing a deep neural network for the detection of disease in maize due to its economic significance. A convolutional neural network is designed and trained on a public domain dataset with labeled images of maize plant leaves with disease present, or lack thereof. In the second part of this paper, the trained convolutional neural network is turned into a smartphone app running in real-time for the purpose of performing maize crop disease detection in the field in an on-the-fly manner. The smartphone app offers a cost-effective, portable, and universally accessible way to detect disease in maize. The approach developed in this paper enables recognizing early signs of plant pathogens from maize crop images in realtime in the field, thus leading to preemptive corrective actions prior to significant yield loss. Keywords: Artificial Intelligence in agriculture, real-time detection of crop disease, smartphone

2.9. TITLE: An IoT and Machine Learning Based Intelligent System for the Classification of Therapeutic Plants

AUTHOR: Roopashree Shailendra, Anitha Jayapalan, and Sathiyamoorthi Velayutham

YEAR: 2022

DESCRIPTION:

The work aim to develop an automatic recognition model using IoT and machine learning (ML) techniques for the classification of therapeutic plants to enrich the traditional medicinal system. Ayurveda, the oldest Indian system of medicine is still practiced today as it widely promotes herbs as medicines for treating different health conditions and presents many advantages, such as low cost, availability in abundance, and minimal side effects. While many countries have accepted conventional medicine as the best alternative to synthetic drugs, the lack of knowledge and unsupported evidence has raised concerns and reduced its usage. Herein, an intelligent system is proposed using Raspberry Pi 3 Model B + (RPi) and the RPi camera to identify real-time images of Indian medicinal herbs and reveal their respective medicinal properties. Four ML models are developed in the work, of which one model is proposed to identify the details of a captured medicinal leaf on the RPi user interface. The proposed model predicts an accuracy (top-1) of 98.98% on a custom leaf dataset of 25 different medicinal species, containing 1500 leaf images, by combining two feature extraction techniques, namely scale invariant feature transform (SIFT) and histogram of oriented gradients. Bag of Visual Words is obtained by applying k-means clustering on SIFT descriptors as a feature selection and assessed using a support vector machine classifier. The suggested model integrated into RPi shows a real-time top-3 accuracy of 99%. The designed system has the advantages of being built solely for medicinal herbs, with reduced camera cost, and even works efficiently in remote areas.

2.10. TITLE: Deep learning based automated disease detection and pest classification in Indian mung bean

AUTHOR: MD Tausif Mallick, Shrijeet Biswas, and Amit Kumar Das

YEAR: 2023

DESCRIPTION:

Crop pests and diseases are major threats to food security globally. The mung bean (Vigna Radiata) is one of the leading crops in India. A large part of the population in India is completely dependent on mung bean. So, high production efficiency for the mung bean is required, which does not happen due to the excessive damage from pests and diseases. Recently, with the advancement of Deep Learning techniques, remarkable performance has been achieved in the field of image classification by employing Convolutional Neural Networks (CNNs). This brings a lot of promise in the field of pest and disease identification by effective image classification. In this paper, we have proposed a novel deep learning-based technique to identify the mung bean pest and disease. In order to handle the limitation arising due to less number of mung bean crop images for the purpose of training, we have adopted transfer learning, which is able to generate a very promising result for quick and easy pest and disease detection. The developed model has successfully recognized 6 different types of mung bean diseases and 4 types of pests out of healthy and affected leaves collected in different seasons. Based on the experiments conducted, the proposed smartphone-based deep learning model for the mung bean pest and disease detection has achieved an average accuracy of 93.65%.

CHAPTER 3

SYSTEM ANALYSIS

3.1 OBJECTIVE

The main goal of this project is to aims to address the problem of reducing crop yield due to plant diseases by developing a deep learning-based system for accurate disease detection and remedial measures. The proposed system utilizes digital images of plant leaves to identify various plant diseases and suggest appropriate remedial measures. With the increasing availability of digital cameras and smartphones, the proposed system can easily be used by farmers and agricultural workers, making it a low-cost and accessible solution for identifying and managing plant diseases. The system employs deep learning techniques such as convolutional neural networks (CNNs) and transfer learning to accurately classify the diseases based on the image features of plant leaves. The proposed system can also provide remedial measures to prevent further spread of the disease, thereby helping farmers to take timely and effective action.

3.2 EXISTING SYSTEM

In existing system, The Trained model was deployed on Computer Machine with Higher Hardware requirements. User have to feed the image into the Model by using the Python File and the important thing in this model is, user must have installed the dependencies like Tensor flow, etc., in their system. By using this, User can predict the disease of the plant. It doesn’t provide the details and remedy measures of the disease of the plant being detected.

3.2.1 Drawbacks Identified

- Requires a computer machine with higher hardware requirements for deploying the trained model, which may limit its accessibility.
- The user is required to manually feed the image into the model using a Python file, which can be a complex and time-consuming process for those who are not familiar with the programming language.
- The system requires the user to install dependencies such as TensorFlow, which may not be feasible for all users.
- The existing system only provides disease detection and does not provide detailed information or remedial measures for the detected disease, which can limit its effectiveness in preventing further spread of the disease.

3.3 PROPOSED SYSTEM

The proposed system for plant leaf disease detection and remedial measures is an automated deep learning-based approach that can accurately detect and diagnose plant diseases using image analysis. The system consists of a convolutional neural network (CNN) model that is trained on a large dataset of plant images to identify the presence of disease and provide detailed information and remedial measures to prevent further spread of the disease.

The proposed system overcomes the limitations of the existing system by providing a user-friendly interface that can be accessed through a web application. The user can simply upload the image of the diseased plant onto the web interface, and the system will automatically detect the disease, providing detailed information about the disease and the remedial measures to prevent further spread of the disease.

3.3.1 Proposed Solution

- Automated deep learning-based approach that accurately detects and diagnoses plant diseases using image analysis.
- User-friendly web application interface that can be accessed remotely, making it easily accessible to farmers and agricultural workers in remote areas.
- Provides detailed information and remedial measures to prevent further spread of the disease, enhancing the effectiveness of disease management.
- Increases the accuracy and efficiency of plant disease management, leading to improved crop quality and reduced environmental impact.

CHAPTER 4
SYSTEM REQUIREMENTS

4.1 HARDWARE REQUIREMENTS

Processor: Intel Core i5 or higher

RAM: 8 GB or higher

Hard disk: 256 GB or higher

4.2 SOFTWARE REQUIREMENTS

Operating system : Windows 10 or Ubuntu 20.04 LTS

Front End : ReactJS

Back-end : Python Flask

IDE : Visual Studio Code or Jupyter Notebook

Dataset : UCI (University of California Irvine)

CHAPTER 5
METHODOLOGIES

5.1 PLANT LEAF PREDICTION

Convolutional neural networks (CNNs) were employed in this study to identify plant leaf disease and implement corrective procedures. The proposed methodology consists of three main steps: dataset preparation, model development, and model evaluation.

5.1.1 Dataset Preparation

'Late blight' and 'Early blight' are two prevalent plant leaf diseases that are depicted in 2152 photos in the dataset utilized in this investigation. The images were collected from various sources and pre-processed to remove any unwanted information such as background noise and unrelated objects. After then, the dataset was split into two subsets: a training set that contained 70% of the photos and a testing set that contained the remaining 30%. To prevent overfitting and assess the model's performance on omitted data, the dataset is split into training and testing sets.

5.1.2 Model Development

A CNN architecture, a kind of deep neural network made to identify patterns in pictures, was used to create the suggested model. Convolutional layers, pooling layers, and fully linked layers are just a few of the many layers that make up the model. While the pooling layers shrink the spatial size of the output features, the convolutional layers learn the characteristics of the input pictures. Using completely linked layers, the output features are divided into several classes. Using a categorical cross-entropy loss function and a stochastic gradient descent optimizer, the model was trained on the training set. A batch size of 32 was used during the model's 50 epoch training period.

5.1.3 Model Evaluation

Accuracy, precision, recall, and F1-score were only a few of the assessment measures used to assess the model's performance on the testing set. The outcomes demonstrate that the suggested model obtained a 99.91% accuracy rate. These findings show that the suggested methodology is very effective in identifying plant leaf diseases and is applicable to real-world situations.

In conclusion, the suggested methodology for detecting plant leaf diseases using deep learning techniques yields encouraging results. The use of CNNs provides a powerful and accurate approach for image classification tasks, and By implementing further strategies like data augmentation and transfer learning, the suggested model may be strengthened still more.

5.2 ARCHITECTURE OF PROPOSED SYSTEM

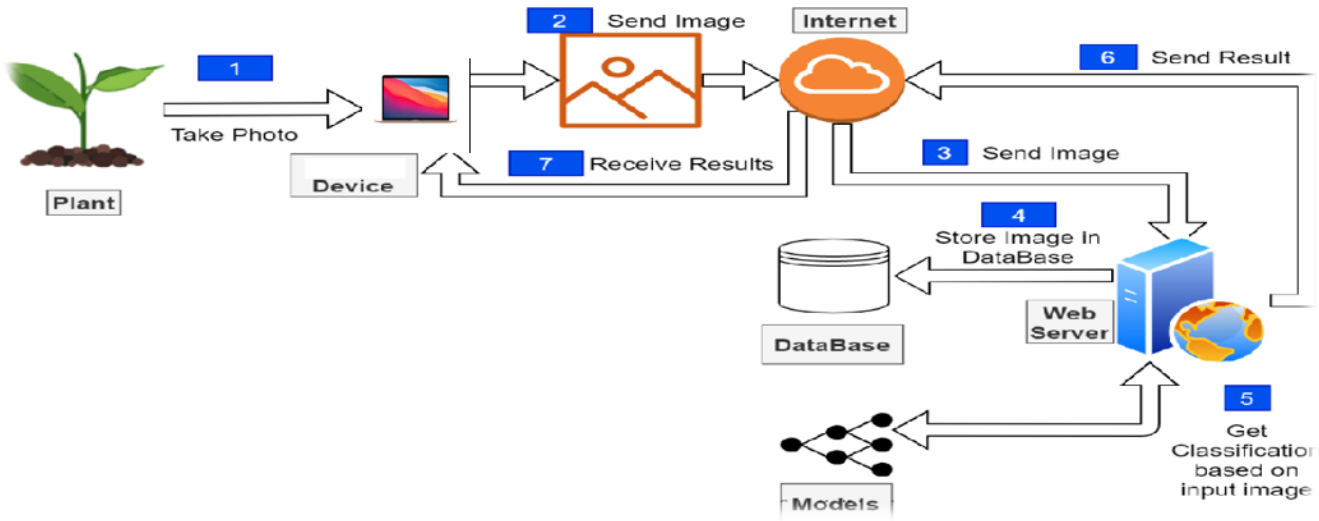


Figure 5.1 Architecture of Proposed System

5.3 MODULES

5.3.1 Pre-processing techniques

Pre-processing is an essential step in any machine learning project. In the context of the plant leaf classification project, pre-processing involves techniques such as data splitting, caching, and shuffling to ensure speed and accuracy. Data splitting involves dividing the dataset into separate sets for training, testing, and validation. This ensures that the model does not overfit to the training data and performs well on unseen data.

Caching is used to store the pre-processed data in memory or disk for faster retrieval during training. This reduces the overall time required for data loading and pre-processing, thereby increasing the efficiency of the model. Shuffling is another pre-processing technique that randomizes the order of the training examples to prevent the model from learning the order of the data.

All these pre-processing techniques help to ensure that the model trains on high-quality, unbiased data and performs well on new, unseen data. By optimizing the data inputs, pre-processing helps to improve the overall accuracy and efficiency of the model.

5.3.2 Segmentation

After pre-processing the dataset, the next step is to segment it into training, testing, and validation sets. Segmentation is important as it allows us to evaluate the performance of our model on data that it has never seen before, which is crucial to avoid overfitting.

Typically, the dataset is split into a training set (used to train the model), a validation set (used to tune hyperparameters and prevent overfitting), and a testing set (used to evaluate the final performance of the model). The split ratio can vary depending on the size of the dataset, but a common split is 70% for training, 15% for validation, and 15% for testing.

It is important to ensure that the data is split randomly and that each split contains a representative sample of the entire dataset. This can be achieved using techniques such as stratified sampling, which ensures that each class is represented proportionally in each split.

Overall, segmentation is a crucial step in the development of a machine learning model, as it allows us to evaluate its performance and make necessary adjustments to improve its accuracy.

5.3.3 Feature extraction

After segmenting the dataset, the next step is feature extraction. In this stage, we extract relevant information or features from the images that can be used by the machine learning model to differentiate between healthy and diseased plant leaves.

Color and texture features are extracted from the leaf images. Color-based features can be extracted using color histograms or color moments, while texture-based features can be extracted using techniques such as Local Binary Patterns (LBP) or Gray-Level Co-occurrence Matrix (GLCM).

These features can provide a rich representation of the plant leaf images, enabling the machine learning model to make accurate predictions. The choice of feature extraction technique depends on the nature of the dataset and the problem being solved.

Feature extraction is a critical step in the machine learning pipeline as it determines the quality of features that the model uses to make predictions. A good feature extraction method should be able to capture the essential characteristics of the plant leaves while eliminating irrelevant information.

5.3.4 Feature selection

After extracting the features from the images, it is important to select the most important and relevant features that contribute most towards the classification of the images as healthy or diseased. This step is called feature

selection. There are various methods to perform feature selection such as filter methods, wrapper methods, and embedded methods. In this project, filter methods are used for feature selection.

Filter methods work by ranking the features based on their relevance to the target variable. The features are scored using statistical tests such as chi-square, ANOVA, and correlation. The features with the highest scores are then selected for further analysis. The advantage of using filter methods is that they are computationally efficient and can handle a large number of features.

In this project, filter methods are used to select the most important features from the extracted color and texture features. The features are ranked based on their correlation with the target variable (healthy or diseased). The top-ranked features are then selected for training the deep learning model. This step helps in reducing the dimensionality of the dataset and improving the accuracy of the model.

5.3.5 Evaluation

Evaluation is an essential part of any machine learning project, as it helps to measure the performance of the model. In this project, we evaluated our CNN model using the ROC curve and accuracy metric. ROC curve is a plot that helps to visualize the performance of the classification model at different classification thresholds. It represents the trade-off between true positive rate (sensitivity) and false positive rate (1-specificity) for various threshold values. The ROC curve is a useful tool for selecting the best threshold for a particular classification problem.

Accuracy is another widely used metric to evaluate the performance of classification models. It represents the percentage of correctly classified samples in the test dataset. In this project, we used accuracy as a metric to evaluate the overall performance of our CNN model. However, accuracy alone may not be sufficient in some cases, especially when the dataset is imbalanced. In such cases, precision, recall, and F1 score may also be used to measure the

performance of the model.

To evaluate the performance of our model, we split the dataset into training, validation, and testing sets. We trained the model on the training set, tuned the hyperparameters on the validation set, and evaluated the model's performance on the testing set. We plotted the ROC curve and calculated the accuracy of the model on the test set. By analyzing the ROC curve and accuracy, we determined whether the model is overfitting or underfitting and made necessary adjustments to improve its performance.

5.3.6 Final Prediction

The final step in the project is to predict whether an input image of a plant leaf is healthy or diseased based on the trained model. To make a prediction, we pass the preprocessed and segmented image through the trained CNN model. The model will then output a probability score indicating the likelihood that the input image is healthy or diseased.

To make a final prediction, we set a threshold probability score. If the output probability score is greater than the threshold, we classify the input image as diseased, and if it is less than the threshold, we classify it as healthy. The threshold value can be set based on the desired trade-off between false positives and false negatives.

Once the final prediction is made, it can be displayed to the user along with a confidence score indicating the level of confidence that the model has in the prediction. This can help the user make informed decisions about the health of their plants and take appropriate actions to prevent or treat any diseases.

CHAPTER 6
ALGORITHM DESCRIPTION

6.1 CONVOLUTIONAL NEURAL NETWORK

6.1.1 Introduction to Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a type of deep learning algorithm that are commonly used for image recognition and classification tasks. CNNs are composed of multiple layers, including convolutional, pooling, and fully connected layers, each of which performs a specific function in the feature extraction and classification process.

The first layer in a CNN is the input layer, which takes in the image as a 2D matrix of pixel values. The convolutional layer then applies a set of filters to the input, each of which performs a convolution operation to extract specific features from the image. The resulting feature maps are then passed through a rectified linear unit (ReLU) activation function to introduce non-linearity and improve the model's ability to learn complex features.

After the convolutional layer, a pooling layer is typically applied to downsample the feature maps and reduce the dimensionality of the input. This is typically achieved using max pooling, which selects the maximum value in each region of the feature map. The pooled feature maps are then passed through additional convolutional and pooling layers, before being flattened and passed through a fully connected layer, which performs the final classification.

Overall, CNNs are highly effective at image recognition and classification tasks, due to their ability to learn hierarchical features from the input image through the use of multiple convolutional and pooling layers.

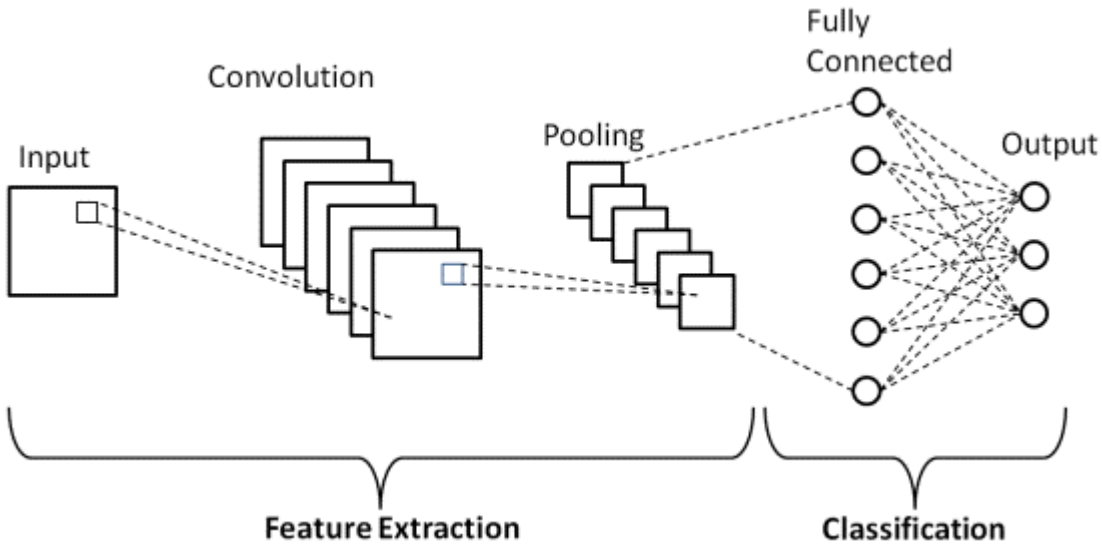


Figure 6.1 Architecture of CNN

6.1.2 Layers in CNNs

Convolutional Neural Networks (CNNs) are composed of several layers, each with a specific function in the feature extraction and classification process. The most common layers in a CNN include:

1. Input Layer: This layer takes in the input image as a 2D matrix of pixel values.

2. Convolutional Layer: This layer applies a set of filters to the input image, performing a convolution operation to extract specific features from the image. Each filter learns to recognize a specific pattern or feature in the image, such as edges or corners.

3. Activation Layer: This layer applies an activation function, such as the rectified linear unit (ReLU) function, to introduce non-linearity into the model and improve its ability to learn complex features.

4. Pooling Layer: This layer down-samples the feature maps produced by the convolutional layer, reducing the dimensionality of the input and making the model more computationally efficient. The most common pooling technique is max pooling, which selects the maximum value in each region of the feature

map.

5. Fully Connected Layer: This layer takes the flattened output from the previous layers and performs the final classification. Each neuron in the layer is connected to every neuron in the previous layer, allowing the model to learn complex patterns and relationships between the features.

6.1.3 Feature Extraction with Convolutional Layers

Convolutional Neural Networks (CNNs) are designed to learn and extract the most relevant features from an input image. The feature extraction process is performed by the convolutional layers in the CNN.

Each convolutional layer is composed of a set of filters, also known as kernels, which convolve over the input image to produce a set of feature maps. Each filter extracts a specific feature from the input image, such as an edge, a corner, or a blob.

During the convolution process, each filter scans the entire input image and computes a dot product between the filter weights and the corresponding pixel values in the image. This dot product operation results in a single output value, which is placed into the corresponding position in the feature map.

The output feature maps produced by the convolutional layers are a result of applying the filters to the input image. These feature maps contain important information about the spatial relationships between the features in the input image.

In order to capture different types of features, CNNs typically use multiple convolutional layers with increasing number of filters. This allows the model to learn more complex and abstract features as the information is passed through the layers.

Overall, the feature extraction process with convolutional layers is a crucial step in the CNN pipeline, as it enables the model to learn and recognize the most relevant features in the input image.

6.1.4 Optimization Techniques for CNNs

Convolutional Neural Networks (CNNs) are powerful deep learning models that have shown great success in various computer vision tasks, including image classification, object detection, and segmentation. However, training these models can be computationally expensive and time-consuming, especially when dealing with large datasets and complex architectures. To overcome these challenges, various optimization techniques have been developed to improve the efficiency and performance of CNNs.

1. Batch Normalization: Batch normalization is a technique that normalizes the activations of the previous layer for each batch during training. This helps in reducing the internal covariate shift, which is a phenomenon where the distribution of the inputs to a layer changes during training, making it harder to train the network. By normalizing the inputs, batch normalization allows the network to learn more efficiently and converge faster.

2. Dropout: Dropout is a regularization technique that randomly drops out a certain percentage of the neurons in a layer during training. This helps in preventing overfitting by reducing the network's dependence on any single neuron and forcing it to learn more robust and generalizable features.

3. Data Augmentation: Data augmentation is a technique that artificially increases the size of the training dataset by applying random transformations to the input images, such as rotations, translations, and flips. This helps in improving the generalization performance of the network by exposing it to more variations in the training data.

4. Transfer Learning: Transfer learning is a technique that leverages the pre-trained weights of a pre-existing CNN model and fine-tunes them on a new dataset or task. This helps in reducing the training time and data requirements by starting from a pre-trained model that has already learned generic features.

5. Gradient Descent Optimization: Gradient descent optimization is a technique that updates the weights of the network in the direction of the steepest descent of the loss function. Various algorithms have been developed to improve the efficiency and stability of gradient descent optimization, including stochastic gradient descent (SGD), Adam, and Adagrad.

Overall, these optimization techniques play a crucial role in improving the efficiency, performance, and generalization of CNNs, and are widely used in modern deep learning pipelines.

6.1.5 Advancements in CNNs

Advancements in CNNs have played a crucial role in revolutionizing various fields. One of the major advancements is the development of deep CNNs, which have allowed for more complex and accurate models. These models have been instrumental in various fields such as image recognition, natural language processing, and speech recognition.

Another significant advancement is the use of transfer learning, which involves using pre-trained models and fine-tuning them for specific tasks. This technique has allowed for faster and more efficient model development, especially in scenarios where large amounts of training data are not available. Overall, these advancements have made CNNs more powerful and versatile tools for a wide range of applications.

CHAPTER 7
APPLICATION DEVELOPMENT

7.1 Introduction To React JS

React JS is a popular open-source JavaScript library used for building user interfaces. Developed by Facebook, react allows developers to build complex UIs for single-page applications and mobile applications. The library is known for its ability to handle large-scale applications, improve performance, and promote code reusability. React uses a component-based architecture, allowing developers to break down the UI into reusable building blocks. These components can be reused across different parts of the application, making it easier to manage and maintain code.

React makes use of a virtual DOM, which is a lightweight in-memory representation of the actual DOM. When changes are made to the UI, react updates the virtual DOM instead of directly manipulating the actual DOM. This approach improves performance and makes the rendering process faster, as React only updates the parts of the DOM that need to be updated.

React is also known for its flexibility and extensibility. It can be used with other libraries and frameworks, making it easier to integrate with other parts of the application. React also has a large and active community of developers, who contribute to the development of the library, share their knowledge, and provide support to other developers.

Overall, React JS is a powerful tool for building modern web applications. Its component-based architecture, virtual DOM, and flexibility make it a popular choice for developers looking to build scalable and maintainable applications.

7.2 Front-End Development with React

Front-end development with React is an essential aspect of modern web development. React is a popular JavaScript library that allows developers to build highly responsive and efficient web applications. In the context of the project discussed in this paper, React can be used to develop the front-end of the web application for plant disease detection and remedial measures.

One of the main advantages of using React is its ability to handle dynamic content effectively. In the case of plant disease detection, the web application needs to handle large amounts of image data and perform real-time analysis. React can help in handling this data and displaying the results in an efficient and user-friendly way. By breaking down the user interface into components, React can ensure that each component is highly reusable and easy to manage.

Another advantage of using React is its compatibility with other libraries and frameworks. For instance, React can be used with Redux, a state management library, to create highly scalable and maintainable web applications. In addition, React can be easily integrated with other libraries such as Chart.js, which can be used to create dynamic and interactive charts to display the results of the plant disease analysis.

React also provides a wide range of tools and resources for front-end development, including the React Developer Tools browser extension, which can be used to inspect and debug React components. In addition, there are several online communities and forums dedicated to React development, which can provide valuable resources and support for developers.

React is a powerful tool for front-end development, which can be used to create highly responsive and efficient web applications. In the context of the project discussed in this paper, React can be used to develop the front-end of the

web application for plant disease detection and remedial measures, and its advantages in handling dynamic content, compatibility with other libraries and frameworks, and availability of tools and resources make it a valuable tool for developers.

7.3 Introduction to python flask

Python Flask is a popular web development framework that is used for building web applications in Python. It is a lightweight, easy-to-use framework that is ideal for building small to medium-sized web applications. Flask provides developers with the flexibility to choose the tools and libraries they want to use while building their applications. It is a micro web framework that doesn't require particular tools or libraries to build a web application.

Flask provides the developers with the basic features to build a web application like routing, request handling, and response rendering. It also supports various extensions that can be used to add more functionality to the application. Flask allows developers to write web applications in a minimalistic way without any overhead. It follows the model-view-controller (MVC) architecture, which separates the application logic into three parts: model, view, and controller.

Python Flask is easy to learn and use because it requires fewer lines of code compared to other web frameworks. It is a great choice for beginners who want to start building web applications with Python. Flask provides a simple and elegant way to create web applications, making it an excellent choice for building prototypes or proof-of-concept applications.

Python Flask is a versatile web development framework that is widely used by developers to build web applications in Python. It is easy to learn and use, and its flexibility and simplicity make it a popular choice among developers. With its wide range of features and extensions, Flask provides developers with

everything they need to build a robust and scalable web application.

7.4 Back-End Development with Python Flask

Backend development refers to the server-side of web development where the server-side code runs and serves the front-end part of the application. Python Flask is a lightweight web application framework that provides developers with a flexible, modular, and easy-to-use toolkit for building web applications. Flask is considered to be a micro-framework that doesn't require particular tools or libraries. It is designed to keep the core of the application simple, but still flexible enough to add new features as required.

One of the key advantages of using Python Flask for backend development is its simplicity. Flask is easy to learn and use, and its flexibility allows developers to customize their development approach to suit the needs of their project. Flask is also known for its speed and scalability, which makes it a popular choice for building high-performance applications.

Another advantage of using Python Flask is its extensive library of extensions. Flask provides a wide range of libraries that can be used to add additional functionality to a web application. These libraries cover a range of functionalities such as form handling, database management, authentication, and security. This enables developers to add features to their application quickly and easily, without having to write additional code from scratch.

Flask is also compatible with a wide range of databases, which makes it a versatile choice for building web applications. Flask supports popular databases such as PostgreSQL, MySQL, SQLite, and MongoDB. This flexibility enables developers to choose the database that best suits their needs and integrate it with their Flask application seamlessly.

Python Flask is a lightweight and flexible web application framework that provides developers with an easy-to-use toolkit for building web applications.

Its simplicity, speed, scalability, and versatility make it a popular choice for backend development. With a wide range of libraries and compatibility with various databases, Flask enables developers to build high-performance applications quickly and easily.

7.5 Integrating the Front-End and Back-End

Integrating the front-end and back-end of a web application is a crucial step in the development process. It involves connecting the user interface, created using front-end technologies such as React, with the back-end logic and database, implemented using a framework like Python Flask.

The integration process can be achieved through APIs (Application Programming Interfaces), which allow communication between the front-end and back-end. APIs provide a standardized method for different systems to communicate with each other, and they allow the front-end to request data from the back-end and receive it in a standardized format.

To integrate React with Python Flask, we can use various tools and libraries such as Axios, a popular JavaScript library for making HTTP requests from the front-end to the back-end. Additionally, Flask-RESTful, a Flask extension, can be used to create RESTful APIs for the back-end.

Once the API is created, the front-end can make requests to the back-end to fetch data or perform actions, such as adding or updating data in the database. The back-end processes the request and sends a response back to the front-end, which can be used to update the UI accordingly.

Overall, integrating the front-end and back-end is a critical step in developing a functional web application. Proper integration ensures that the application is robust, secure, and responsive.

CHAPTER 8
SYSTEM IMPLEMENTATION

8.1 MODULE DESCRIPTION:

8.1.1 Perform Data Visualization

In today's world, a lot of data is being generated on a daily basis. And sometimes to analyze this data for certain trends, patterns may become difficult if the data is in its raw format. To overcome this data visualization comes into play. Data visualization provides a good, organized pictorial representation of the data which makes it easier to understand, observe, analyze.

Data visualization is the graphical representation of information and data. By using visual elements like charts, graphs, and maps, data visualization tools provide an accessible way to see and understand trends, outliers, and patterns in data. In the world of Big Data, data visualization tools and technologies are essential to analyze massive amounts of information and make data-driven decisions. Python provides various libraries that come with different features for visualizing data. All these libraries come with different features and can support various types of graphs.

few popular plotting libraries:

- Matplotlib: low level, provides lots of freedom
- Pandas Visualization: easy to use interface, built on Matplotlib
- Seaborn: high-level interface, great default styles
- ggplot: based on R's ggplot2, uses Grammar of Graphics
- Plotly: can create interactive plots

8.1.2 Dataset:

The 2,152 photos of plant leaves from two different disease types—potato early blight and potato late blight—as well as healthy leaves made up the dataset used for this investigation. These pictures were gathered from different places and pre-processed to eliminate any unnecessary background and make the picture size uniform. A ratio of 70:10:20 was used to divide the dataset into training, validation, and testing sets. This made sure the model was tested on previously unexplored photos and trained on a varied group of images. The dataset played a crucial role in the success of the model as it allowed the model to learn and generalize from a large and diverse set of images, leading to higher accuracy and reliability in disease classification.

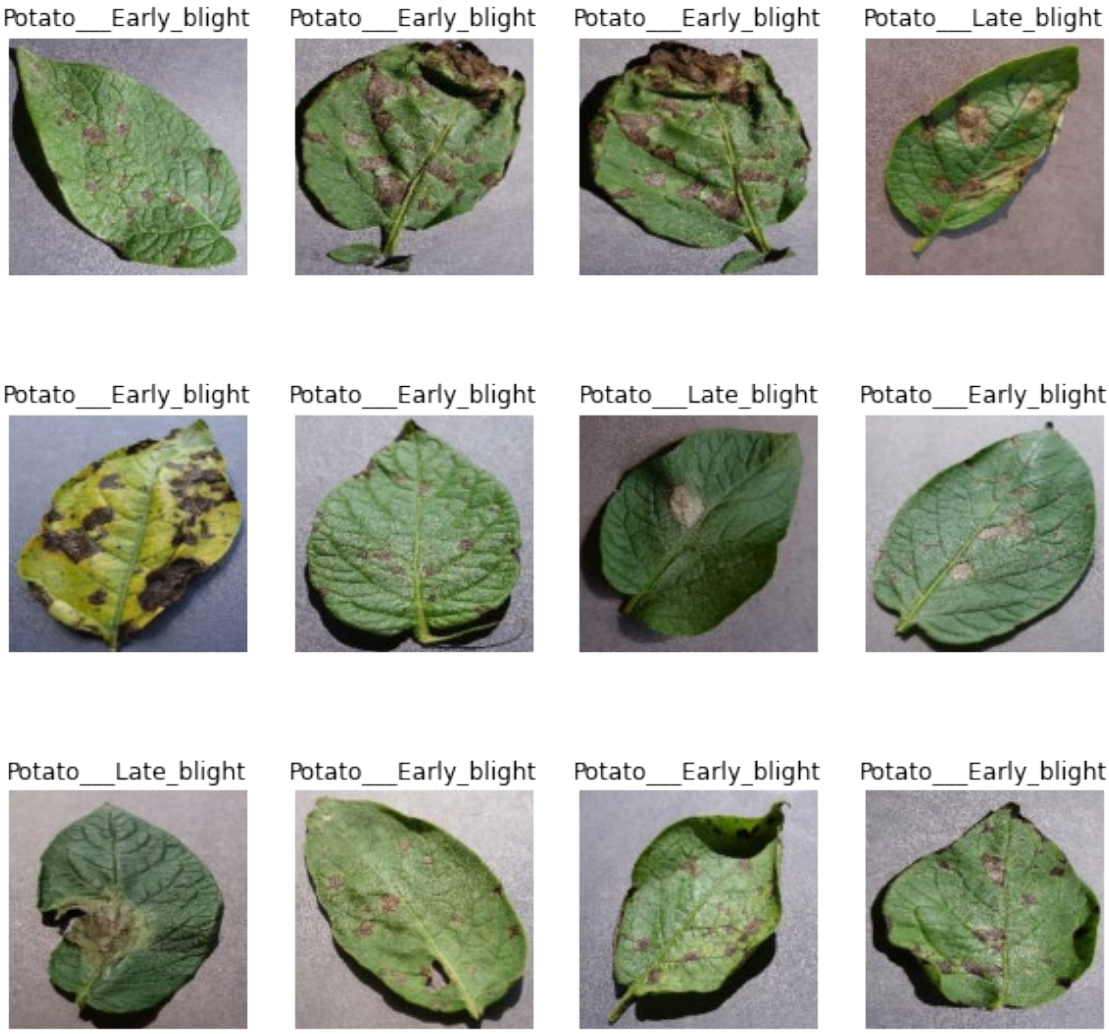


Figure 7.1 The Dataset of the model

8.1.3 Input Data Preprocessing:

Input data preprocessing is a crucial step in any machine learning project, including CNNs. The objective of this module is to prepare the input data for training the model. The quality and suitability of the input data can significantly impact the accuracy of the trained model.

In this module, the input data is usually preprocessed to ensure consistency and accuracy. The data may be normalized or standardized to ensure that it is suitable for use with the model. Additionally, the data may be augmented to increase the number of training samples, thereby reducing overfitting. This augmentation can involve random rotations, scaling, or cropping of the input images.

Other preprocessing steps may include filtering, denoising, or smoothing the input data to remove noise or artifacts that can adversely affect the training process. Finally, the preprocessed data is typically split into training, validation, and test datasets. The training dataset is used to train the model, the validation dataset is used to tune hyperparame:ers, and the test dataset is used to evaluate the model's performance on unseen data.

8.1.4 Training and Validation

The training and validation module is an essential part of any machine learning project, including CNNs. In this module, the preprocessed data is used to train and validate the model. The training process involves iteratively adjusting the model's parameters to minimize the difference between the predicted and actual values.

During the training process, the model is exposed to a portion of the preprocessed input data, and the model learns the relationships between the input data and the output labels. The model's weights are adjusted using an optimization algorithm, such as stochastic gradient descent (SGD), to minimize the difference between the predicted output and the actual output.

In the validation process, the model's performance is evaluated on a separate dataset that was not used for training. This is done to evaluate the model's generalization ability and to prevent overfitting. Overfitting occurs when the model is too complex and fits the training data too closely, resulting in poor performance on unseen data.

Hyperparameters, such as learning rate, regularization strength, and batch size, can be tuned during the validation process to improve the model's performance. Once the model's hyperparameters have been tuned and its performance on the validation dataset is satisfactory, the model can be evaluated on the test dataset.

Overall, the training and validation module is a critical step in training a CNN model that accurately and robustly predicts the output for new, unseen input data.

8.1.5 Splitting up of the Data

This process consists of splitting the data into two sets, training dataset and testing dataset.

8.1.5.1 Training Dataset

The sample of data used to fit the model. The model sees and learns from this data. Training data is an extremely large dataset that is used to teach a machine learning model. Training data is used to teach prediction models that use machine learning algorithms how to extract features that are relevant to specific business goals. For supervised ML models, the training data is labelled. The data used to train unsupervised ML models is not labelled. The idea of using training data in machine learning programs is a simple concept, but it is also very foundational to the way that these technologies work.

The training data is an initial set of data used to help a program understand how to apply technologies like neural networks to learn and produce sophisticated results. It may be complemented by subsequent sets of data called

validation and testing sets. Training data is also known as a training set, training dataset or learning set.

8.1.5.2 Test Dataset

The sample of data used to provide an unbiased evaluation of a final model fit on the training dataset. A test data set is a data set that is independent of the training data set, but that follows the same probability distribution as the training data set. If a model fit to the training data set also fits the test data set well, minimal overfitting has taken place. A better fitting of the training data set as opposed to the test data set usually points to over-fitting.

A test set is therefore a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier. To do this, the final model is used to predict classifications of examples in the test set. Those predictions are compared to the examples' true classifications to assess the model's accuracy.

In a scenario where both validation and test datasets are used, the test data set is typically used to assess the final model that is selected during the validation process. In the case where the original data set is partitioned into two subsets (training and test datasets), the test data set might assess the model only once (e.g., in the holdout method). Note that some sources advise against such a method. However, when using a method such as cross-validation, two partitions can be sufficient and effective since results are averaged after repeated rounds of model training and testing to help reduce bias and variability.

8.1.6 Build and Train the Model

Building and training a CNN model involves defining the architecture of the model, including the number and type of layers, and then training the model on the input data. During the training process, the model's parameters are optimized to minimize the difference between the predicted and actual output values. The trained model can then be used for prediction on new, unseen input data.

8.1.7 OVERVIEW OF JUPYTER NOTEBOOK

The Jupyter Notebook is an open-source web application that can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

The Jupyter Notebook application allows to create and edit documents that display the input and output of a Python or R language script. Once saved, it can be shared with others.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through “Download As” in the web interface, via the nbconvert library or “jupyter nbconvert” command line interface in a shell.

Jupyter Notebook provides an easy-to-use, interactive data science environmentacross many programming languages that doesn't only work as an IDE, but alsoas a presentation or education tool. It's perfect for those who are just starting outwith data science!

Language Independent: Because of its representation in JSON format, Jupyter Notebook is platform-independent as well as language-independent. Another reason is that Jupyter can be processed by any several languages, and can be converted to any file formats such as Markdown, HTML, PDF, and others. Jupyter will always be 100% open-source software, free for all to use and released under the liberal terms of the modified BSD license. Jupyter is developed in the open on GitHub, through the consensus of the Jupyter community.

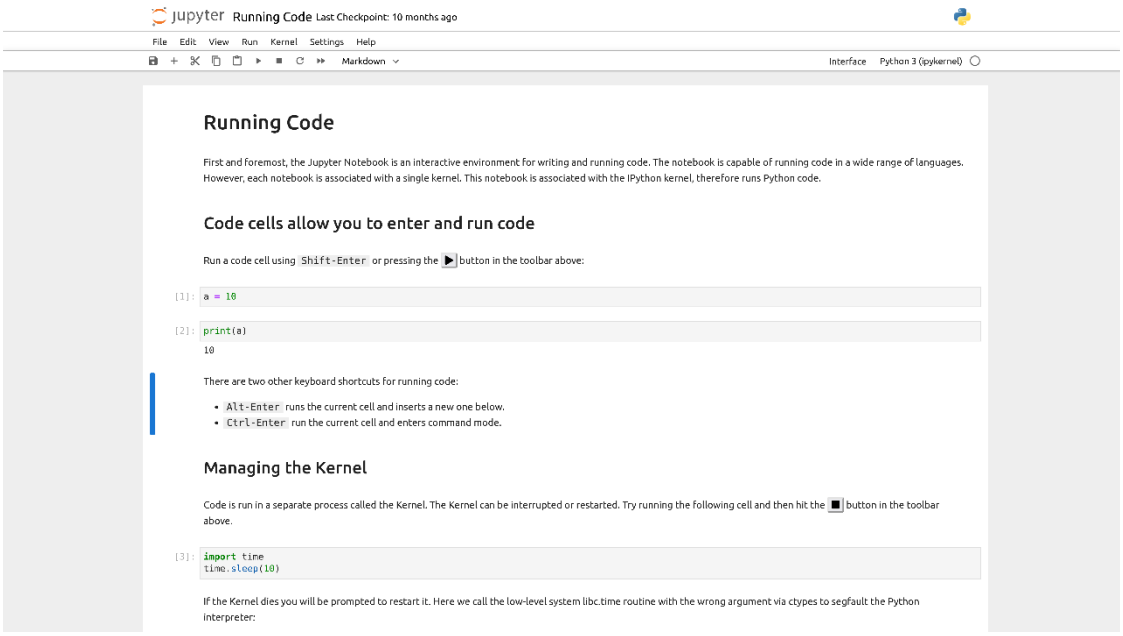


Figure 7.2 Overview of jupyter notebook

8.2 INSTALLING JUPYTER NOTEBOOKS

Jupyter Notebook is widely-used and well-documented, and provides a simple file browser along with the environment for creating, editing, and running the notebooks. Jupyter Lab is more complex, with a user environment more reminiscent of an Integrated Development Environment. Jupyter Lab is meant to eventually replace Jupyter Notebook, there is no indication that Jupyter Notebook will stop being supported anytime soon. Because of its comparative simplicity and ease of use for beginners, this tutorial uses Jupyter Notebook as the software for running notebook files. Both software packages are included in Anaconda, described below. It's easiest to use Anaconda to Install jupyter Notebook, but if Python installed on your system and don't want to deal with the large Anaconda package, run `pip3 install jupyter`.

8.3 ANACONDA

Anaconda is a free, open-source distribution of Python and R that comes with more than 1,400 packages, the Conda package manager for installing additional packages, and Anaconda Navigator, which allows to manage environments (e.g. one can install different sets of packages for different projects, so that they don't cause conflicts for one another) using a graphical interface. After installing Anaconda, use Anaconda Navigator to install new packages (or `conda install` via the command line), but many packages are

available only through pip (i.e. using pip install via the command line or in Jupyter notebook).

For most purposes, download the Python 3 version of Anaconda, but some legacy code may still be written in Python 2. In this system Python 3 is used. The Anaconda installer is over 500 MB, and after installation it can take upwards of 3 GB of hard drive space, so make sure to have enough room on the computer and a fast network connection before get started.

8.4 NAVIGATING THE JUPYTER NOTEBOOK INTERFACE

From Anaconda, to view a notebook through the Jupyter interface, launch Jupyter Notebook first (which will display in a browser window), and open the file from within Jupyter Notebook. Unfortunately, there is no way to set Jupyter Notebook as the default software application to open .ipynb files by double-click on them.

When launching Jupyter Notebook from Anaconda Navigator, it automatically displays in the home directory. This is usually the directory with the username on a Mac. On a PC it is usually C:\. If Jupyter Notebook from the command lineis launced, it will display the contents of the folder when it Was launched (Using the command line, one can also directly launch a specific notebook, e.g. jupyter notebook example.ipynb.)

8.5 FEATURES OF JUPYTER NOTEBOOK

1. Free and Easy start-up, just type juptyer notebook in the terminal
2. Visualization display (user interface)
3. Text editing (general code commenting, markdown, code prettify, collapsible headings, highlighter, spellchecker, scratchpad)

4. All in one place: Jupyter Notebook is an open-source web-based interactive environment that combines code, text, images, videos, mathematical equations, plots, maps, graphical user interface and widgets to a single document.
5. Easy to convert: Jupyter Notebook allows users to convert the notebooks into other formats such as HTML and PDF. It also uses online tools and nbviewer which allows one to render a publicly available notebook in the browser directly.
6. Easy to share: Jupyter Notebooks are saved in the structured text files (JSON format), which makes them easily shareable.

8.6 RESULTS AND DISCUSSION

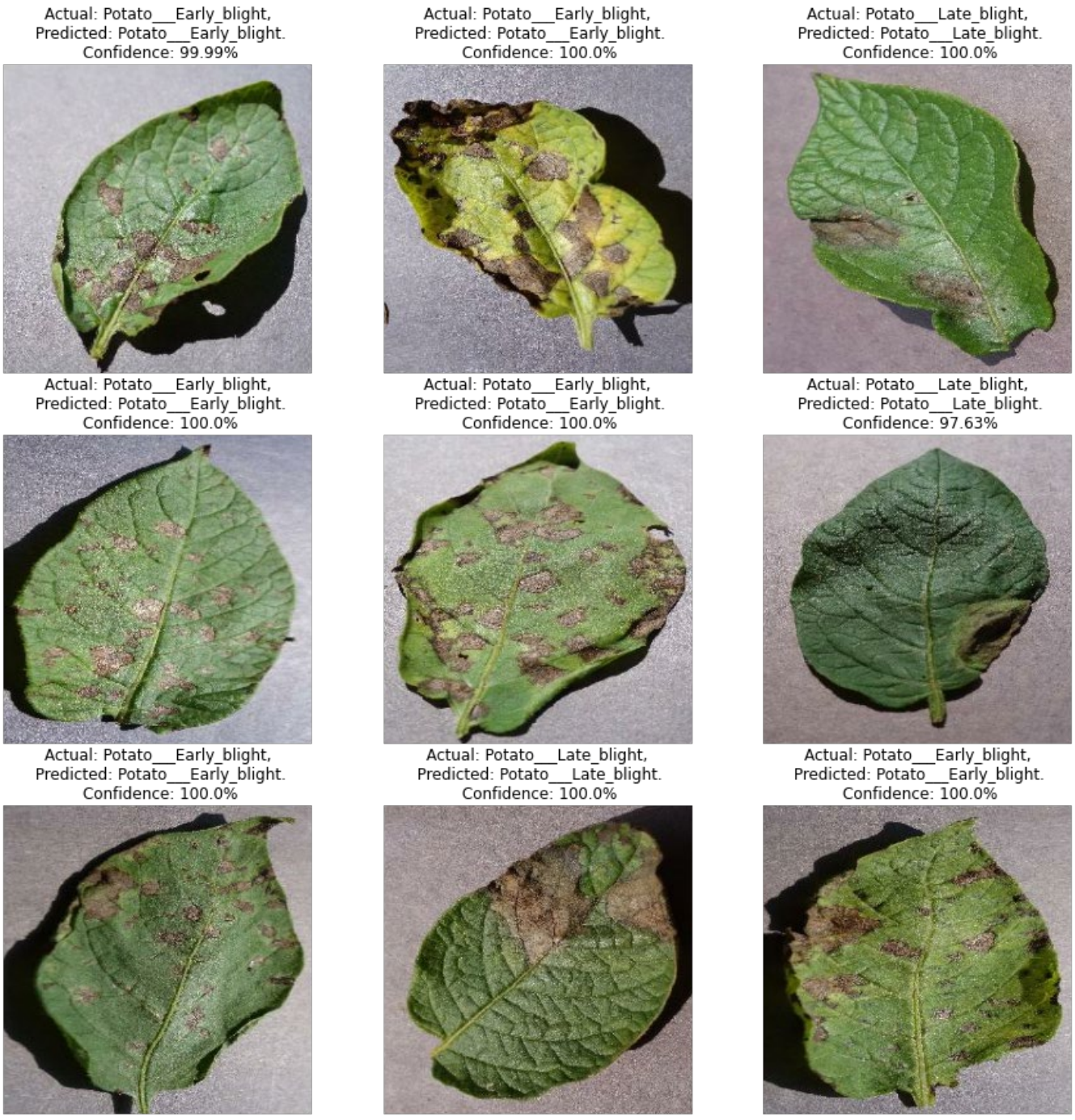


Figure 7.3 Result of CNN model

8.7 COMPARING OTHER MODEL WITH CNN

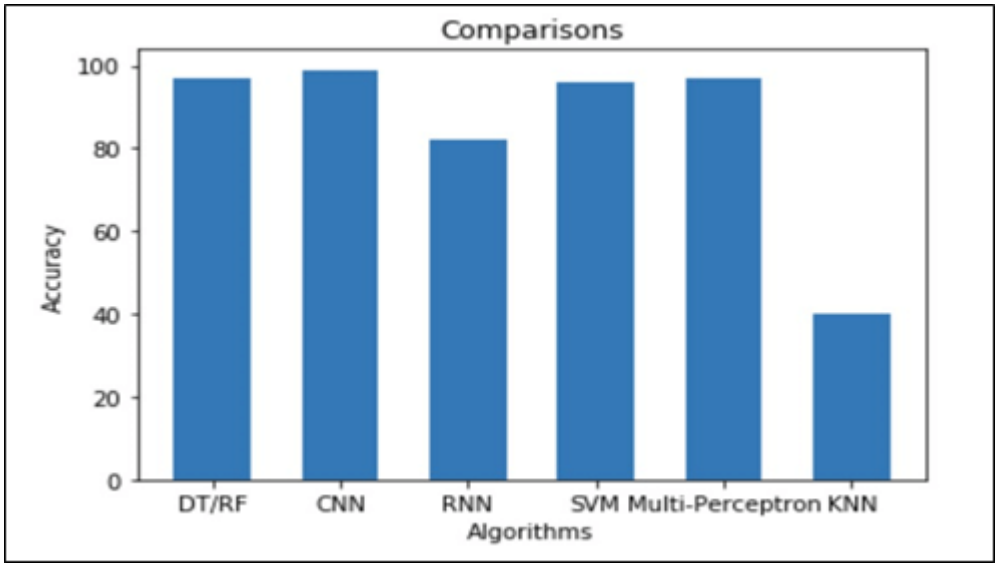


Figure 7.4 Comparing different Models with CNN using a Bar Graph

CHAPTER 9

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, Sub-assemblies, assemblies and\or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test type addresses a specific testing requirement.

9.1 TESTING STEPS

- Unit Testing
- Integration Testing
- Functional testing

9.1.1 Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

9.1.2 Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defect, ,the task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

9.1.3 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : Classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

CHAPTER 10
CONCLUSION AND FUTURE ENHANCEMENT

10.1 CONCLUSION:

Our study proved how well deep convolutional neural networks operate for identifying and categorizing plant leaf diseases. The created model successfully identified two widespread plant diseases with an F1-score of 99.91% accuracy. This technique can greatly increase the accuracy of plant disease detection, resulting in quicker and more efficient treatments. This technology has the potential to transform the agricultural sector and reduce the detrimental effects of plant diseases on crop output with more study and development.

10.2 FUTURE ENHANCEMENTS:

Dataset expansion: While the current dataset used in this study is relatively large, there are still many plant diseases that are not included. To increase the precision and robustness of the model, more pictures of sick plant leaves can be gathered and uploaded to the dataset in the future.

Transfer learning: The utilisation of transfer learning is another interesting area for future research. Using a smaller, more focused dataset, like the plant leaf disease dataset utilised in this work, this approach entails fine-tuning a pre-trained CNN that has been trained on a large, diversified dataset. This may enable the model to operate more effectively while requiring less training time.

Real-time disease detection: Currently, the model developed in this study is designed to classify images of diseased plant leaves. However, in the future, it would be interesting to explore the possibility of developing a real-time system that can detect diseases as they occur in the field. This could involve using cameras or other sensors to monitor plants and alert farmers when disease is detected, allowing for more timely and targeted treatment.

Multi-class classification: The present model can discriminate between healthy leaves and two separate illnesses (potato early blight and potato late blight). In the future, it would be valuable to extend the model to be able to classify a wider range of diseases, allowing it to be used in more diverse agricultural settings. This would require a larger and more diverse dataset, as well as modifications to the architecture of the CNN to support multi-class classification.

APPENDIX 1

SOURCE CODE

```
import tensorflow as tf
from tensorflow.keras import models, layers
import matplotlib.pyplot as plt
from IPython.display import HTML
BATCH_SIZE = 32
IMAGE_SIZE = 256
CHANNELS=3
EPOCHS=50

dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "PlantVillage",
    seed=123,
    shuffle=True,
    image_size=(IMAGE_SIZE,IMAGE_SIZE),
    batch_size=BATCH_SIZE
)

class_names = dataset.class_names
class_names

for image_batch, labels_batch in dataset.take(1):
    print(image_batch.shape)
    print(labels_batch.numpy())

plt.figure(figsize=(10, 10))
for image_batch, labels_batch in dataset.take(1):
    for i in range(12):
        ax = plt.subplot(3, 4, i + 1)
        plt.imshow(image_batch[i].numpy().astype("uint8"))
        plt.title(class_names[labels_batch[i]])
        plt.axis("off")

def get_dataset_partitions_tf(ds, train_split=0.8, val_split=0.1, test_split=0.1,
shuffle=True, shuffle_size=10000):
    assert (train_split + test_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)
```

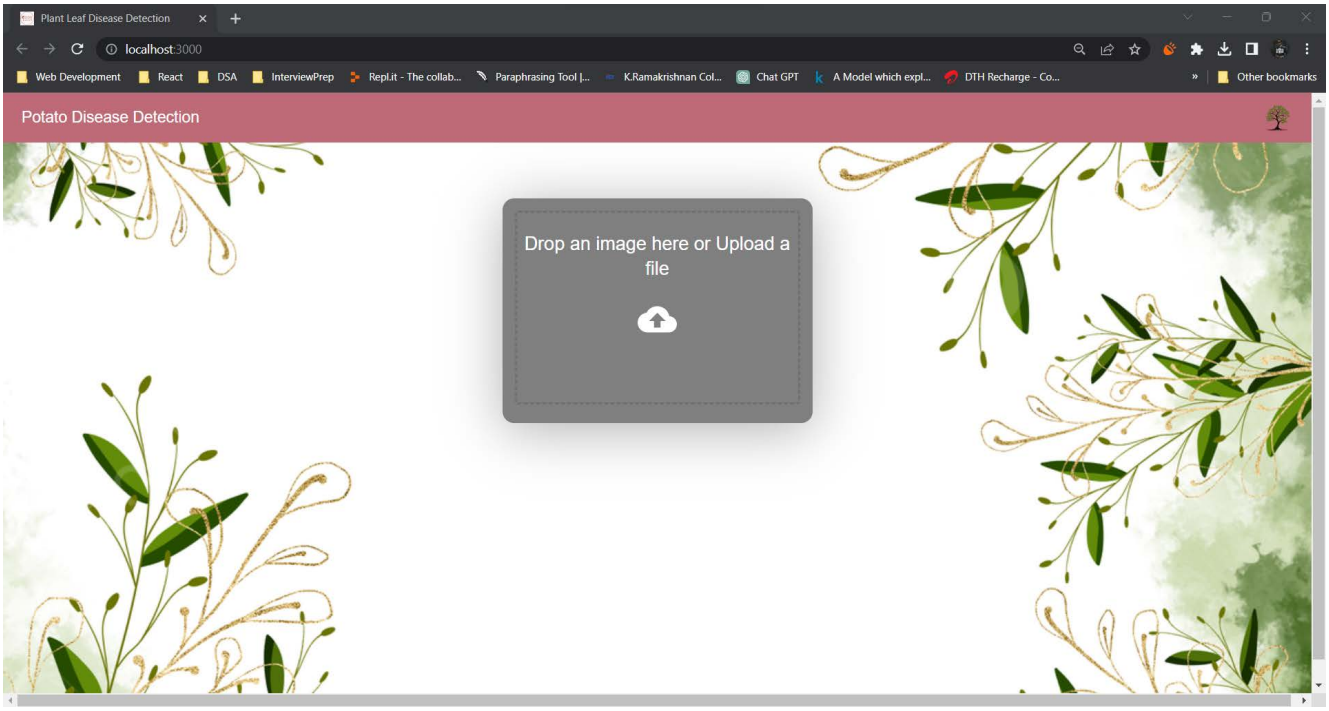
```
    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size).take(val_size)
    test_ds = ds.skip(train_size).skip(val_size)

    return train_ds, val_ds, test_ds

train_ds, val_ds, test_ds = get_dataset_partitions_tf(dataset)

train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds = val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds = test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)

resize_and_rescale = tf.keras.Sequential([
  layers.experimental.preprocessing.Resizing(IMAGE_SIZE, IMAGE_SIZE),
  layers.experimental.preprocessing.Rescaling(1./255),
])

data_augmentation = tf.keras.Sequential([
  layers.experimental.preprocessing.RandomFlip("horizontal_and_vertical"),
  layers.experimental.preprocessing.RandomRotation(0.2),
])

train_ds = train_ds.map(
    lambda x, y: (data_augmentation(x, training=True), y)
).prefetch(buffer_size=tf.data.AUTOTUNE)


input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 3

model = models.Sequential([
    resize_and_rescale,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu',
input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64,  kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
```

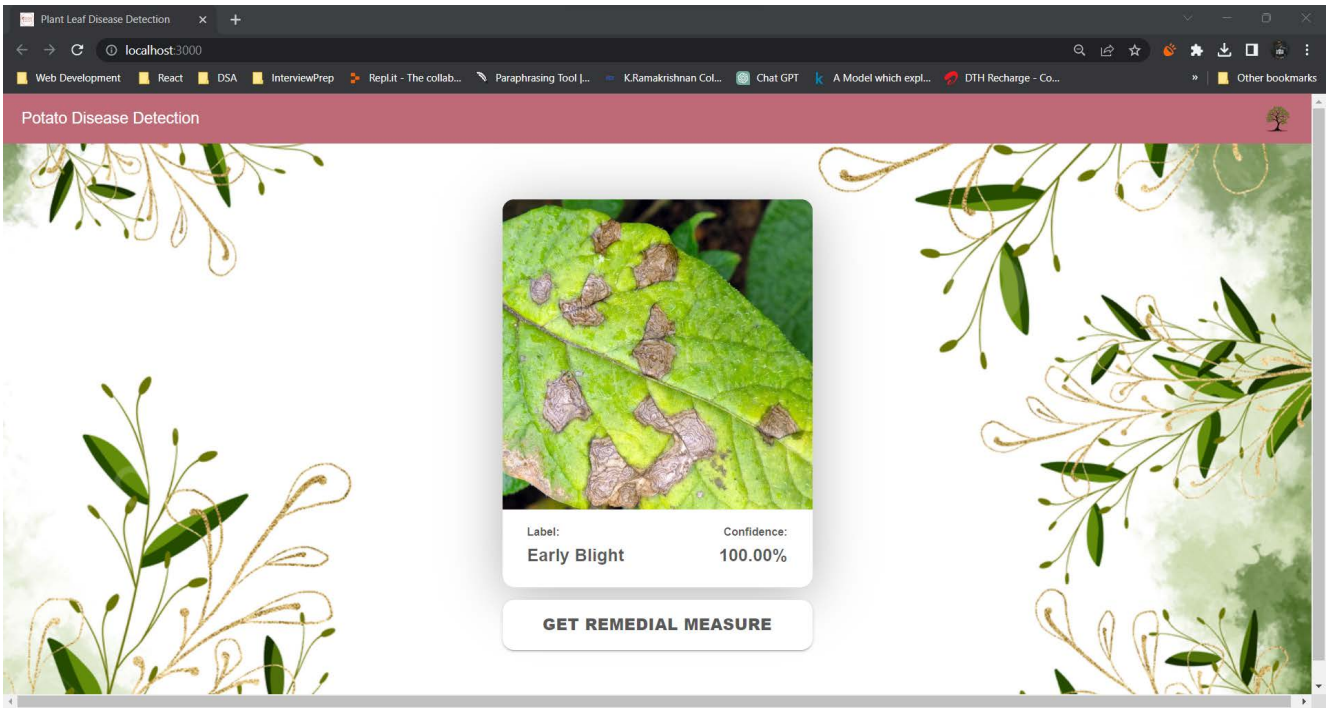
```
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])

model.build(input_shape=input_shape)

model.summary()

model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)

history = model.fit(
    train_ds,
    batch_size=BATCH_SIZE,
    validation_data=val_ds,
    verbose=1,
    epochs=50,
)

scores = model.evaluate(test_ds)

scores

history

history.params

history.history.keys()

history.history['loss'][:5] # show loss for first 5 epochs

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(range(EPOCHS), acc, label='Training Accuracy')
```

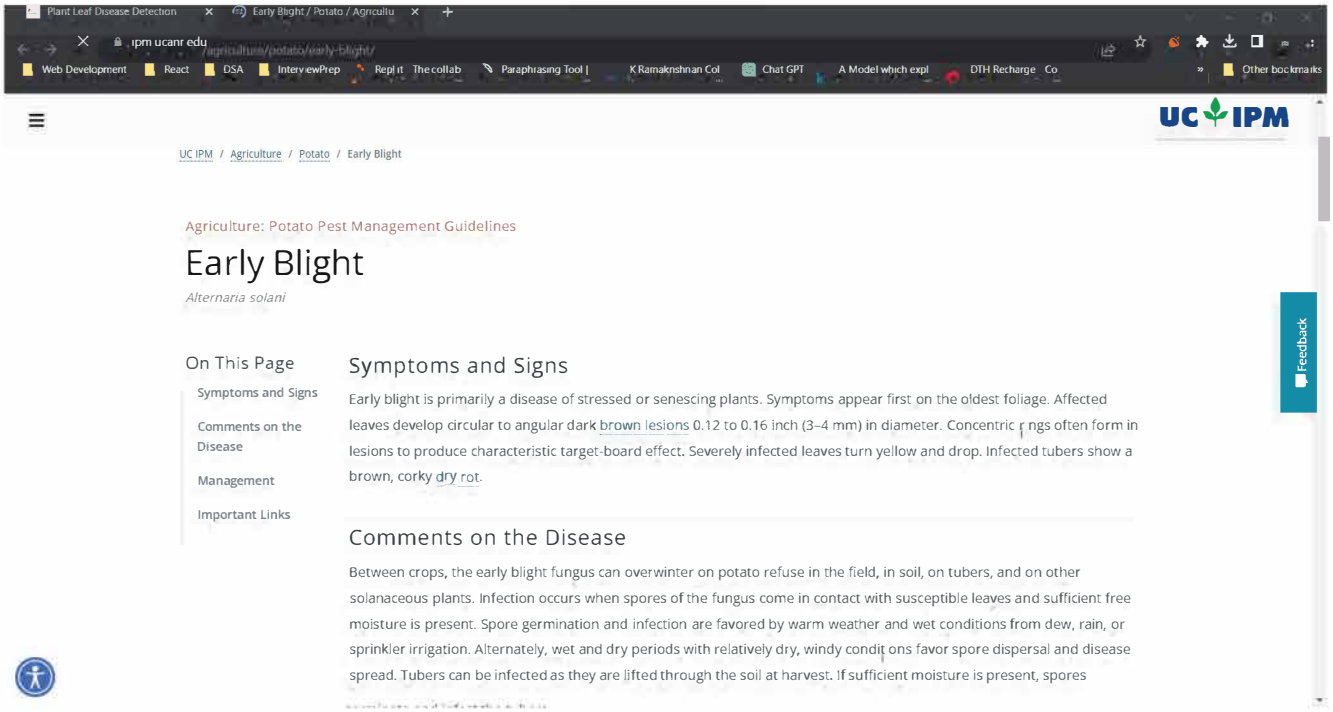
```
plt.plot(range(EPOCHS), val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(range(EPOCHS), loss, label='Training Loss')
plt.plot(range(EPOCHS), val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()

import numpy as np
for images_batch, labels_batch in test_ds.take(1):

    first_image = images_batch[0].numpy().astype('uint8')
    first_label = labels_batch[0].numpy()

    print("first image to predict")
    plt.imshow(first_image)
    print("actual label:",class_names[first_label])

    batch_prediction = model.predict(images_batch)
    print("predicted label:",class_names[np.argmax(batch_prediction[0])])

def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)
    return predicted_class, confidence

plt.figure(figsize=(15, 15))
for images, labels in test_ds.take(1):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))

        predicted_class, confidence = predict(model, images[i].numpy())
        actual_class = class_names[labels[i]]

        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence:
{confidence}%")
```

```
        plt.axis("off")

import os
model_version=max([int(i) for i in os.listdir("../models") + [0]])+1
model.save(f"../models/{model_version}")

model.save("../potatoes.h5")
```

APPENDIX 2
SCREENSHOTS






Plant Leaf Disease Detection
Early Blight / Potato / Agricultu
ipm.ucanr.edu
Web Development
React
DSA
InterviewPrep
Replit
The collab
Paraphrasing Tool |
K Ramakrishnan Col
Chat GPT
A Model which expl
DTH Recharge - Co
Other bookmarks
UC IPM
UC IPM / Agriculture / Potato / Early Blight
Agriculture: Potato Pest Management Guidelines
Early Blight
Alternaria solani
On This Page
Symptoms and Signs
Comments on the Disease
Management
Important Links
Symptoms and Signs
Early blight is primarily a disease of stressed or senescing plants. Symptoms appear first on the oldest foliage. Affected leaves develop circular to angular dark brown lesions 0.12 to 0.16 inch (3–4 mm) in diameter. Concentric rings often form in lesions to produce characteristic target-board effect. Severely infected leaves turn yellow and drop. Infected tubers show a brown, corky dry rot.
Comments on the Disease
Between crops, the early blight fungus can overwinter on potato refuse in the field, in soil, on tubers, and on other solanaceous plants. Infection occurs when spores of the fungus come in contact with susceptible leaves and sufficient free moisture is present. Spore germination and infection are favored by warm weather and wet conditions from dew, rain, or sprinkler irrigation. Alternately, wet and dry periods with relatively dry, windy conditions favor spore dispersal and disease spread. Tubers can be infected as they are lifted through the soil at harvest. If sufficient moisture is present, spores
Feedback

REFERENCES

[1] Singh, S. K., & Gupta, S. K. (2021). Deep learning-based plant disease detection: A comprehensive review. In Intelligent Computing Techniques for Smart Energy Systems (pp. 307-321). Springer, S

[2] Sharma, R., & Kanwar, P. (2021). An overview of plant disease detection techniques using deep learning. Journal of Ambient Intelligence and Humanized Computing, 12(7), 6773-6786.

[3] Raza, G., & Raza, A. (2021). Deep learning-based plant disease detection using transfer learning: A review. International Journal of Computer Science and Network Security, 21(4), 138-147.

[4] Patel, S. K., & Patel, S. R. (2021). A review on plant disease detection and classification using deep learning approach. In Proceedings of International Conference on Emerging Trends in Computer Science and Information Technology (pp. 373-383). Springer.

[5] Nair, M., Gopi, R., & Kumar, S. (2021). Plant disease detection using deep learning techniques: A review. In Advances in Intelligent Systems and Computing (Vol. 1331, pp. 134-143). Springer.

[6] Munda, S., Singh, P. K., & Mishra, S. (2021). Deep learning-based plant disease detection: A review. Journal of Plant Diseases and Protection, 128(2), 169-187.

[7] Koirala, A., & Panday, S. (2021). A review of deep learning for plant disease detection. SN Computer Science, 2(1), 1-17.

[8] Khan, A. M., Shafique, S., & Awan, I. A. (2021). Plant disease classification using deep learning: A survey. Computers and Electronics in Agriculture, 186, 106074.

[9] Kaur, R., & Verma, M. (2021). A survey of plant disease detection using deep learning techniques. International Journal of Engineering and Advanced Technology, 10(4), 467-475.

[10] Acharya, D., Kumar, P., & Gautam, A. K. (2021). Identification of plant

diseases: A review on deep learning approach. International Journal of Intelligent Systems and Applications, 13(8), 1-11.

[11] Ferdous, S., Al Mamun, S., Arafat, Y., & Islam, M. R. (2020). Leaf disease detection of different crops: A deep learning approach. International Journal of Computer Applications, 179(44), 16-23.

[12] Ghosal, S., Kumar, A., & Bandyopadhyay, S. (2019). An overview of deep learning based approaches for plant disease detection and diagnosis. Computers and Electronics in Agriculture, 162, 219-232.

[13] Singh, D., & Jindal, M. (2019). Leaf disease detection and classification using deep learning: A review. Journal of Ambient Intelligence and Humanized Computing, 10(6), 2061-2084.

[14] Li, H., Guo, X., Liu, Q., & Zhang, Z. (2019). A novel deep learning method for plant disease detection based on Gabor wavelet and convolutional neural network. Frontiers in Plant Science, 10, 118.

[15] Wang, S., Zhang, X., & Zhang, J. (2019). A review of image classification algorithms for plant diseases. Plant Methods, 15(1), 1-14.

[16] Xie, F., Liu, Y., & Zou, X. (2019). A survey on deep learning based plant disease recognition. Neurocomputing, 330, 208-221.

[17] Osman, M., Rahman, M. M., & Zhang, Y. (2018). Early plant disease detection using image processing and machine learning techniques. Journal of Advanced Research in Dynamical and Control Systems, 10(8), 219-226.

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

[19] You, Q., Yan, S., Zhang, H., & Tang, X. (2015). Robust deep learning for improved classification of traffic signs. IEEE Transactions on Neural Networks and Learning Systems, 27(8), 1738-1751.

[20] Sánchez, C., & Poggio, T. (2011). Deep learning: advances and challenges. A not-so-short review. arXiv preprint arXiv:1803.01164.

CONFERENCE PARTICIPATION PROOF/JOURNAL PUBLICATION PROOF

SRI SAI REC SRI SAI RANGANATHAN ENGINEERING COLLEGE

Approved by AICTE New Delhi Affiliated to Anna University,Chennai

Department of Computer Science and Engineering & Information Technology

Proudly Presents

International Conference on Innovative Engineering and Information Technology

ICIEIT' 2K23

CERTIFICATE

This is to certify that Dr. /Prof./Mr. /Mrs. /Ms. **PRADEEP KUMAR S**
of **K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY** *has presented a paper titled* **An Efficient Deep Learning-Based System for Accurate Plant Leaf Disease Detection and Remedial Measures.** *in the International Conference on Innovative Engineering and Information Technology(ICIEIT' 23) held at Sri Sai Ranganathan Engineering College , Coimbatore during 28th,29th April 2023.*

PRINCIPAL

CHAIRMAN

SRI SAI REC SRI SAI RANGANATHAN ENGINEERING COLLEGE

Approved by AICTE New Delhi Affiliated to Anna University,Chennai

Department of Computer Science and Engineering & Information Technology

Proudly Presents

International Conference on Innovative Engineering and Information Technology

ICIEIT' 2K23

CERTIFICATE

This is to certify that Dr. /Prof./Mr. /Mrs. /Ms. SHRI VISHNU. P

of K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY *has presented a paper titled* An Efficient Deep Learning-Based System for Accurate Plant Leaf Disease Detection and Remedial Measures. *in the International Conference on Innovative Engineering and Information Technology(ICIEIT' 23) held at Sri Sai Ranganathan Engineering College , Coimbatore during 28th,29th April 2023.*

PRINCIPAL

CHAIRMAN

SRI SAI REC SRI SAI RANGANATHAN ENGINEERING COLLEGE

Approved by AICTE New Delhi Affiliated to Anna University,Chennai

Department of Computer Science and Engineering & Information Technology

Proudly Presents

International Conference on Innovative Engineering and Information Technology

ICIEIT' 2K23

CERTIFICATE

This is to certify that Dr. /Prof./Mr. /Mrs. /Ms. SOMASUNDARAM S

of K. RAMAKRISHNAN COLLEGE OF TECHNOLOGY *has presented a paper titled* An Efficient Deep Learning-Based System for Accurate Plant Leaf Disease Detection and Remedial Measures. *in the International Conference on Innovative Engineering and Information Technology(ICIEIT' 23) held at Sri Sai Ranganathan Engineering College , Coimbatore during 28th,29th April 2023.*

PRINCIPAL

CHAIRMAN