RSET
RAJAGIRI SCHOOL OF ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

Project Phase 2 Report On

Smart Farming System

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

Bachelor of Technology

in

Electronics And Communication Engineering

By

Anithra Ross Ajith (U2101035)
Alka Denny (U2101023)
Anna Joju (U2101039)
Anu Xavier (U2101042)

Under the guidance of

Ms. S.Santhi Jabarani

Department Of Electronics And Communication Engineering
Rajagiri School of Engineering & Technology (Autonomous)
(Parent University: APJ Abdul Kalam Technological University)
Rajagiri Valley, Kakkanad, Kochi, 682039

April 2025

CERTIFICATE

This is to certify that the Project phase 2 report entitled ***"Smart Farming System"*** *is a bonafide record of the work done by* ***Anithra Ross Ajith (U2101035)*** *submitted to the Rajagiri School of Engineering & Technology (RSET) (Autonomous) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in "Electronics And Communication Engineering" during the academic year 2024-2025.*

Project Guide:
Ms. S.Santhi Jabarani
Assistant Professor
Dept. of ECE
RSET

Project Co-ordinator:
Ms. Swapna Davies
Assistant Professor
Dept. of ECE
RSET

HoD:
Dr. Jisa David
Associate Professor
Dept. of ECE
RSET

External Examiner:

ACKNOWLEDGMENT

I wish to express my sincere gratitude towards **Dr. Jaison Paul Mulerikkal**, Principal of RSET, and **Dr. Jisa David**, Head of the Department Of Electronics And Communication Engineering for providing me with the opportunity to undertake my project, "Smart agriculture system".

I am highly indebted to my project coordinator, **Ms. Swapna Davies** , Assistant Professor, Department Of Electronics And Communication Engineering,for her valuable support.

It is indeed my pleasure and a moment of satisfaction for me to express my sincere gratitude to my project guide **Ms. S.Santhi Jabarani** for her patience and all the priceless advice and wisdom she has shared with me.

Last but not the least, I would like to express my sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Anithra Ross Ajith

Abstract

Agriculture has always been the backbone of human civilization, and with the right technology, we can make it more efficient, sustainable, and productive. Our project, Smart Agricultural System Using FarmBot, brings together deep learning and automation to help farmers detect plant diseases early and manage their farms more effectively.

At first, we trained our model using DenseNet121 and ResNet50 on a dataset containing 7–9 plant diseases. DenseNet121 initially performed better, but when we expanded the dataset to 39 plant diseases, ResNet50 outperformed it in classification accuracy. We implemented the trained model on a Raspberry Pi, which processes real-time images captured by an Web-Cam module. These images are analyzed and matched with the trained model's predictions to accurately identify plant diseases.

Once the system detects an issue, it automatically takes necessary actions using FarmBot. This includes spraying fertilizers, adjusting irrigation, monitoring soil moisture, and keeping track of weather conditions. By automating these tasks, the system reduces manual labor, optimizes resource usage, and ensures healthier crops, ultimately leading to better agricultural productivity.

By integrating machine learning and precision farming, our project offers a cost-effective and scalable solution for modern agriculture. With real-time disease detection and automated farm management, it provides farmers with the tools they need to maximize yield and minimize losses.

This project highlights the power of smart farming, paving the way for a future where technology and agriculture work hand in hand to ensure sustainable and efficient food production.

Contents

List of Abbreviations

CNN - Convolution Neural Network

VGG - Visual Geometry Group

IoT - Internet of Things

AI - Artifical Intelligence

DHT - Digital Humidity and Temperature Sensor

GPIO- General Purpose Input/Output

LM - Linear Microcircuit

DC - Direct Current

OpenCV - Open Source Computer Vision

PR - Precision - Recall

F1 - F1 Score

PO - Project Outcomes

PRO - Project Objective

PSO - Project Specific Outcomes

TP - True Positives

TN - True Negatives

FP - False Positives

FN - False Negatives

List of Figures

List of Tables

Chapter 1

Introduction

Agriculture remains the backbone of our economy, supplying food and essential raw materials, yet traditional farming practices often struggle with inefficiencies such as labor-intensive processes and imprecise resource management. As the global demand for food increases, so does the need for sustainable methods that enhance crop health and yield. Modern agricultural practices are evolving to integrate advanced technologies that address these challenges by leveraging data-driven insights and automation.

This project focuses on developing a smart agriculture system that automates plant disease detection and optimizes resource management. Utilizing state-of-the-art computer vision and deep learning techniques—specifically the ResNet50 model—the system is designed to analyze plant images in real time for early detection of diseases. Upon identifying a problem, the system triggers automated responses such as precise irrigation and targeted fertilizer spraying, thereby reducing manual intervention and ensuring crops receive the care they need promptly.

By incorporating sensor data and robust machine learning models, this approach improves farming efficiency and promotes sustainable practices without relying on remote connectivity. The local processing capabilities of the system ensure quick decision-making and precise control over agricultural inputs. Ultimately, this project represents a significant step toward modernizing traditional farming, reducing resource waste, and increasing overall productivity in a sustainable manner.

1.1 Background

Traditional farming practices, while foundational, often suffer from inefficiencies in resource management, labor-intensive processes, and delayed disease detection, which contribute to reduced crop yields, increased production costs, and environmental degradation. Recent advances in computer vision and deep learning have provided promising

solutions to these challenges, with models like ResNet50 enabling the automated analysis of crop images to extract critical features and identify early signs of disease. Our project builds on these technological breakthroughs by developing a smart agriculture system that locally processes real-time images of crops to detect diseases, and then automatically triggers targeted actions such as precise irrigation and fertilizer spraying without relying on external connectivity. This integrated approach not only streamlines farm operations and reduces dependence on manual labor but also optimizes the use of resources, thereby paving the way for more sustainable, efficient, and productive farming practices that can support long-term food security.

1.2 Problem Definition

- **Inefficiency in Traditional Farming**:Conventional farming methods rely on manual labor and outdated techniques, leading to low productivity and resource wastage.Farmers struggle with unpredictable weather conditions, soil degradation, and inefficient irrigation practices.
- **Lack of Real-Time Monitoring and Data**:Manual observation of crops and environmental factors is time-consuming and prone to errors.Without real-time data, farmers make uninformed decisions, resulting in poor resource management and lower yields.
- **Overuse of Resources and Environmental Impact**:Excessive use of water, fertilizers, and pesticides leads to environmental pollution and soil depletion.Inefficient irrigation systems contribute to water scarcity and increased farming costs.
- **Need for Smart Technology in Agriculture**:The integration of IoT, sensors, and automation can optimize farming operations and enhance productivity. Smart Agriculture Systems provide data-driven insights for better crop management, sustainability, and cost reduction.

1.3 Scope and Motivation

The scope of the Smart Agriculture System is to revolutionize traditional farming practices by integrating modern technologies such as IoT, sensors, and automation. This system enables real-time monitoring of critical parameters like soil moisture, temperature,

humidity, and weather conditions, allowing farmers to make data-driven decisions. By optimizing irrigation, reducing wastage of fertilizers and pesticides, and improving crop health, the system enhances overall efficiency, sustainability, and productivity in agriculture. Additionally, automation minimizes manual effort and operational costs while ensuring precise and timely interventions for better yield and resource management.

The motivation behind this project arises from the increasing challenges faced by farmers, such as unpredictable weather patterns, inefficient irrigation methods, and excessive use of chemical inputs, leading to soil degradation and environmental harm. Traditional farming methods are often labor-intensive and lack real-time data, resulting in suboptimal decision-making. With the growing global demand for food and the need for sustainable agricultural practices, implementing smart technology offers a viable solution. By equipping farmers with intelligent tools, this system not only improves productivity but also promotes eco-friendly farming, ensuring long-term food security and environmental conservation.

1.4 Objectives

- Implement smart techniques to optimize the use of water, fertilizers, and pesticides, reducing waste and ensuring sustainable farming practices.
- Utilize advanced agricultural methods to enhance crop growth, prevent diseases, and increase overall productivity.
- Develop automated solutions for irrigation, fertilization, or other critical farming operations to reduce manual labor and improve efficiency

1.5 Challenges

One of the major challenges in implementing a Smart Agriculture System is the high initial cost associated with automation and advanced farming techniques, which can be a barrier for small-scale farmers. Additionally, many farmers lack technical knowledge to operate and maintain automated systems effectively, making training and awareness essential. The maintenance and reliability of these systems also pose difficulties, as regular calibration and servicing are required to ensure optimal performance. Furthermore, environmental and soil variability can impact the efficiency of automation, making it

difficult to create a universally adaptable solution. Another significant challenge is the dependence on electricity and power supply, particularly in rural areas where access to stable energy sources may be limited. Integrating automation with traditional farming methods can also be complex, as many farmers are accustomed to conventional techniques and may be hesitant to transition. Additionally, the cost and availability of high-quality equipment can affect the affordability and scalability of the system. Overcoming these challenges requires a combination of financial support, proper training, and technological advancements to make smart agriculture more accessible and effective for all farmers.

1.6 Assumptions

In designing the Smart Agriculture System, several assumptions are made to ensure smooth operation and reliability. It is assumed that the 3.7V battery will provide a stable and continuous power supply without frequent failures or voltage drops. The system is designed to function effectively under normal agricultural conditions, considering moderate temperature and humidity levels. Additionally, the sensors used are assumed to be accurate and reliable, ensuring precise data collection for better decision-making. All electronic components are expected to be fully compatible, allowing seamless integration and efficient performance. Furthermore, the system is assumed to require minimal maintenance, with the battery lasting for a reasonable period before needing replacement or recharging. The system is also expected to function continuously without major disruptions, provided proper setup and periodic monitoring are carried out. It is assumed that farmers or users will have basic knowledge of operating and maintaining the system. Lastly, the system is expected to function without significant interference from external factors such as extreme weather conditions or electromagnetic disturbances.

1.7 Societal / Industrial Relevance

The Smart Agriculture System plays a crucial role in improving farming efficiency and sustainability. It helps farmers optimize resource usage, reducing water and fertilizer wastage. By enhancing crop productivity, it contributes to food security and economic growth. The system minimizes environmental impact by promoting eco-friendly agricultural practices. Farmers benefit from reduced labor costs and increased automation,

making farming more efficient. Industries related to agriculture and agritech can leverage this system for precision farming and better crop monitoring. It supports innovation in the agricultural sector by integrating modern technology into traditional farming methods. The system ensures cost-effective solutions for large-scale and small-scale farmers alike. With improved crop management, it enhances profitability and long-term sustainability. Overall, it bridges the gap between conventional farming and modern agricultural advancements.

In a nutshell, smart agriculture systems pave the way for transformation in the sector-mostly for the small and large farmer. With advanced technology, the system further extends sustainable solutions for maximum productivity, minimal destruction of the environment, and maximum economic benefits for farmers. This shift not only supports food security and climate adaptation but also helps in innovation and global competitiveness. Smart agriculture is an essential pillar for establishing farming prosperity in the future.

Chapter 2

Literature Survey

This literature review presents a variety of studies based on the suitability of deep learning architectures, among them AlexNet, DenseNet, and ResNet, for the accurate identification and classification of plant diseases. Each of the studies exploits particular properties of these architectures, namely model fine-tuning and transfer learning, to overcome challenges introduced by the particular nature of agricultural datasets - variability in lighting, image quality, and inter-class similarity for classes of disease. This review tracks these contributions to develop insight on the effectiveness of different deep learning models in real-time disease identification and points to potential integration into automated agricultural systems as a means to enhance precision, reduce waste, and improve crop health.

2.1 Smart farming using Machine Learning and Deep Learning techniques

Advancements in precision agriculture have significantly improved crop monitoring and automated irrigation systems. Various studies have explored the integration of deep learning models, such as ResNet50, for plant disease detection using image processing techniques. Researchers have demonstrated that convolutional neural networks (CNNs) can accurately classify plant diseases by analyzing leaf images, leading to early diagnosis and timely intervention. Furthermore, automated spraying mechanisms controlled by microcontrollers like Arduino and Raspberry Pi have been implemented to optimize the use of water and fertilizers. Studies highlight the effectiveness of relay-based control systems for activating pumps and sprayers based on sensor inputs or AI-driven disease detection models. The integration of IoT with smart farming techniques has also shown promising results in reducing manual labor and resource wastage while increasing crop yield. This research builds upon these advancements by employing a ResNet50-based plant disease detection system, which triggers automated spraying mechanisms through relay modules upon disease identification. [1]

2.2 Plant Disease Detection using ResNet

Deep learning has significantly improved plant disease detection, particularly with Convolutional Neural Networks (CNNs) like ResNet50. Unlike traditional machine learning methods that relied on handcrafted features, ResNet50 enables deeper network training, improves feature extraction, and achieves high classification accuracy. Studies have reported over 97% accuracy on datasets like PlantVillage, demonstrating its effectiveness in identifying plant diseases with minimal human intervention. Furthermore, its residual learning framework allows for better gradient flow, making it suitable for complex classification tasks. Recent research focuses on enhancing model efficiency, integrating attention mechanisms, and deploying lightweight models for real-time precision agriculture applications. Additionally, efforts are being made to combine ResNet50 with IoT-based smart farming solutions to enable automated disease monitoring and targeted intervention. [2]

2.3 Palm Leaf Health Management: A Hybrid Approach for Automated Disease Detection and Therapy Enhancement

Deep learning, hybrid methods, and IoT have significantly advanced plant disease detection and management. Convolutional Neural Networks (CNNs) ensure high accuracy, while hybrid models combining traditional image processing with machine learning enhance efficiency. IoT-based monitoring systems enable early disease detection and automated treatment, improving precision agriculture. However, challenges such as data scarcity, environmental variability, and model interpretability persist, necessitating further research into federated learning and multimodal fusion techniques. This study aims to address these gaps by integrating disease detection with real-time intervention, ultimately enhancing agricultural productivity. [3]

2.4 FarmBot- A Platform for Backyard Precision Farming: Installation and Initial Experimental Layout.

The advancement of automated agricultural systems has significantly improved precision farming, integrating IoT, computer vision, and deep learning models like ResNet50 for plant disease detection. Several studies have explored the efficacy of convolutional neural networks (CNNs) in identifying plant diseases from images, with models such as ResNet50

proving effective due to their deep-layered architecture and feature extraction capabilities. Moreover, the use of automated spraying systems, including relay modules and actuators for water and fertilizer application, has enhanced efficiency in modern farming. Research also emphasizes the role of real-time monitoring through webcam modules, facilitating instant disease identification and targeted spraying. Additionally, integration with smart irrigation mechanisms, such as automated water sprinklers, optimizes resource usage, reducing water and fertilizer wastage. These advancements collectively contribute to sustainable agriculture, ensuring higher yields and reduced environmental impact. However, challenges such as computational resource constraints and the need for high-quality datasets remain areas of ongoing research and improvement. [4]

2.5 Farmbot: an IoT-Based Wireless Agricultural Robot for Smart Cultivation

Recent advancements in smart agriculture have leveraged deep learning, hybrid approaches, and IoT for automated plant disease detection and precision farming. Convolutional Neural Networks (CNNs) such as ResNet50 and MobileNet have demonstrated high accuracy in disease classification, while hybrid models integrating traditional image processing with machine learning techniques enhance efficiency. IoT-based systems equipped with wireless sensor networks enable real-time monitoring, allowing early detection and automated treatment of diseases. Additionally, automated irrigation and spraying mechanisms, often controlled via relay modules and cloud-based platforms, optimize resource utilization. However, challenges such as environmental variability, data scarcity, and system scalability persist. Future research should explore federated learning, multimodal fusion, and improved sensor calibration to enhance accuracy and reliability. This study builds on these advancements by integrating disease detection with real-time intervention, contributing to the development of a more efficient and intelligent agricultural framework. [5]

Chapter 3

Methodology

The smart farming system presented in this project integrates machine learning and automation to enhance agricultural efficiency. The system is designed to operate autonomously, detecting leaf diseases in plants, analyzing soil moisture conditions, and making real-time decisions to optimize farming processes. The key component of this system is a rover, which navigates a predefined track in the field, scanning crops for potential diseases using a deep learning model based on ResNet50. Once a disease is identified, the rover halts its movement, classifies the detected disease, and initiates a sequence of automated actions.

The rover is equipped with a soil moisture sensor that measures the moisture content of the soil at the point where the disease is detected. If the soil is found to be dry, an automatic water sprinkling mechanism is activated to ensure adequate hydration for the plants. Additionally, upon disease detection, the system triggers a fertilizer spraying mechanism to apply the necessary treatment to the affected crops. Furthermore, the system is capable of continuously monitoring environmental conditions by measuring temperature and humidity levels every two seconds. These readings provide valuable insights into the farm's microclimate, allowing for better analysis and optimization of the agricultural environment.

The entire system is implemented through an integrated hardware-software approach, where the hardware components, including sensors and actuators, are controlled using embedded programming techniques. The software aspect is primarily responsible for processing sensor inputs, executing machine learning predictions, and controlling the actuators based on the detected conditions. This methodology ensures a seamless and automated farming process that reduces manual labor, enhances precision in disease detection, and optimizes resource utilization.

3.1 Block Diagram for Leaf Dataset Classification

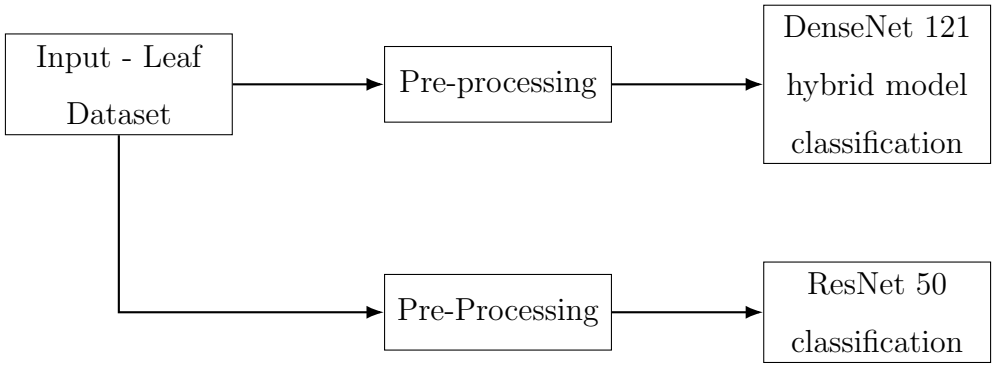


Figure 3.1: Block Diagram for Leaf Dataset Classification

A machine learning project designed to classify tomato leaf diseases using two advanced deep learning models, DenseNet-121 and ResNet-50, arranged in a dual-path system was implemented. The project begins with an input dataset containing images of healthy and diseased tomato leaves, representing common ailments like bacterial spots and blights. This dataset undergoes a pre-processing phase where images are standardized through resizing, normalization, and background removal to eliminate irrelevant variations and enhance feature clarity. Pre-processed data is then split into two paths: one directed to DenseNet-121, known for its densely connected layers that enable efficient feature propagation and reuse, and the other to ResNet-50, a model that employs residual blocks to effectively handle complex image features. DenseNet-121, possibly enhanced as a hybrid model in this project, excels at capturing detailed patterns, while ResNet-50's deep architecture is adept at recognizing subtle variations, providing complementary strengths for classification. Both models independently analyze the data to classify each leaf image based on disease indicators, and their outputs are integrated into a trained dataset, representing the final classifications. This dataset may use ensemble techniques or decision rules to combine model predictions, resulting in highly accurate classifications for each leaf sample. By leveraging both models in parallel, this system enhances its ability to generalize across various disease types and achieve reliable results. Ultimately, this project offers a robust automated solution for detecting tomato leaf diseases, helping farmers and agricultural professionals to monitor plant health and take preventive action promptly. The dual-model structure and meticulous pre-processing contribute to a powerful tool that could improve disease management practices, reduce crop losses, and

boost agricultural productivity through timely disease diagnosis.

3.2 Block Diagram of the Smart Agriculture System

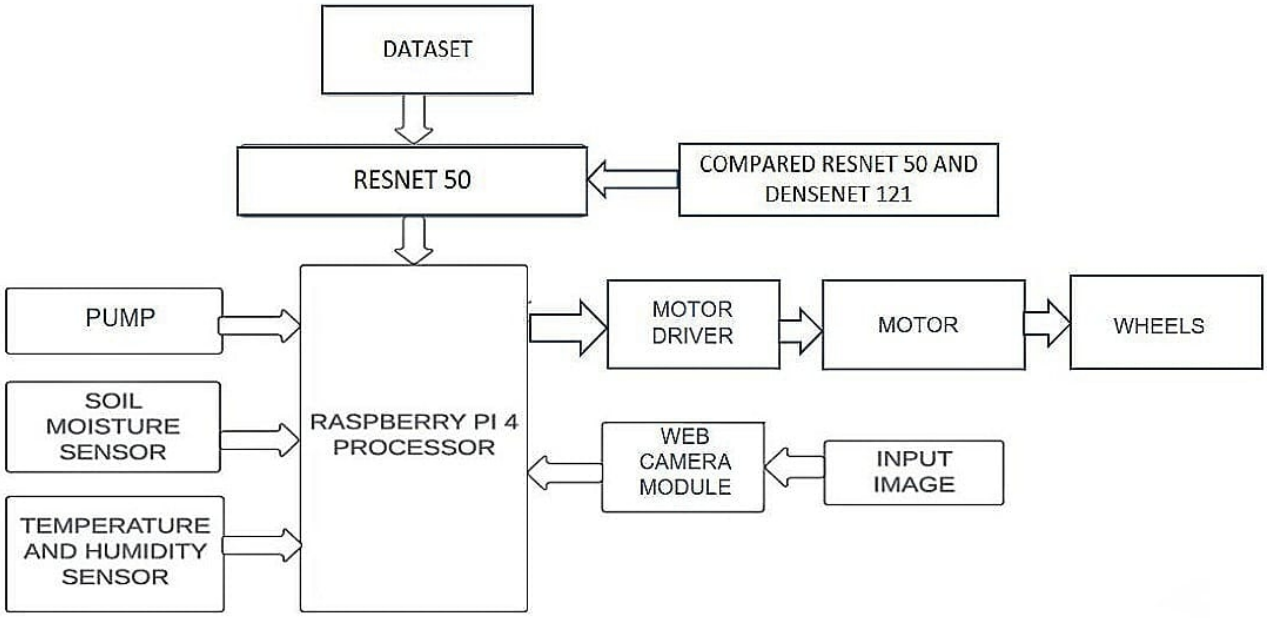


Figure 3.2: Block Diagram

3.3 Machine Learning Algorithms

In this project, we explore two powerful deep learning architectures ResNet50 and DenseNet121 both widely used for image classification tasks. These models are based on Convolutional Neural Networks (CNNs) and are known for their exceptional ability to learn complex features from visual data.

3.3.1 DenseNet 121

DenseNet121 (Densely Connected Convolutional Network) is a powerful convolutional neural network architecture comprising 121 layers, designed to maximize feature reuse and improve the efficiency of deep neural networks. It is particularly well-suited for image classification tasks such as plant disease detection due to its ability to maintain strong gradient flow and reduce the number of parameters. One of the most distinctive characteristics of DenseNet121 is its dense connectivity, where each layer receives inputs from all preceding layers and forwards its own feature maps to all subsequent layers within

the same block. This leads to enhanced feature propagation, more compact models, and mitigates the vanishing gradient problem common in very deep networks.

The image processing pipeline in DenseNet121 begins with preprocessing steps applied to the input images. These steps ensure the data is in the correct format and quality for the network to process effectively. Each input image is resized to 224×224 pixels to match the network's required input dimensions. Since images can come in varying sizes and aspect ratios, resizing is crucial for maintaining consistency across the dataset. Next, normalization is applied to scale the pixel values, typically using the mean and standard deviation values of the ImageNet dataset if pre-trained weights are used. This normalization helps stabilize and accelerate the training process. In many practical applications such as plant disease classification, data augmentation techniques like random rotation, flipping, cropping, and brightness adjustment are also applied to increase the diversity of the training data and reduce overfitting.

Once the image is preprocessed, it is passed through an initial 7×7 convolutional layer with a stride of 2, which extracts low-level features such as edges and textures. This is followed by a 3×3 max pooling layer that reduces the spatial dimensions, effectively downsampling the image and focusing on the most prominent features. The image then passes through four dense blocks containing 6, 12, 24, and 16 convolutional layers respectively. Each convolutional layer within a dense block consists of batch normalization, ReLU activation, a 1×1 bottleneck convolution, and a 3×3 convolution. The 1×1 convolution reduces the number of input channels to improve computational efficiency, while the 3×3 convolution extracts deeper features.

Between the dense blocks, transition layers are used. These consist of a 1×1 convolution followed by a 2×2 average pooling layer, which further compresses the feature maps and controls the complexity of the network. The final layers of DenseNet121 include a global average pooling layer that reduces each feature map to a single value by averaging, effectively summarizing the spatial features. This output is then passed to a fully connected (dense) layer which performs the final classification based on the learned features.

DenseNet121's ability to preserve and reuse features throughout the network makes it highly accurate and computationally efficient. When used for applications like plant disease detection, it can be fine-tuned using transfer learning. Pre-trained on large datasets

like ImageNet, the model can adapt to specific tasks with relatively fewer training examples, achieving excellent performance in classifying various plant diseases with high accuracy. Its dense connections ensure that even early features in the network are not lost, making it an ideal choice for deep learning tasks that require precision and generalization.

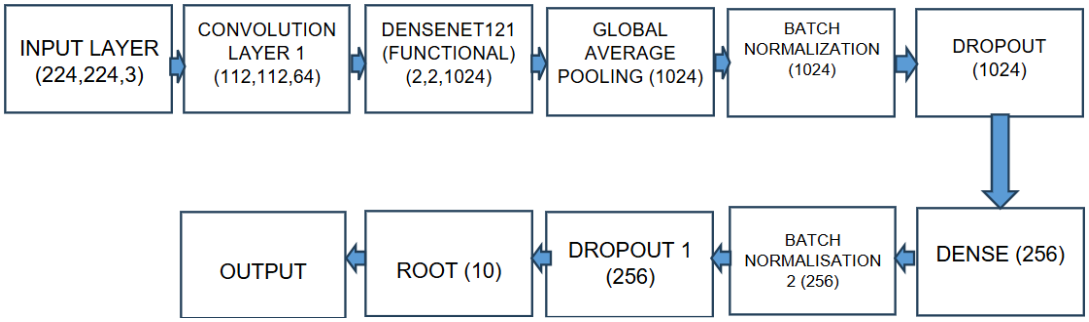


Figure 3.3: DenseNet121 Architecture

3.3.2 ResNet 50

ResNet50, short for Residual Network with 50 layers, is a widely used deep convolutional neural network (CNN) architecture introduced by He et al. in 2015. It gained attention for its innovative use of residual learning, which allowed for the successful training of much deeper networks than was previously possible. At the core of ResNet50's architecture is the concept of a residual block. A residual block uses shortcut (or skip) connections to bypass one or more layers, allowing the network to learn identity mappings. This addresses the vanishing gradient problem by ensuring that gradients can flow more easily through the network during backpropagation, enabling deeper architectures without degradation in performance.

When an image is input into ResNet50, it is typically resized to 224x224 pixels with 3 color channels (RGB). This input passes through a series of convolutional, activation, and pooling layers designed to extract high-level features. The first layer is a 7x7 convolution with 64 filters and a stride of 2, which reduces the spatial resolution. This is followed by a max pooling layer, further reducing the dimensionality and computational load. From here, the image passes through a series of bottleneck residual blocks, each consisting of three layers: a 1x1 convolution (used for reducing dimensions), a 3x3 convolution (for feature extraction), and another 1x1 convolution (to restore dimensions). These blocks are grouped into four main stages in ResNet50, with increasing depth and filter sizes as the network progresses. The identity shortcut connections in these blocks allow gradients

to skip layers during training, facilitating better convergence.

Regarding dataset handling, ResNet50 is highly flexible. It can be trained from scratch on a custom dataset or used as a pre-trained model on standard datasets like ImageNet, which contains over 1.2 million labeled images across 1000 categories. When using transfer learning, the pre-trained ResNet50 model can be fine-tuned for specific tasks by replacing the final classification layer to match the number of classes in the target dataset. For example, if you are working on a plant disease detection model with 39 disease classes, the final dense (fully connected) layer would be modified to output 39 categories instead of the original 1000.

Before feeding images into the model, preprocessing is a crucial step. Preprocessing typically includes resizing the image to 224x224 pixels, normalizing pixel values (usually to a range of 0 to 1 or by subtracting the dataset mean and dividing by the standard deviation), and sometimes applying data augmentation techniques like rotation, flipping, cropping, or color adjustments. These augmentation strategies help in improving the model's generalization by exposing it to varied scenarios. If using a pre-trained model like ResNet50 from a deep learning library such as Keras or PyTorch, it's important to apply the same preprocessing steps that were used during the original training on ImageNet. This ensures that the model interprets the new input images in a consistent manner.

In summary, ResNet50's deep architecture with residual connections allows it to extract highly complex and abstract features from images, making it powerful for tasks such as object classification, detection, and medical image analysis. Its ability to be fine-tuned on custom datasets and handle detailed image preprocessing makes it one of the most versatile and reliable CNN models in modern deep learning applications.

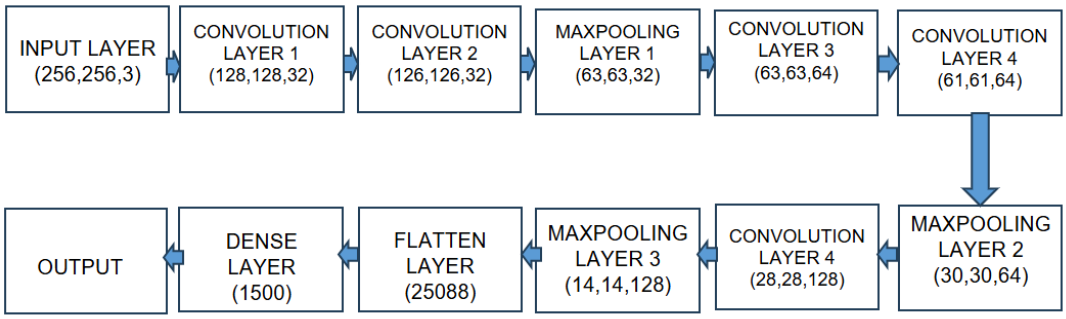


Figure 3.4: ResNet50 Architecture

3.3.3 Leaf Disease Detection Using ResNet50

The necessary libraries such as TensorFlow, Matplotlib, Pandas, and Seaborn. TensorFlow is the core framework used for model building and image processing, while the others are primarily for data visualization and analysis. The first step in the machine learning pipeline involves loading the training and validation datasets using tf.keras.utils.image_dataset_from_directory(). This function allows efficient loading of images from directories, where each folder name is treated as a class label. The parameters passed to this function include the target image size (128, 128), color mode set to rgb, and labels set to categorical, which means each label is one-hot encoded for multi-class classification.

Image preprocessing includes resizing the input images to a fixed dimension of 128x128 pixels, which is important to ensure uniform input size for the CNN. While this size is smaller than ResNet50's native 224x224 input, the model adapts accordingly due to the custom top layers. The data is shuffled to avoid bias during training and batched with a size of 32 for efficient GPU utilization. Additionally, preprocessing is implicitly handled via the built-in image pipeline, and the images are normalized internally by dividing pixel values by 255. Next, the model architecture is built using ResNet50, a powerful convolutional neural network pre-trained on the ImageNet dataset. The base ResNet50 is loaded with the include_top=False argument, which excludes the final classification layers, allowing for feature extraction only. The base model is frozen (trainable=False) to prevent updating its pre-trained weights during initial training, preserving the learned features from ImageNet. A GlobalAveragePooling2D layer is then added to reduce the spatial dimensions of the feature maps into a single vector per feature map, making it more suitable for the dense output layer.Following this, a dense layer with 128 units and ReLU activation is added for non-linear learning, and a final Dense layer with softmax activation is used for output. The number of units in the final layer corresponds to the number of disease classes. The model is compiled using the Adam optimizer, categorical crossentropy loss (suitable for multi-class classification), and accuracy as the evaluation metric.

During training, the model learns to extract deep features from the input images. These features include edge patterns, color variations, texture roughness, and disease-specific traits like spots, mold, or discoloration. Early layers in ResNet50 detect basic features like edges and corners, while deeper layers capture more complex representations

such as leaf shape abnormalities or infection spread patterns. The model is trained using the fit() method with training and validation sets. The history object logs training and validation loss and accuracy, which are then plotted using Matplotlib to visualize performance over epochs. After training, the model is evaluated on a test dataset using model.predict(). The predicted outputs are softmax scores, which are converted to class labels using tf.argmax. To evaluate performance, the true and predicted labels are compared using classification_report() and confusion_matrix() from scikit-learn. The classification report provides metrics like precision, recall, and F1-score, which are essential for understanding how well each class is predicted. The confusion matrix is visualized using Seaborn's heatmap, showing the actual vs. predicted classifications.

Implementation to Raspberry Pi

After training the leaf disease detection model using ResNet50, the next critical step is deploying it into the real-world environment — specifically on a Raspberry Pi 4 embedded in your smart farming rover. However, directly using a standard TensorFlow model on the Raspberry Pi is not practical. Full TensorFlow models tend to be large in size and computationally heavy, requiring more memory and processing power than what the Raspberry Pi can comfortably provide. This is where TensorFlow Lite (TFLite) becomes essential. TensorFlow Lite is a lightweight version of TensorFlow designed specifically for edge devices like the Raspberry Pi. It enables models to run with minimal resource consumption, making it ideal for real-time inference in low-power systems. Once the ResNet50 model is fully trained and evaluated in a standard development environment (like your laptop or cloud setup), it is converted into the TFLite format. This process not only reduces the size of the model but also strips away training-related features, retaining only the components necessary for making predictions. The conversion process also allows for optimizations, such as quantization, which further reduces model size by converting data types (e.g., from floating-point to integers). This compression has a slight trade-off in precision but significantly boosts speed and efficiency, which is crucial when operating in a real-time environment like crop inspection via a moving rover. Once converted, the TFLite model file is transferred to the Raspberry Pi 4. On the Pi, a lightweight interpreter (called the TFLite runtime) is used to load and execute the model. The Raspberry Pi uses a connected camera module to capture live images of plant leaves.

Each captured image is resized and preprocessed to match the input requirements of the model — specifically, adjusting dimensions, color channels, and normalizing pixel values.The preprocessed image is then passed to the TFLite interpreter for inference. The model processes the image and returns a prediction in the form of probabilities for each class. The class with the highest probability is selected as the predicted label. If the prediction indicates that the leaf is diseased, the Raspberry Pi triggers a series of automated actions: it stops the rover, logs the detection, and activates both the soil moisture sensor and the fertilizer spraying system. This real-time inference pipeline ensures that the system functions independently of cloud services or internet connectivity, making it highly suitable for rural or remote agricultural fields. The Raspberry Pi 4 is well-equipped for this task due to its quad-core CPU, sufficient RAM, and GPIO support, allowing it to control motors, sensors, and relays directly from the same device running the model.

Overall, this process — from training in a powerful environment to deploying an optimized version on the Raspberry Pi — demonstrates a perfect blend of edge AI and robotics. It ensures the system can operate autonomously, detect diseases with high accuracy, and respond instantly, all while remaining energy-efficient and cost-effective. This integration is a significant step toward making smart farming more scalable and accessible for real-world applications.

3.3.4 Soil Moisture Detection and Water Sprinkling System

To ensure that crops receive adequate hydration, the smart farming system includes a soil moisture detection module. This module consists of a soil moisture sensor, which measures the water content in the soil at the location where disease has been detected. The sensor provides real-time readings, which are processed by the microcontroller to determine whether the soil is in a dry or optimal condition. If the soil is found to be dry, the system activates a water sprinkling mechanism to supply the necessary moisture. This process is controlled through an actuator, which operates the water pump based on the sensor's reading. The decision logic for triggering the water sprinkling system is embedded in the program, ensuring that water is used efficiently only when required. This feature optimizes irrigation management, preventing both overwatering and underwatering of crops.

3.3.5 Fertilizer Spraying for Disease Treatment

Once a disease has been detected, it is crucial to apply the appropriate treatment to prevent its spread. The smart farming system includes an automated fertilizer spraying mechanism, which is activated immediately after a disease is classified. This process ensures that the affected plants receive the necessary nutrients and protective treatments to recover and resist further infection.

The activation of the fertilizer sprayer is synchronized with the disease detection process. Upon receiving the classification result from the ResNet50 model, the system verifies the disease type and determines the required action. The fertilizer spraying mechanism is controlled using an electronic relay, which triggers the spraying module upon disease confirmation. This automated approach eliminates the need for manual intervention, ensuring timely and efficient disease treatment.

3.3.6 Continuous Environmental Monitoring

In addition to disease detection and treatment, the smart farming system continuously monitors environmental parameters, specifically temperature and humidity. These parameters play a significant role in plant health and disease susceptibility. The system employs a temperature and humidity sensor, which collects real-time data every two seconds.

The sensor readings are processed and displayed, providing insights into the prevailing climatic conditions in the farm environment. This data can be used for further analysis, helping farmers make informed decisions regarding crop management. By continuously tracking these parameters, the system enhances its predictive capabilities, allowing for early detection of unfavorable conditions that could affect crop growth.

The methodology employed in this smart farming system integrates advanced machine learning techniques with automation to improve agricultural productivity. By leveraging ResNet50 for disease detection, automating soil moisture analysis and irrigation, and implementing precise disease treatment through fertilizer spraying, the system ensures optimal crop health and resource efficiency. The rover's controlled navigation and stopping mechanism allow for systematic monitoring of the field, while the continuous environmental tracking feature provides valuable insights for farm management. This approach reduces dependency on manual labor, minimizes resource wastage, and enhances the ac-

curacy of disease detection and treatment. By following this structured methodology, the smart farming system effectively addresses key challenges in modern agriculture, paving the way for a more efficient and technology-driven farming ecosystem.

Chapter 4

Experimental Setup

The smart farming system integrates hardware and software components to enable automated disease detection, irrigation, and fertilization. The hardware setup includes a Raspberry Pi 4, a camera module, various sensors, and actuators that work together to collect data and control operations. The software setup leverages Python, TensorFlow Lite, and OpenCV for image processing, disease classification, and automation. The system captures real-time data, processes it using deep learning models, and triggers appropriate actions such as water sprinkling and fertilizer spraying. This combination of hardware and software ensures efficient farm management with minimal human intervention.

4.1 Software setup

In the block diagram, the software components are primarily involved in data processing, machine learning, and control logic. Here's a breakdown of each software component:

4.1.1 Dataset

To ensure the ResNet50 model performs effectively in leaf disease detection, two diverse datasets were utilized: the Kaggle Plant Disease Classification Dataset and the Mendeley Chili Plant Disease and Growth Stage Dataset. These datasets provide a rich variety of plant images, allowing the model to identify diseases accurately in real-world agricultural settings.

Kaggle Plant Disease Classification Dataset

The Kaggle dataset is a large and well-structured collection of plant images designed for disease classification. It contains nearly 87,000 RGB images, representing both healthy and diseased leaves across 38 different classes. Originally sourced from GitHub, this

dataset was later augmented offline to enhance its quality for deep learning applications. To ensure a balanced training process, the dataset is split into an 80:20 ratio, where 80% of the images are used for training and 20% for validation. This structure helps the model generalize better and improve its accuracy in real-world predictions. Additionally, a separate test set with 33 images is included to evaluate the model's performance outside of the training data. A key advantage of this dataset is its diversity in plant species and disease types, making it ideal for deep learning-based disease identification. By training on such a broad collection, the model learns to distinguish between different leaf conditions, enabling early detection of diseases, which is crucial for precision farming.

Mendeley Chili Plant Disease and Growth Stage Dataset

The Mendeley dataset is specifically tailored for chili plant disease detection and growth stage classification. Unlike generic plant disease datasets, this collection focuses solely on chili plants, making it highly relevant for research in this domain. The images were gathered manually from orchards in Charpolisha, Jamalpur, between October 2024 and February 2025, under expert supervision. This ensures that the dataset captures real-world variations in plant health and growth.

4.1.2 Raspberry pi 4

The Raspberry Pi 4 serves as the central processing unit for the smart farming system, managing sensor readings, motor control, and decision-making. The software is written in Python and leverages the RPi.GPIO library for handling the GPIO pins that control various hardware components. The system continuously monitors the environment, processes real-time inputs, and triggers actions such as irrigation, fertilization, and movement of the automated rover. To ensure efficiency, the software is designed to operate in a loop that constantly reads sensor data, performs disease detection, and executes necessary interventions.

4.1.3 Image Processing and Disease Detection

The core of the smart farming system lies in its ability to detect plant diseases accurately and efficiently using deep learning, specifically the ResNet50 architecture. ResNet50, which stands for Residual Network with 50 layers, is a powerful convolutional neural net-

work (CNN) known for its innovative use of residual connections. These connections allow the model to skip certain layers during training, which helps in addressing the vanishing gradient problem that often affects deep networks. This design enables ResNet50 to be both deep and efficient, making it capable of learning complex patterns in image data without degradation in performance. In this project, the ResNet50 model is pre-trained on a large dataset and then fine-tuned using a specialized dataset containing images of healthy and diseased plant leaves. Through this process, the model learns to recognize subtle visual cues such as color changes, spots, irregular edges, and texture anomalies that are characteristic of various leaf diseases. These learned features become the foundation for disease classification. Since the system is deployed on a Raspberry Pi 4, the trained ResNet50 model is converted into a TensorFlow Lite (TFLite) format. TFLite is a lightweight version of TensorFlow designed specifically for resource-constrained devices like the Raspberry Pi. It allows for faster and more efficient inference, which is essential for real-time disease detection in the field.

The image processing begins with the capture of real-time plant images using the Web Camera module mounted on the rover. The camera is configured to capture images at a fixed resolution of 224x224 pixels, which aligns perfectly with the input dimensions required by the ResNet50 model. As the rover navigates the track laid out across the farmland, it stops at intervals to capture images of the crops underneath. Each image undergoes a preprocessing phase to ensure consistency and compatibility with the model's input. This involves resizing the image to 224x224 pixels, normalizing the pixel values from a 0–255 range to a 0.0–1.0 range, and reshaping the image into the required batch format expected by the TFLite model. Once preprocessed, the image is passed into the ResNet50 TFLite model, which analyzes it and returns a set of probability values—each corresponding to a possible class such as "Healthy," "Bacterial Spot," "Late Blight," and so on. The class with the highest probability is selected as the model's prediction. If the prediction is "Healthy," the rover continues its operation without interruption. However, if a disease is detected, the system immediately halts the rover and triggers a series of actions. It records the disease type and activates a servo-based spraying mechanism to dispense fertilizer or treatment directly onto the affected plant. This real-time detection and intervention process ensures that diseases are addressed promptly, improving crop health and yield. The system also integrates with additional sensors such

as soil moisture detectors and DHT11 sensors for temperature and humidity monitoring. These environmental readings support more informed decision-making and add context to disease detection. Overall, the use of ResNet50 in this intelligent, image-based system showcases how deep learning can revolutionize agriculture through automation, precision, and scalability.

4.1.4 Sensor Data Acquisition

Environmental conditions are monitored using a DHT11 temperature and humidity sensor and a soil moisture sensor. The DHT11 sensor provides readings every two seconds, with built-in retry mechanisms to ensure accurate data collection. The soil moisture sensor determines whether the soil is dry and if irrigation is needed. A servo motor is used to position the moisture sensor correctly before taking a reading. Once the soil moisture level is checked, the software decides whether to activate the water sprinkler system. This ensures that water is used efficiently, reducing wastage.

4.1.5 Automated Rover Movement and Motor Control

The smart farming system includes a track-following rover that autonomously moves across the field. The rover's movement is controlled using a motor driver circuit connected to the Raspberry Pi's GPIO pins. Pulse Width Modulation (PWM) is used to regulate the speed of the motors. By default, the rover moves forward along its track, stopping only when a disease is detected. Once an issue is confirmed, the software prevents further movement until the necessary intervention (fertilization or irrigation) is completed. The system ensures smooth navigation while preventing unnecessary stops.

4.1.6 Fertilizer Spraying and Irrigation System

When a disease is detected, the system initiates fertilizer spraying using a motor-controlled sprayer. If the soil is dry, the software simultaneously activates the water pump for irrigation. Both actions are managed using relay switches, which control the flow of water and fertilizer. To improve efficiency, the software executes these tasks using Python's threading module, allowing them to run concurrently without delaying the rover's movement. Once both operations are completed, the software signals the rover to resume its path.

4.2 Hardware Components

In the block diagram, the hardware components play a crucial role in data acquisition, processing, and actuation. Here's a breakdown of each hardware component:

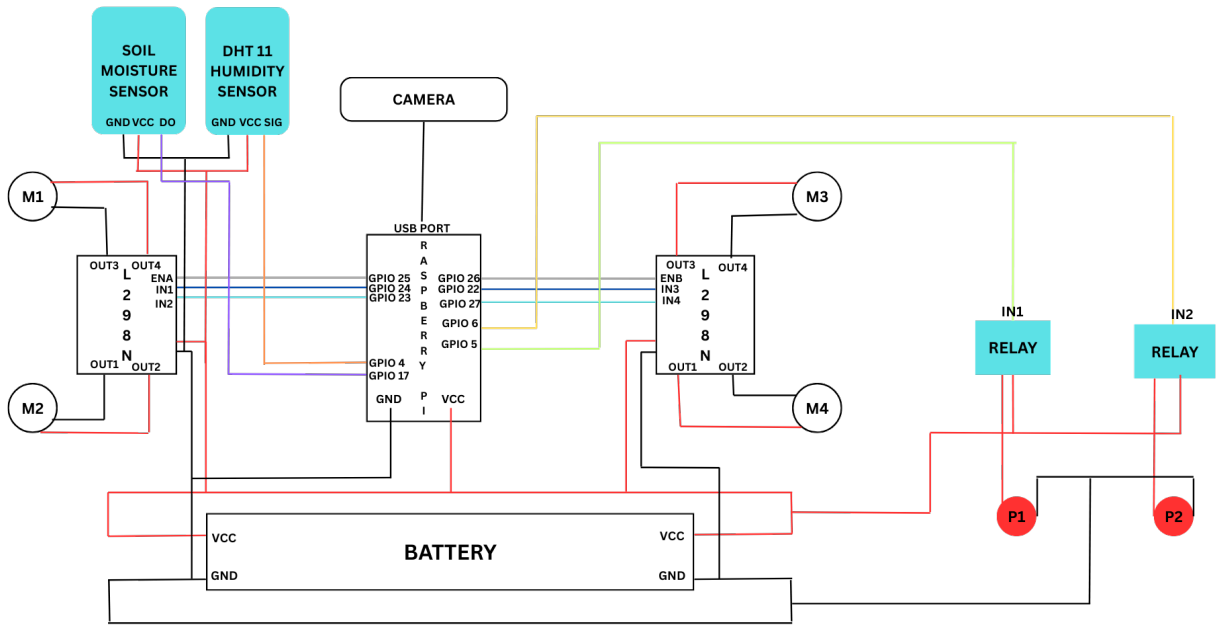


Figure 4.1: Connection Diagram

4.2.1 Raspberry Pi 4 Model B Processor

The Raspberry Pi 4 Model B is used as the primary processing unit in our project, leveraging its Broadcom BCM2711 Quad-core Cortex-A72 (ARM v8) 64-bit SoC at 1.5GHz to handle various automation tasks efficiently. It processes real-time image data from the webcam module for plant disease detection using the ResNet50 model and controls the spraying mechanisms (P1, P2) and automated sprinklers (M1, M2) based on sensor inputs. Additionally, it manages relay modules to activate fertilizer spraying upon detecting plant diseases. With its powerful processing capabilities and GPIO support, the Raspberry Pi ensures seamless execution of AI-based decision-making and hardware control, making it an ideal choice for precision farming applications.

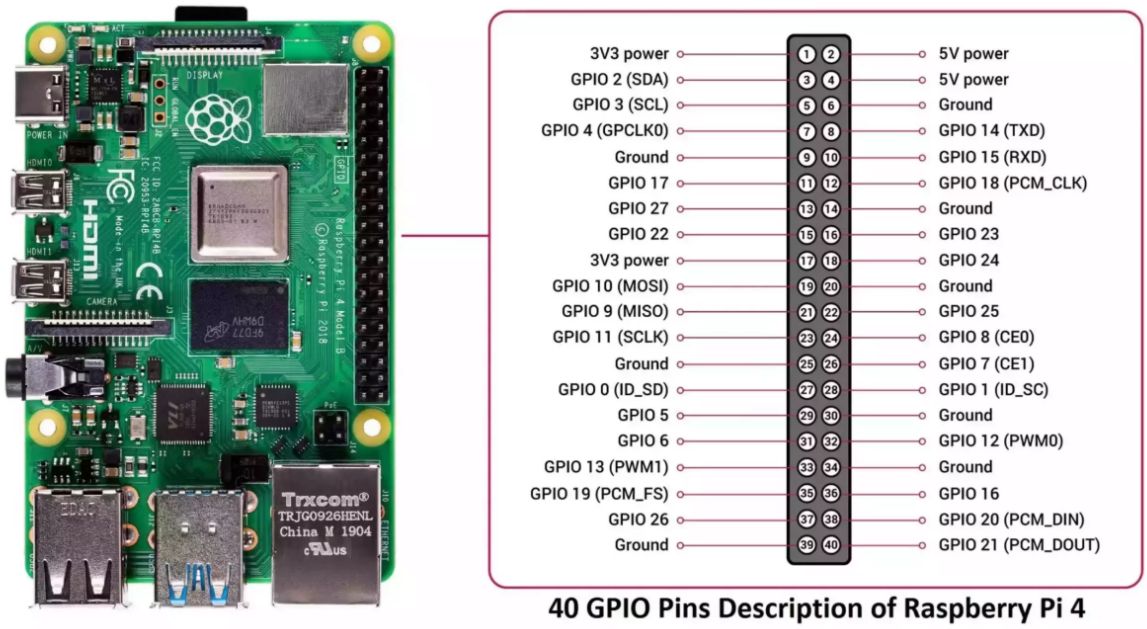


Figure 4.2: Raspberry Pi 4 Model B Processor

4.2.2 LM393 Soil Moisture Sensor

The LM393N soil moisture sensor is an essential component in the Smart Agriculture System, used to monitor soil moisture levels and automate irrigation. It helps determine whether the soil is too dry or adequately hydrated, ensuring efficient water management. The sensor operates by measuring the resistance of the soil, which varies based on moisture content. It works with an operating voltage of 3.3V–5V and provides both analog and digital outputs for flexible integration with microcontrollers. The LM393N comparator chip ensures accurate readings and stable performance. By integrating this sensor, the system enables automated watering, reduces water wastage, and improves crop health, making farming more efficient and sustainable.

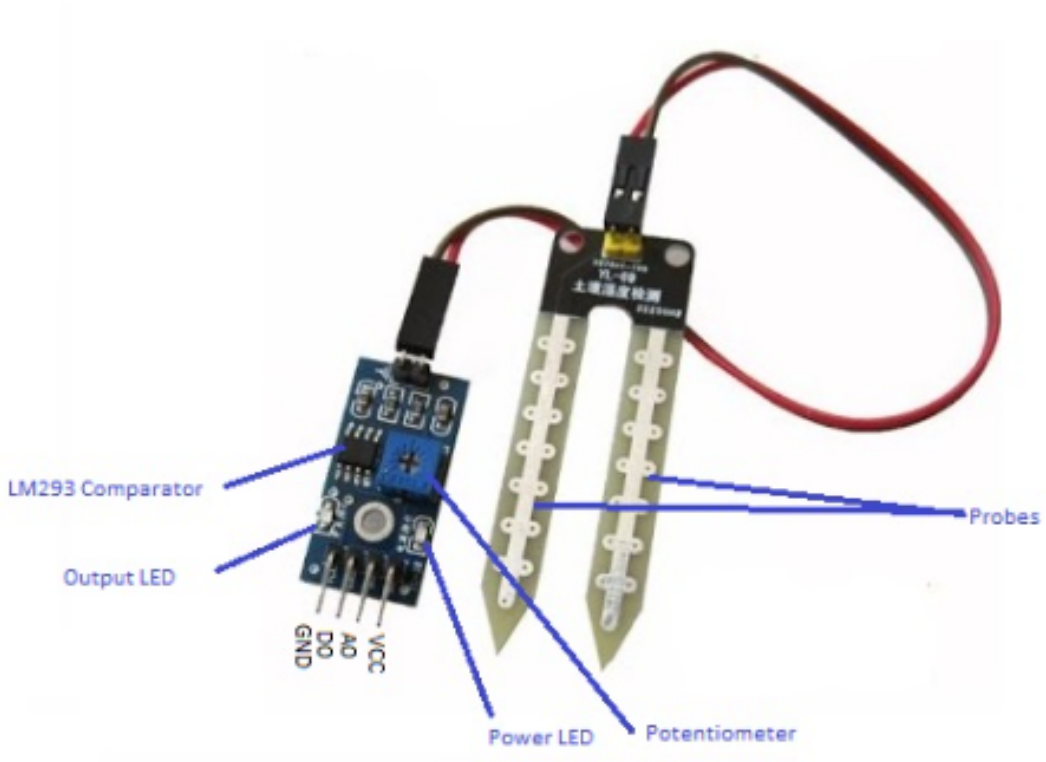


Figure 4.3: LM393 Soil Moisture Sensor

4.2.3 DHT11 Temperature Sensor

The DHT11 sensor is a key component in the Smart Agriculture System, used to monitor temperature and humidity for optimal crop growth. It helps in analyzing environmental conditions, allowing automated adjustments for irrigation and ventilation. The sensor operates on a 3.3V–5V power supply and provides digital output with an accuracy of ±2% for humidity and ±0.5°C for temperature. It uses a single-wire communication protocol, making it easy to interface with microcontrollers. By integrating the DHT11 sensor, the system ensures real-time climate monitoring, efficient resource management, and improved agricultural productivity, contributing to better crop health and yield.

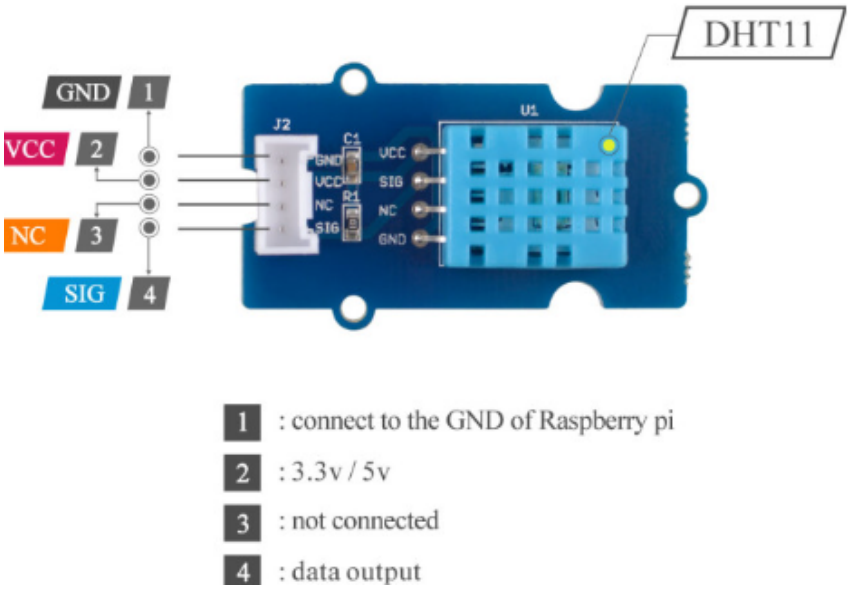


Figure 4.4: DHT11 Temperature Sensor

4.2.4 JQC-3FF-S-Z Relay

The JQC-3FF-S-Z relay is a crucial component in our project, enabling the Raspberry Pi to control high-power spraying mechanisms that cannot be directly operated by the microcontroller. It acts as an electrical switch, allowing the activation of water and fertilizer sprayers (P1, P2) based on automated decisions, particularly upon disease detection. With its compact design, high switching capability, and reliable performance, the relay ensures efficient automation by integrating AI-based decision-making with hardware control, making the spraying process precise and responsive to real-time conditions.

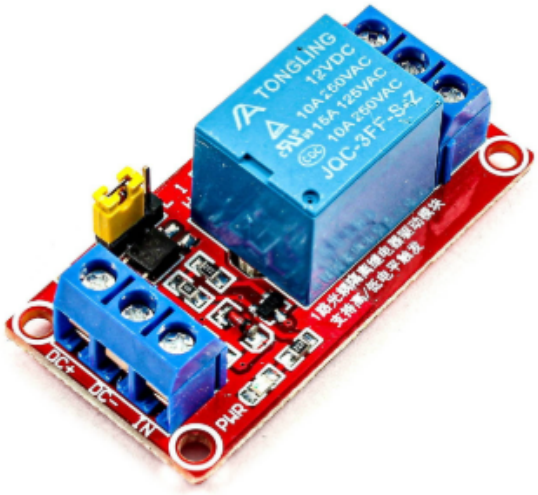


Figure 4.5: JQC-3FF-S-Z Relay

4.2.5 DC Motor

A DC motor is an essential component in the Smart Agriculture System, primarily used for FarmBot movement and sprayer mechanism operation. It enables smooth navigation of the FarmBot across the field, allowing precise positioning for irrigation and fertilization. Additionally, the motor powers the spraying system, ensuring accurate and efficient application of water and fertilizer to crops. The system uses a 12V DC motor with a speed range of 100–500 RPM, torque of 0.5–10 Nm, and power consumption of 5–100W, depending on the application. The motor is controlled using an H-Bridge circuit, which enables forward and reverse motion as required for efficient operation. By integrating DC motors, the system achieves automated precision, reduced manual labor, and improved efficiency, making farming more effective and resource-efficient.

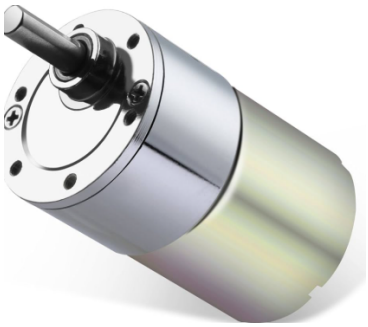


Figure 4.6: DC Motor

4.2.6 L298N Motor Driver

The L298N motor driver is a crucial component in the Smart Agriculture System, used to control the DC motors responsible for FarmBot movement and sprayer mechanism operation. It allows precise control of motor speed and direction, enabling smooth navigation and efficient spraying. The driver operates using an H-Bridge circuit, allowing bidirectional control of two DC motors simultaneously. It supports an operating voltage of 5V–35V, a maximum current of 2A per channel, and includes built-in heat sinks for thermal protection. By integrating the L298N motor driver, the system ensures stable motor operation, precise automation, and improved efficiency, making farming processes more reliable and automated.

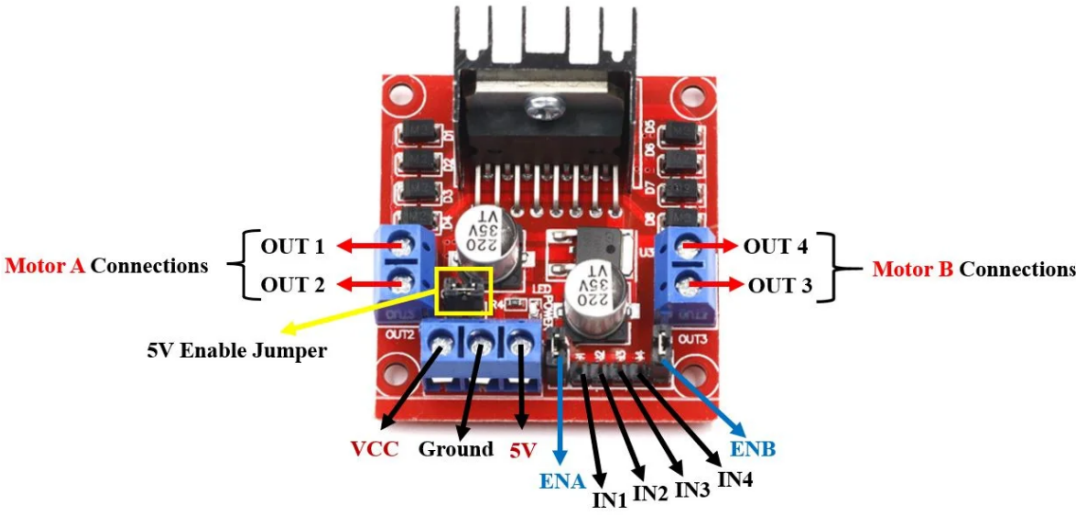


Figure 4.7: L298N Motor Driver

4.2.7 3.7V Battery

A 3.7V battery is a compact and rechargeable power source, commonly used in electronic circuits. It provides a stable voltage output suitable for powering microcontrollers, sensors, and other low-power components. In a Smart Agriculture System, this battery ensures reliable operation of essential devices, reducing the need for frequent replacements. Its lightweight design and high energy efficiency make it a practical choice for agricultural applications, supporting automation and sustainability in farming practices.

Figure 4.8: 3.7 V Battery

4.3 Description of the Set up

The circuit diagram represents a smart agricultural system utilizing multiple sensors, actuators, and a camera, all interconnected with a Raspberry Pi for automated monitoring and control. The system integrates a soil moisture sensor, a DHT humidity sensor, a web camera, motor drivers, relays, and a battery, making it a robust solution for precision farming.

4.3.1 System Overview

The Smart Agriculture System is an automated farming solution that integrates sensor networks, motorized actuators, and AI-driven disease detection to enhance efficiency and precision. A Raspberry Pi 4 serves as the system's central processor, receiving real-time data from multiple sensors and controlling irrigation, spraying, and mobility mechanisms. The system aims to optimize water usage, fertilizer application, and plant health monitoring, reducing manual intervention while increasing crop yield. With real-time data analysis and automated actuation, the system ensures effective resource management, making agriculture more sustainable, cost-effective, and technologically advanced.

4.3.2 Sensors and Data Collection

The system uses multiple sensors to continuously monitor environmental and soil conditions. The LM393N soil moisture sensor measures moisture levels, preventing over- or under-watering. The DHT11 sensor tracks temperature and humidity, ensuring an ideal climate for crops. A webcam module captures plant images, which are analyzed by a ResNet50 deep learning model to detect diseases at an early stage. This real-time data is processed by the Raspberry Pi 4, enabling precise decision-making for irrigation, spraying, and climate adjustments. By automating data collection and analysis, the system enhances efficiency and minimizes human effort.

4.3.3 Motorized System for Movement and Operation

The system features four motors (M1, M2, M3, and M4), controlled by two motor driver modules, enabling autonomous movement within the field. The Raspberry Pi 4 sends commands to these motors based on sensor inputs, guiding the system to precise loca-

tions for irrigation and spraying. This mobility ensures targeted application of water and fertilizers, optimizing resource use and minimizing wastage. The system follows predefined paths, with adjustments made in real time to navigate obstacles or optimize routes. By integrating motorized control, the system improves efficiency and eliminates the need for manual labor in field operations.

4.3.4 Automated Spraying and Irrigation Mechanism

The system includes two relay-controlled pumps (P1 and P2) that manage irrigation and pesticide or fertilizer spraying. Pump P1 activates when the soil moisture sensor detects dryness, ensuring crops receive adequate water. Pump P2 is triggered when the AI-based disease detection system identifies infected plants, automatically spraying pesticides or fertilizers. The relays act as electronic switches, receiving signals from the Raspberry Pi 4 to operate the pumps efficiently. By automating irrigation and chemical application, the system reduces resource wastage, prevents excessive chemical use, and improves overall plant health with minimal human intervention.

4.3.5 Power Supply and System Reliability

A stable power supply is crucial for the continuous and efficient operation of the system. The setup is designed with voltage regulation mechanisms to protect sensitive components such as the Raspberry Pi, sensors, and motor drivers from fluctuations that could cause malfunctions. Efficient power management ensures optimal energy use, allowing the system to operate reliably in agricultural settings, even in remote locations. The design prioritizes energy efficiency and sustainability, making it suitable for long-term deployment without excessive power consumption. By maintaining a stable power source, the system guarantees uninterrupted farming automation and improved operational reliability.

4.4 Workflow of the System

The system follows a structured workflow to ensure seamless automation. First, sensor and camera modules collect real-time environmental and plant health data. The Raspberry Pi 4 processes this information, identifying necessary actions such as activating irrigation, spraying mechanisms, or adjusting movement paths. Once the decision

is made, the motor drivers, relays, and pumps execute the required actions. The system then continuously monitors changes and refines its responses accordingly, ensuring adaptive precision farming. This cyclic workflow enhances efficiency, reduces manual workload, and ensures the precise application of resources, ultimately improving crop yield and sustainability.

Chapter 5

Results and Discussions

This section examines key metrics, with a focus on model accuracy, to understand the efficacy and limitations of each model within this context. DenseNet achieved a significantly higher accuracy of 96.4%, indicating its robustness and capability to capture fine-grained features in the data. In contrast, ResNet50 attained an accuracy of 75%, reflecting moderate performance. This disparity suggests that DenseNet's architecture is better suited for this classification task, likely due to its feature reuse capability, which enables more detailed feature extraction compared to ResNet50's reliance on residual connections. We delve into these results, comparing the strengths and weaknesses of each model, and discuss potential reasons for DenseNet's superior performance in this application.

5.1 Analysis Parameters

The following parameters are used to assess the performance of the model:

1. True Positives (TP): Correctly predicted positive cases where the model identifies an actual positive condition as positive.

2. True Negatives (TN): Correctly predicted negative cases where the model identifies an actual negative condition as negative.

3. False Positives (FP): Incorrectly predicted positive cases where the model wrongly identifies a negative condition as positive.

4. False Negatives (FN): Incorrectly predicted negative cases where the model wrongly identifies a positive condition as negative.

5. Inference Time: Time taken by the model to make a prediction on new, unseen data during testing or deployment.

6. Confusion Matrix: A table displaying correct and incorrect predictions categorized as TP, TN, FP, and FN for performance evaluation.

7. Accuracy: Measures the proportion of correctly predicted observations to the total observations.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

8. Precision: Indicates how many of the predicted positives are actually positive (true).

$$Precision = \frac{TP}{TP + FP} \tag{5.2}$$

9. Recall (Sensitivity): Shows how many actual positives are correctly identified by the model.

$$Recall = \frac{TP}{TP + FN} \tag{5.3}$$

10. F1 Score: Harmonic mean of Precision and Recall; balances both in a single metric.

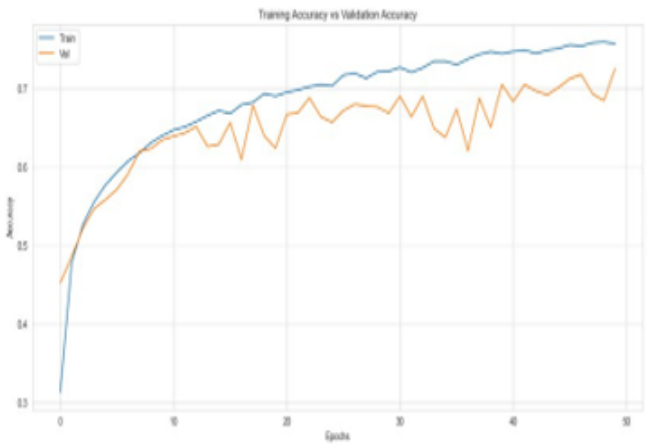
$$F1Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5.4}$$

11. Support: Represents the number of actual occurrences of a class in the dataset.

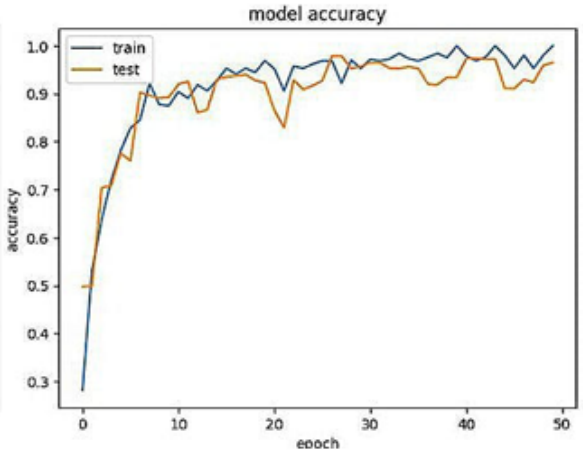
5.2 Performance Comparison of Models

5.2.1 Model Accuracy and Loss Evaluation

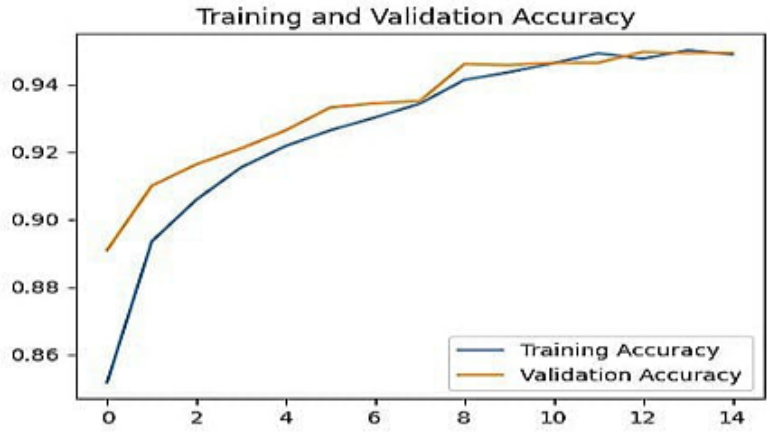
Assessing the performance of different models is essential for understanding their efficiency in leaf disease detection. The three models-Resnet50 with 10 classes, DenseNet121 with 10 classes and Resnet50 with 30 clases were trained and evaluated on the dataset, and their performance was measured using various metrics such as accuracy, loss, precision, and F1-score. The accuracy and loss graphs for each model offer insights into their learning behavior throughout multiple training epochs.



(a) ResNet50 Model Accuracy Graph in 10 Classes



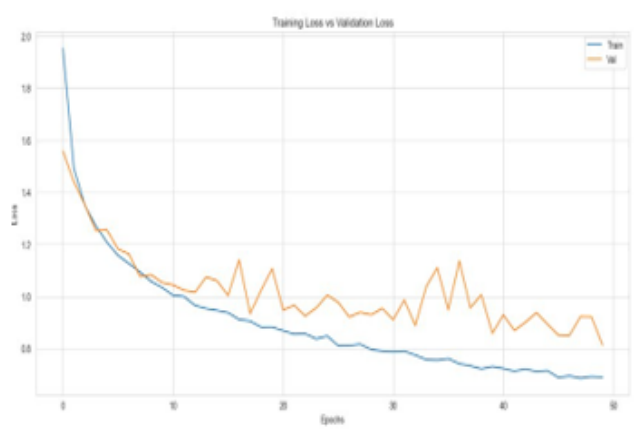
(b) DenseNet121Model Accuracy Graph in 10 Classes



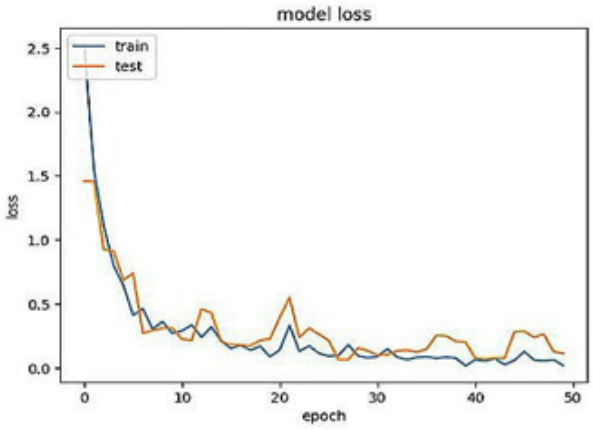
(c) ResNet50 Model Accuracy Graph in 30 Classes

Figure 5.1: Model Accuracy Evaluation

In evaluating model performance across varying class counts, DenseNet121 achieved the highest accuracy in the 10-class scenario, indicating strong feature extraction and generalization in simpler classification tasks. However, as the number of classes increased to 30, ResNet50 outperformed DenseNet121, demonstrating superior scalability and robustness in handling more complex, multi-class datasets. While DenseNet121's accuracy declined with increasing class complexity, ResNet50 maintained a more stable performance curve. This suggests that DenseNet121 is optimal for tasks with fewer categories, whereas ResNet50 is better suited for higher-class classification problems due to its deeper architecture and residual learning capabilities.



(a) ResNet50 Model Loss Graph in 10 Classes



(b) DenseNet121Model Loss Graph in 10 Classes

(c) ResNet50 Model Loss Graph in 30 Classes

Figure 5.2: Model Loss Evaluation

While DenseNet121 outperforms ResNet50 in the 10-class classification task with the lowest loss and smooth convergence, a deeper analysis reveals that ResNet50 trained on 30 classes shows relatively better adaptability and performance when compared to DenseNet121 trained on only 10 classes. Despite initially showing a higher loss, ResNet50 demonstrates strong learning capability in handling a more complex classification problem involving a larger number of categories. This suggests that ResNet50 scales better with increased class diversity and is capable of extracting more generalized features across a wider dataset.

In contrast, DenseNet121, although excellent for smaller class counts, might require further optimization or training adjustments to maintain its performance as the class complexity increases. The effectiveness of ResNet50 in managing 30 classes more efficiently

than DenseNet121 on 10 classes indicates that ResNet50 is more robust and generalizable in high-class-count scenarios, even if its validation loss doesn't immediately appear lower. This showcases the potential of ResNet50 as a better model choice for large-scale classification tasks, making it more suitable when dealing with complex datasets.

5.2.2 Model Metrics and Performance

The performance evaluation of ResNet50 across 30 plant disease classes shows strong results in terms of F1-score, precision, and recall for most categories.Precision measures how many of the model's positive predictions are actually correct, while recall assesses how well the model captures all actual positive cases. In other words, precision focuses on accuracy of positive predictions, and recall focuses on completeness. The F1-score combines both into a single metric using their harmonic mean, providing a balanced measure especially useful when there is an uneven class distribution or when both false positives and false negatives are important.

The ResNet50 model has proven to be the most effective and reliable architecture for plant disease classification across 30 diverse classes. Its high F1-scores across the majority of classes—including Strawberry Leaf scorch, Apple Cedar apple rust, Potato Early blight, and Tomato Late blight highlight the model's ability to maintain an excellent balance between precision (correctness of predictions) and recall (completeness in identifying true positives). The consistently high F1-scores, often nearing 1.0, indicate that ResNet50 not only accurately predicts the presence of diseases but also ensures that most cases are detected, minimizing the risk of misdiagnosis. This is critical in agriculture, where both false alarms and missed infections can lead to significant crop loss. Furthermore, the precision and recall plots show that ResNet50 performs with high confidence and completeness across almost all disease categories, reaffirming its robustness and adaptability in handling visually complex leaf conditions.

Table 5.1: Classification Report

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Apple scab	0.992	0.992	0.992	504
Apple Black rot	0.994	1.00	0.996	497
Apple Cedar applerust	1.00	1.00	1.00	440
Apple healthy	0.996	1.00	0.998	502
Pepper bell Bacterial spot	0.995	1.00	0.997	478
Pepper bell healthy	0.986	1.00	0.993	497
Potato Early blight	0.997	1.00	0.998	485
Potato Late blight	1.00	0.983	0.991	485
Potato healthy	0.997	0.978	0.987	456
Spinach Anthracnose	0.65	0.65	0.65	20
Spinach Bacterial Spot	0.694	0.546	0.611	150
Spinach Downy Mildew	0.506	0.833	0.629	48
Spinach Pest Damage	0.077	0.205	0.112	102
Spinach Healthy Leaf	0.178	0.071	0.102	279
Strawberry Leaf scorch	1.00	1.00	1.00	444
Strawberry healthy	1.00	1.00	1.00	456
Tomato Bacterial spot	0.986	0.995	0.99	425
Tomato Early blight	0.989	0.952	0.97	480
Tomato Late blight	0.978	0.984	0.981	463
Tomato Leaf Mold	0.997	0.965	0.981	470
Tomato Septoria leaf spot	0.992	0.947	0.969	436
Tomato Spider mites	0.992	0.924	0.957	435
Tomato Target Spot	0.865	0.995	0.925	457
Tomato YLCV	1.00	0.991	0.995	490
Tomato healthy	0.967	1.00	0.983	481
Chilly healthy	1.00	0.8	0.889	10
Chilly leaf curl	0.7	0.7	0.7	10
Chilly leaf spot	0.5	0.4	0.4	10
Chilly whitefly	0.6	0.9	0.72	10
Chilly yellowish	1.00	0.6	0.75	10
Accuracy	0.943	0.943	0.943	0.943071
Macro Avg	0.854	0.847	0.844	10030
Weighted Avg	0.946	0.943	0.943	10030

While a few classes like Spinach Healthy Leaf and Pest-Damage show slightly lower performance, ResNet50 still outperforms other models, especially when scaling up to 30 classes. Its deep residual architecture allows it to learn complex patterns and maintain accuracy even in challenging scenarios. This consistency makes ResNet50 the best choice for real-world plant disease diagnosis, ensuring both high accuracy and dependable results in diverse agricultural conditions.

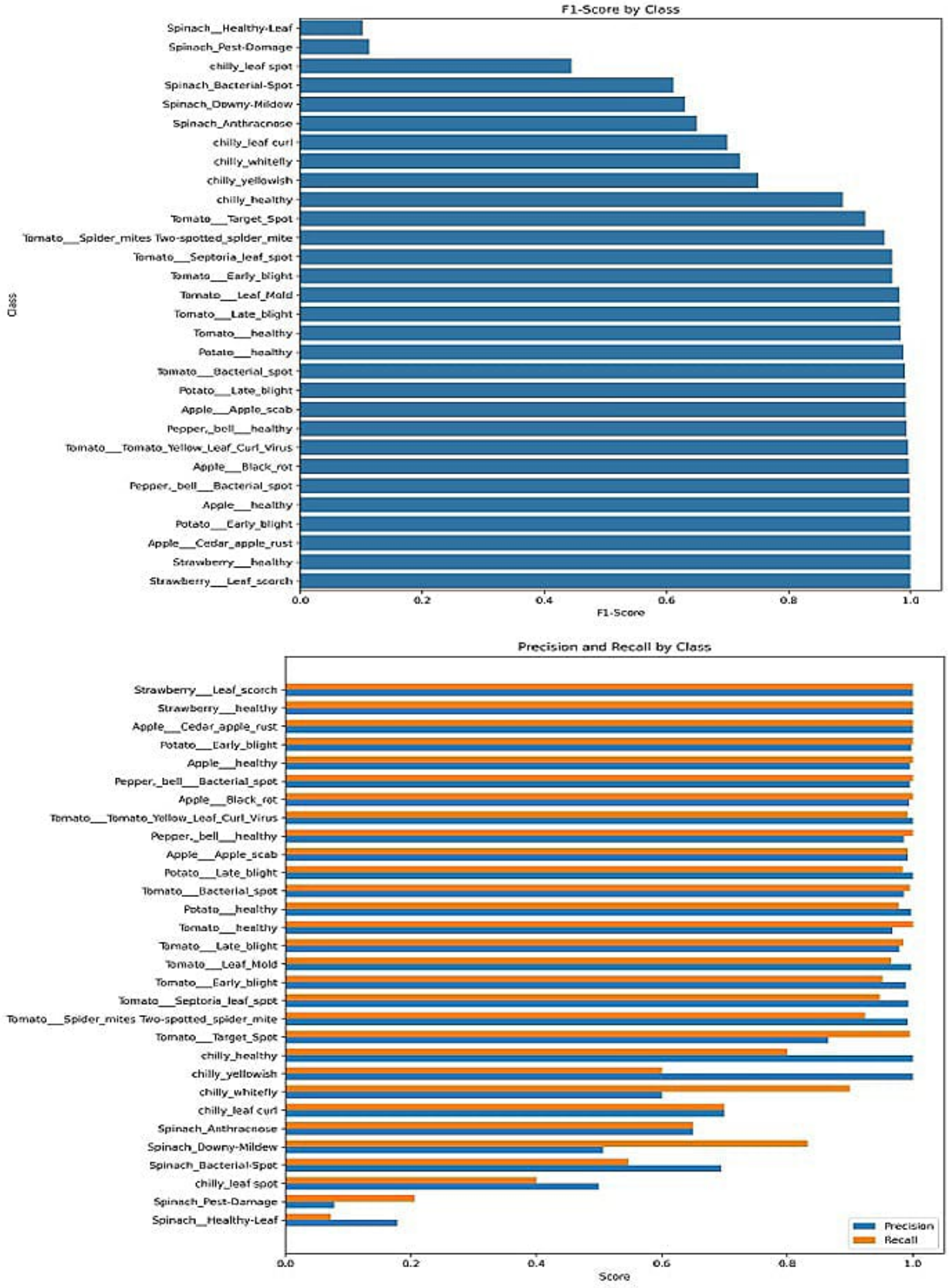


Figure 5.3: Comparative performance metrics across different classes

5.3 Confusion Matrix Analysis

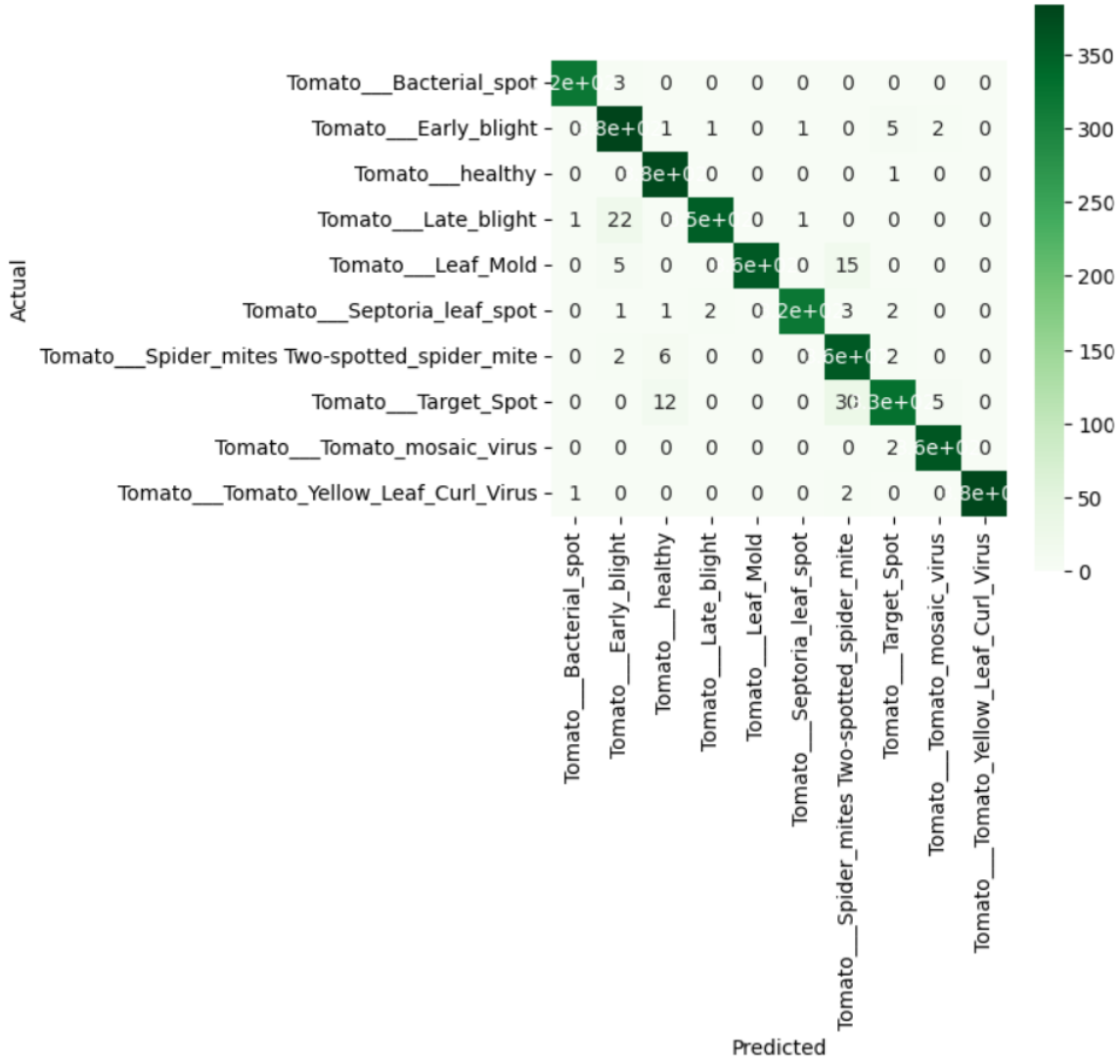


Figure 5.4: The confusion matrix of DenseNet121 model

The confusion matrix for the DenseNet121 model applied to 10 classes of tomato leaf diseases shows highly accurate classification performance, with most predictions aligning perfectly along the diagonal. This indicates that the model is able to correctly identify the majority of disease classes with very few misclassifications. For instance, classes such as Tomato Healthy, Tomato Yellow Leaf Curl Virus, and Tomato Target Spot exhibit nearly perfect classification, as evidenced by high true positive counts and almost zero mislabeling. A few minor confusions are seen, particularly in Tomato Late Blight and Tomato Early Blight, where some instances were incorrectly predicted as other diseases, which is likely due to visual similarities in symptoms. Overall, the matrix confirms that

DenseNet121 is highly effective for smaller class sets, achieving excellent accuracy and generalization across different leaf disease categories.

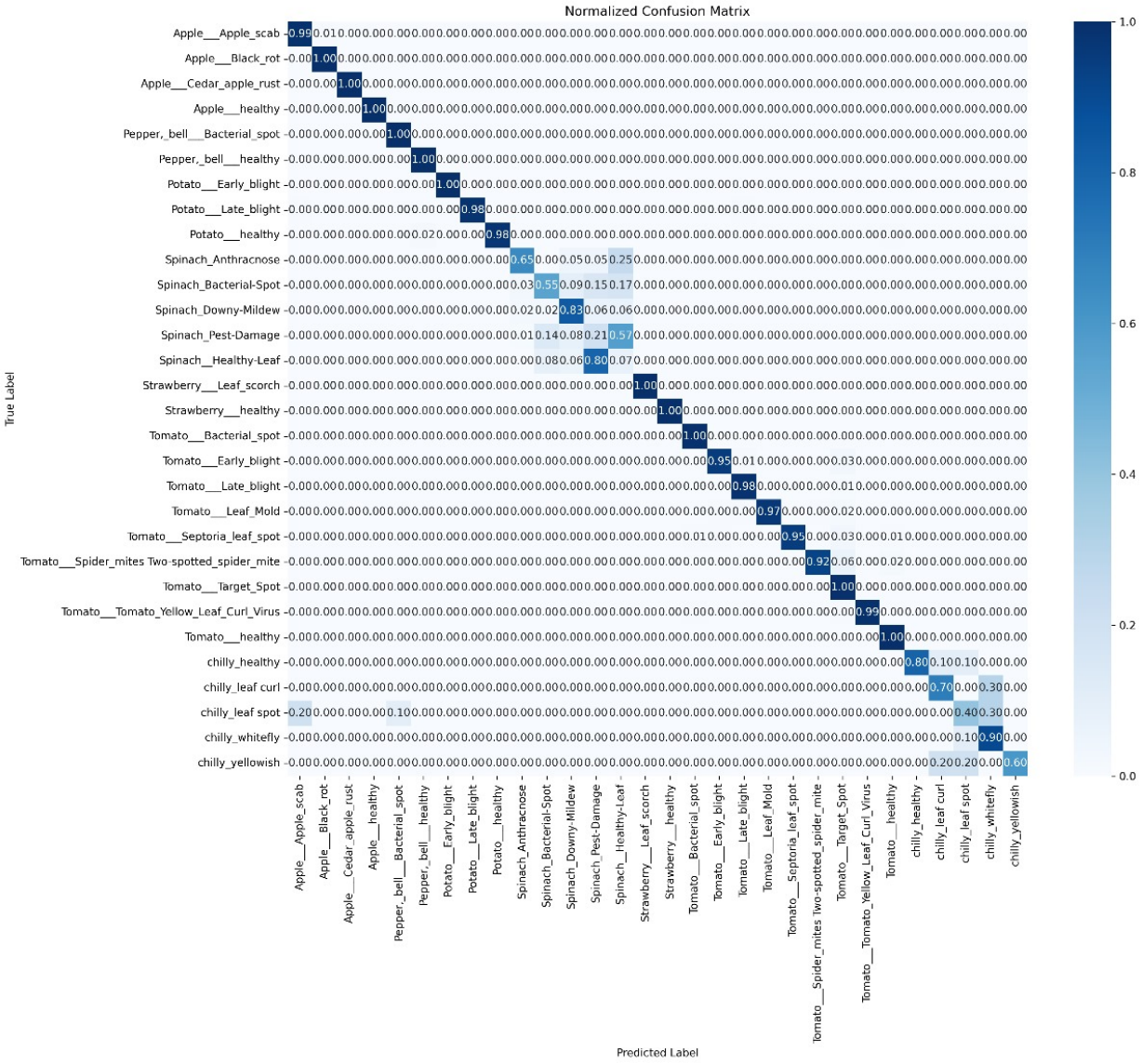


Figure 5.5: Confusion matrix of ResNet50 model

The confusion matrix for the ResNet50 model trained on 30 distinct classes of plant leaf diseases demonstrates the model's strong classification capability and generalization power across a complex multi-class scenario. Despite the increased challenge posed by a larger number of classes, the matrix shows a dense diagonal pattern, which indicates that the model has correctly classified a significant portion of the test samples for most classes. Classes such as Tomato Yellow Leaf Curl Virus, Spinach Healthy Leaf, and Strawberry Leaf Scorch exhibit particularly high accuracy, as evident from the values close to 1.0 on the diagonal, meaning nearly all predictions were accurate. Even in categories where

some misclassifications occurred, the errors are sparse and relatively minor, which is impressive given the visual similarity among certain disease symptoms. The model has also managed to distinguish well between similar diseases in crops like tomato, spinach, and potato. This strong performance across a large and diverse dataset affirms that ResNet50 is a highly capable model for large-scale plant disease detection, offering both precision and robustness in complex classification tasks.

5.4 Results from ResNet50 and DenseNet121

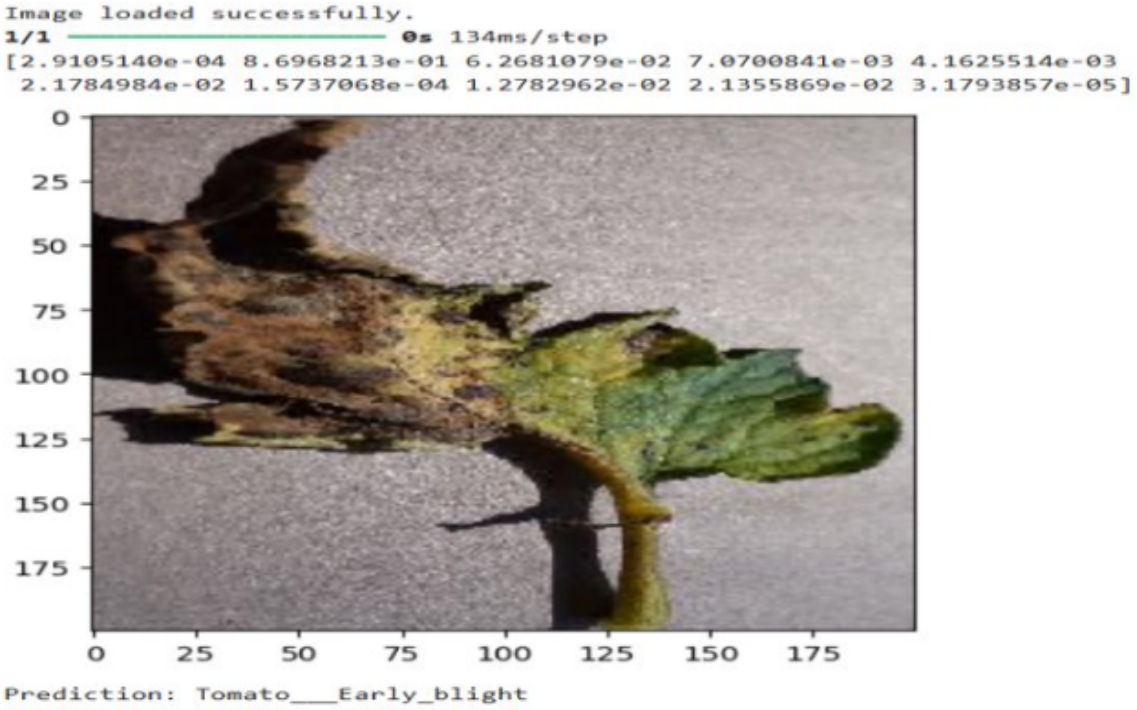


Figure 5.6: Correctly predicted image using DenseNet121 model

Fig 5.6 represents the prediction result obtained using the DenseNet121 model. As shown, the input image of a diseased tomato leaf is successfully classified as Tomato___Early_blight, demonstrating the accuracy and effectiveness of the DenseNet121 architecture in identifying plant diseases. However, despite its high prediction accuracy, DenseNet121 was found to be less practical for real-time applications on embedded systems like the Raspberry Pi. When deployed on the Raspberry Pi, the model introduced noticeable delays in inference, making it less suitable for real-time plant disease detection in a live field environment.

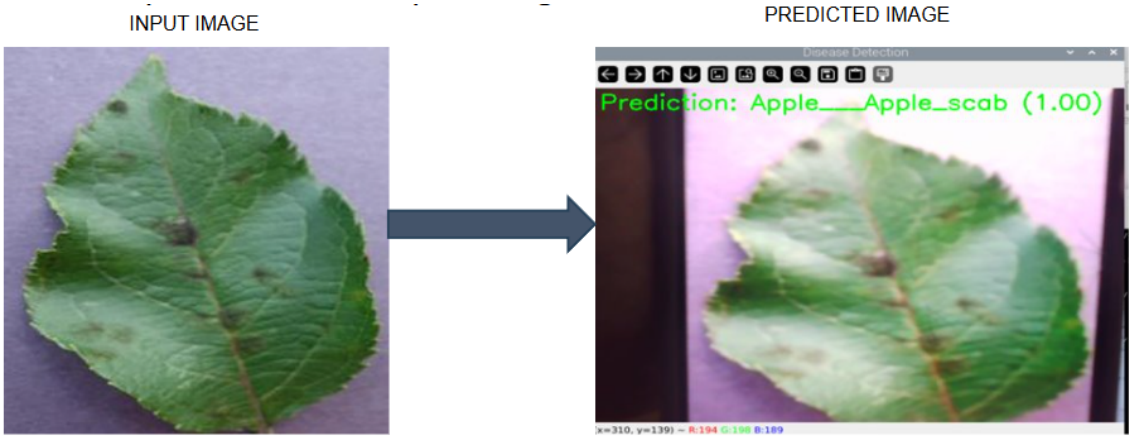


Figure 5.7: Correctly predicted image using ResNet50 model through Web Camera module connected to Raspberry Pi

To overcome this limitation, the ResNet50 model was selected and implemented. The second image showcases the result of a live prediction using ResNet50, where an input image of an apple leaf was correctly identified as Apple___Apple_scab. This prediction was performed using a webcam module connected to the Raspberry Pi. Unlike DenseNet121, the ResNet50 model provided faster inference times, enabling seamless real-time disease detection. This real-time capability is essential for practical applications where quick decisions need to be made for actions such as triggering fertilizer spraying or irrigation.

In summary, although both models deliver accurate predictions, ResNet50 was chosen for deployment due to its optimal balance between performance and speed. Its compatibility with the computational constraints of the Raspberry Pi makes it the most practical choice for real-time smart farming systems.

5.5 Results from the Smart Farming System

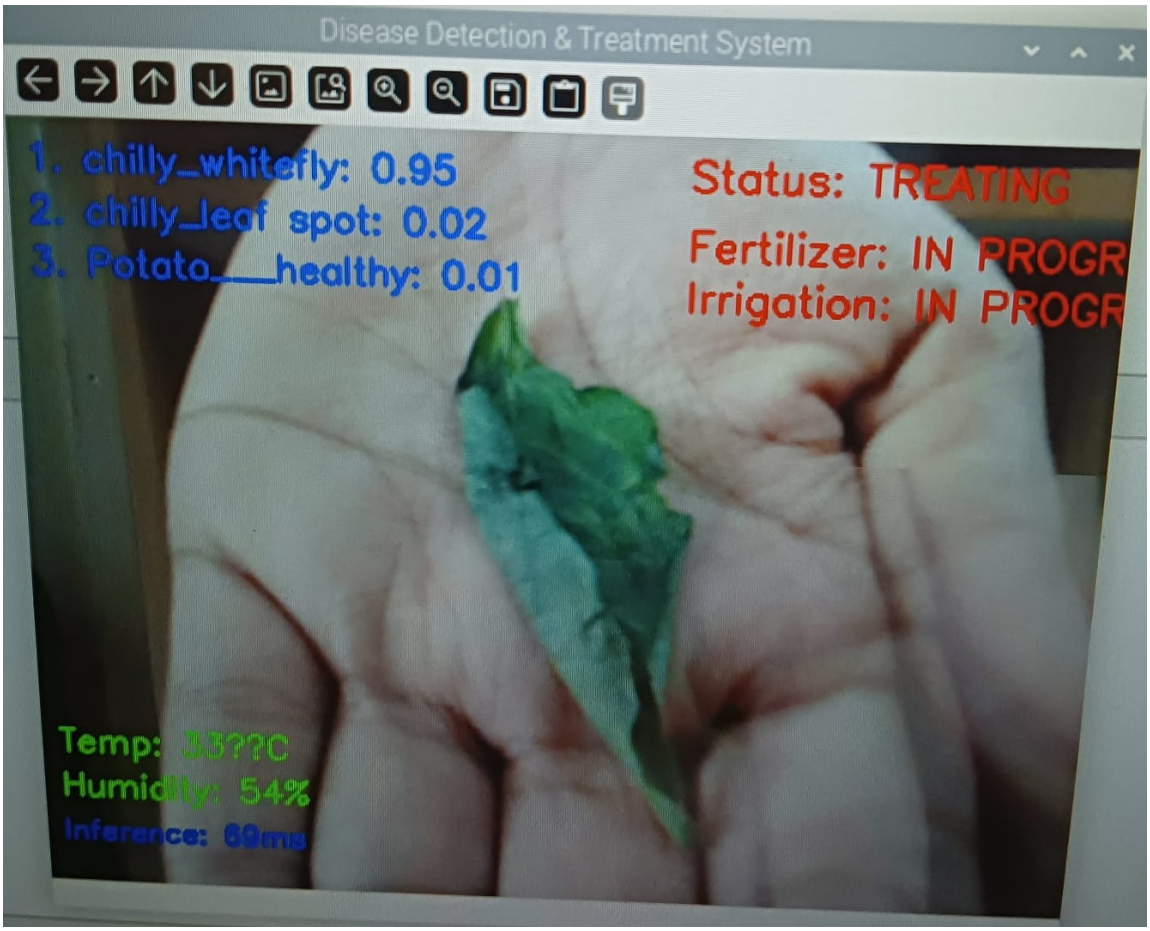
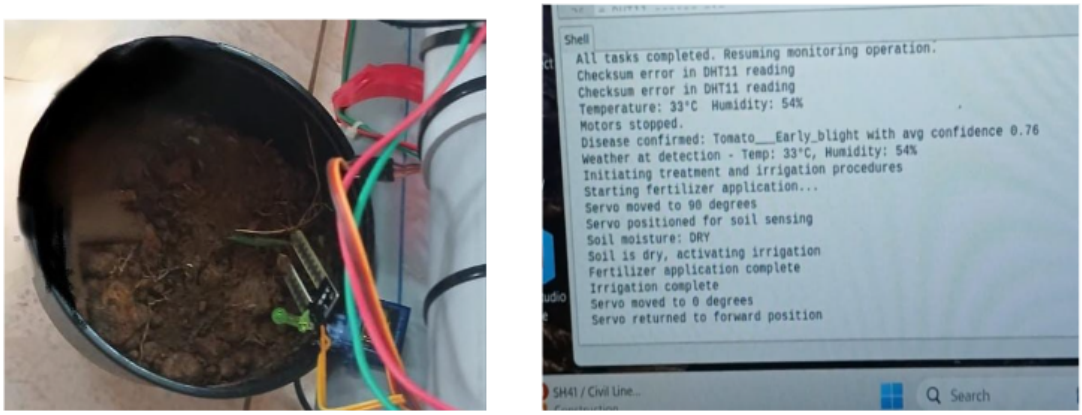


Figure 5.8: Output from Smart Farming System

5.5.1 Automated Irrigation System



(a) Dry Soil (b)Soil Moisture detecting the soil as dry

Figure 5.9: Output from Smart Farming System

The soil moisture sensor detects the dryness level of the soil when inserted. As shown in Figure 5.9, it identifies the soil as dry and immediately activates the irrigation system. Alongside watering, the fertilizer spraying process also begins. This automated setup ensures timely treatment without manual intervention for improved plant health. Figure 5.8 showcases an automated plant disease detection and treatment system powered by a Raspberry Pi. Once an image of the leaf is captured through the connected webcam module, the system processes it using a pre-trained deep learning model to identify any disease. In the example shown, the model confidently predicts the disease as "chilly__whitefly" with a 95% probability. Following the detection, the system automatically initiates the treatment process. As part of this, the soil moisture sensor evaluates the current condition of the soil to determine whether it is dry or wet. If the soil is dry, the irrigation system is triggered, and water is sprayed to support plant health and recovery. Conversely, if the soil is already moist, the system avoids unnecessary watering, thus conserving water. The interface provides real-time updates, displaying the treatment status as "TREATING" and indicating that both fertilizer and irrigation are "IN PROGRESS." Additionally, environmental parameters such as temperature, humidity (54%), and inference time (69ms) are also displayed on the screen, demonstrating the system's efficiency and practical applicability in smart farming.

5.5.2 Automated Fertilizer Spraying

Based on Figure 5.8, the system not only performs automated irrigation but also handles fertilizer spraying as part of the treatment process. Once the disease is accurately predicted using the Raspberry Pi and the connected camera module, the system begins a series of automated actions. First, the soil moisture sensor checks the soil condition and triggers irrigation if the soil is found to be dry. Following the completion of the irrigation process, the system proceeds to spray the appropriate fertilizer corresponding to the detected disease. This ensures that the plant receives the necessary nutrients or chemical treatment required to combat the identified disease. The display clearly indicates this sequential automation, showing "Fertilizer: IN PROGRESS" and "Status: TREATING" in red text, confirming that the fertilizer spraying is actively underway. This intelligent and responsive system minimizes human intervention, ensures timely action, and supports efficient disease management in smart farming practices.

5.5.3 Temperature and Humidity Sensing

In the smart farming system, temperature and humidity play a crucial role in monitoring the environmental conditions that affect plant health. The DHT11 sensor is used for this purpose, as it is a low-cost, digital sensor capable of measuring both temperature and relative humidity with reasonable accuracy. The DHT11 sensor successfully measured temperature and humidity levels in real-time, providing accurate environmental data to the system. During testing, the sensor consistently delivered stable readings that were used to monitor plant surroundings and ensure optimal growing conditions. These results enabled the smart farming system to make informed decisions, such as adjusting irrigation based on humidity levels, contributing to efficient and automated crop management.

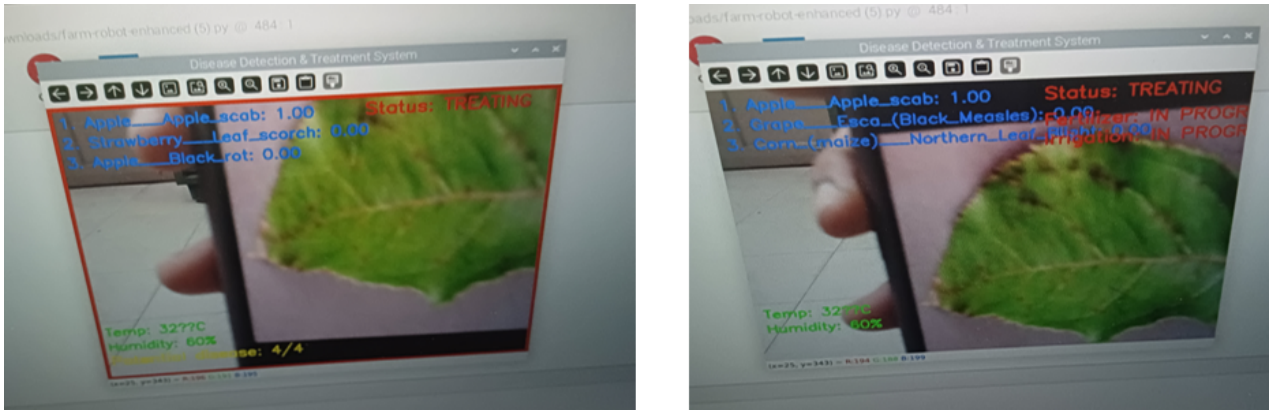


Figure 5.10: Temperature and Humidity reading

5.5.4 Rover Movement in the Smart Agriculture System

The rover in this smart farming system is designed to operate autonomously, navigating along a predefined path while continuously monitoring plant health using a camera module connected to the Raspberry Pi. The camera captures real-time images of the leaves, and these are processed using a pre-trained deep learning model to detect any signs of disease. The rover is programmed to stop only when an unhealthy leaf is detected i.e., when the prediction confidence indicates the presence of a disease. Once the rover halts, it initiates the treatment process, which includes checking the soil moisture, triggering irrigation if necessary, and spraying fertilizer specific to the diagnosed disease. If the leaf is predicted to be healthy, the rover continues moving without interruption. This behavior ensures efficient and selective treatment, reducing unnecessary stops and enhancing the overall efficiency of the smart agriculture system.

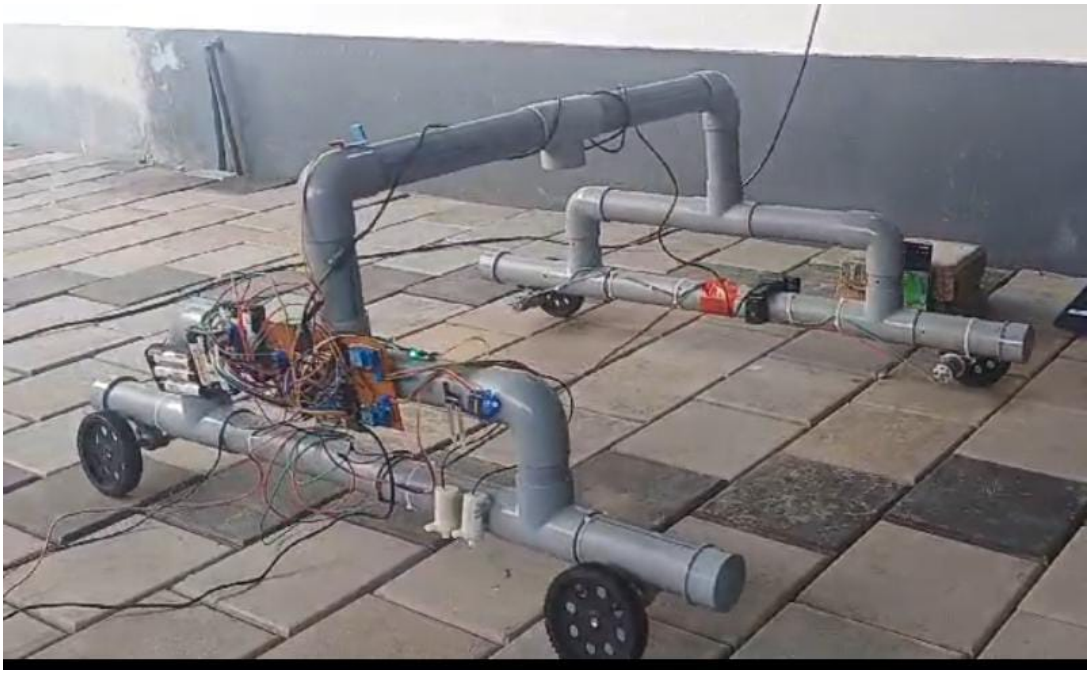


Figure 5.11: Structure of the Smart Farming System

Chapter 6

Conclusions & Future Scope

The smart farmbot developed in this project successfully automates essential farming tasks such as disease detection, irrigation, and fertilizer application. By autonomously moving across the farm, detecting leaf diseases, and responding with targeted fertilizer spraying, the system reduces manual labor and optimizes resource usage. Additionally, it monitors soil moisture levels and activates irrigation when necessary, ensuring efficient water management. Despite its success, the project faced challenges such as maintaining accurate disease detection under varying environmental conditions, optimizing the movement of the farmbot, and managing power consumption for continuous operation. However, the farmbot demonstrates a scalable and efficient approach to modernizing agriculture through automation, helping farmers improve productivity while minimizing resource wastage.

Future advancements can further enhance the farmbot's capabilities and sustainability. A moving camera module can be integrated to scan larger areas for better disease detection, while a weed removal mechanism using robotic arms can help eliminate unwanted plants, improving crop yield. AI-driven pesticide prevention can optimize chemical usage by applying pesticides only when necessary. Additionally, integrating solar panels will make the system energy-efficient, reducing dependency on external power sources. IoT-based remote monitoring and LoRa-based wireless communication can enable real-time tracking of soil conditions and farmbot performance over large farmlands. Further improvements, such as AI-driven decision support for predictive farming and multi-crop support, will make the system more adaptable and intelligent. With these advancements, the farmbot has the potential to revolutionize precision agriculture, making farming more automated, sustainable, and resource-efficient.

Chapter 7

Individual Contribution

My role involved setting up the Raspberry Pi 4 and connecting the key hardware parts to the FarmBot system. It starts with configuring the Raspberry Pi and installing the required software needed for detecting plant diseases and automating farming actions.

A camera module is connected to capture clear, high-resolution images of plants. These images are later analyzed by an AI model to detect any plant diseases. Along with this, important FarmBot sensors—like soil moisture, temperature, and humidity sensors—are also integrated. These sensors help monitor the farm's environment and support automated actions like watering or fertilizing when needed.

Efforts are also made to ensure smooth communication between the Raspberry Pi, sensors, and the AI model.Together with Anu Xavier, testing is done on the movement and automation parts of the FarmBot. This ensures that the machine can perform all its functions smoothly, including responding to sensor inputs and taking actions like spraying or irrigation.

By setting up and connecting all the hardware parts correctly, the system is made ready to collect real-time data and make smart farming decisions using AI. This setup is important to make sure the FarmBot works as planned, linking the software with the actual physical tools on the farm.

References

[1]. Senthil Kumar Swami Durai, Mary Divya Shamili, Smart farming using Machine Learning and Deep Learning techniques, Decision Analytics Journal, Volume 3, 2022,100041,ISSN 2772-6622, https://doi.org/10.1016/j.dajour.2022.100041.

[2]. Durai, Senthil Kumar Swami, and Mary Divya Shamili. "Smart farming using machine learning and deep learning techniques." Decision Analytics Journal 3 (2022): 100041.

[3]. Archana, U., et al. "Plant disease detection using resnet." 2023 International Conference on Inventive Computation Technologies (ICICT). IEEE, 2023. DOI : 10.1109/ICICT57646.2023.10133938

[4]. Nobel, SM Nuruzzaman, et al. "Palm leaf health management: A hybrid approach for automated disease detection and therapy enhancement." IEEE access 12 (2024): 9097-9111.

[5]. Pandya, Jesu Raj, et al. "FarmBot-A Platform for Backyard Precision Farming: Installation and Initial Experimental Layout." 2019 ASABE Annual International Meeting. American Society of Agricultural and Biological Engineers, 2019.

[6]. Kar, Sudipto, et al. "Farmbot: an IoT-Based Wireless Agricultural Robot for Smart Cultivation." 2023 26th International Conference on Computer and Information Technology (ICCIT). IEEE, 2023.

[7]. Too, Edna Chebet, et al. "A comparative study of fine-tuning deep learning models for plant disease identification." Computers and Electronics in Agriculture 161 (2019): 272-279.

[8]. Swaminathan, Akshay, C. Varun, and S. Kalaivani. "Multiple plant leaf disease classification using densenet-121 architecture." Int. J. Electr. Eng. Technol 12 (2021): 38-57.

[9]. Mukti, Ishrat Zahan, and Dipayan Biswas. "Transfer learning based plant diseases detection using ResNet50." 2019 4th International conference on electrical information and communication technology (EICT). IEEE, 2019.

[10]. Eunice, Jennifer, et al. "Deep learning-based leaf disease detection in crops using images for agricultural applications." Agronomy 12.10 (2022): 2395.

[11]. Sawant, Divya, Anchal Jaiswal, Jyoti Singh, and Payal Shah. "AgriBot – An Intelligent Interactive Interface to Assist Farmers in Agricultural Activities." 2019 IEEE Bombay Section Signature Conference (IBSSC) (2019): 1–6.

[12]. Abinaya, A., S. Nivetha, K. Ranjitha, and M. Anusha. "Automated Irrigation System Based on Wireless Sensor Network." International Journal of Trend in Scientific Research and Development 2.2 (2018): 441–445.

[13]. Gao, Shang-Hua, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. "Res2Net: A New Multi-scale Backbone Architecture." IEEE Transactions on Pattern Analysis and Machine Intelligence (2021): 1–14.

[14]. Aravind, Karthikeyan, and Subramanian R. "A Survey on Precision Agriculture and the Role of Machine Learning." Frontiers in Plant Science 14 (2023): 1258658.

[15]. Arthi, C., M. Manjupriya, and S. Ranjitha. "Mulberry Leaf Disease Detection Using CNN-Based Smart Android Application." International Journal of Research Publication and Reviews 3.9 (2022): 208–213.

Appendix A: IEEE Conference Paper

Smart Farming System

Alka Denny
Dept. of Electronics and Communication Engineering
Rajagiri School of Engineering And Technology
Kochi,India
alkaden2020@gmail.com

Anithra Ross Ajith
Dept. of Electronics and Communication Engineering
Rajagiri School of Engineering And Technology
Kochi,India
anithrarossajith1@gmail.com

Anna Joju
Dept. of Electronics and Communication Engineering
Rajagiri School of Engineering And Technology
Kochi,India
jojuanna12@gmail.com

Anu Xavier
Dept. of Electronics and Communication Engineering
Rajagiri School of Engineering And Technology
Kochi,India
anuannaxavier73@gmail.com

Swapna Davies
Dept. of Electronics and Communication Engineering
Rajagiri School of Engineering And Technology
Kochi,India
swapnad@rajagiritech.edu.in

S. Santhi Jabarani
Dept. of Electronics and Communication Engineering
Rajagiri School of Engineering And Technology
Kochi,India
santhij@rajagiritech.edu.in

***Abstract*—Agriculture faces significant challenges due to plant diseases and inefficient water usage, leading to reduced crop yield and resource wastage. This paper presents a smart agriculture system integrating deep learning and IoT technologies to enhance precision in disease detection and irrigation. A ResNet50-based Convolutional Neural Network (CNN) model is employed for accurate leaf disease classification, enabling targeted fertilizer spraying based on the identified disease. Additionally, a soil moisture sensor ensures precise irrigation, conserving water by supplying it only when necessary. The system also incorporates real-time weather monitoring to provide farmers with environmental insights for better decision-making. By leveraging automation and machine learning, this approach enhances crop health, optimizes water usage, and improves agricultural productivity. Experimental results demonstrate the system's effectiveness in reducing resource consumption while maintaining high accuracy in disease detection.**

***Keywords*—Classification, Machine learning, Precise irrigation, Agriculture**

I. Introduction

Agriculture plays a vital role in the global economy, with a significant portion of the population depending on it for livelihood. However, traditional farming practices often result in challenges such as low productivity and inefficient resource utilization. One of the major threats to crop yield is plant diseases, which can cause significant economic losses if not detected and managed promptly. The emergence of deep learning techniques, particularly Convolutional Neural Networks (CNNs), has shown promise in addressing these challenges by enabling accurate and automated detection of plant diseases through image-based analysis [1].

In recent years, ResNet50, a deep residual learning framework, has gained prominence for its ability to handle complex image classification tasks with high accuracy. Its use of residual blocks helps mitigate the vanishing gradient problem, allowing the model to learn deeper representations effectively. Studies have demonstrated the efficacy of ResNet50 in identifying various plant diseases with remarkable accuracy. For instance, a study employing ResNet50 for detecting rice plant diseases achieved an accuracy of 96.35%, highlighting its potential for real-world agricultural applications [2].

Building on this foundation, the proposed system integrates ResNet50 for leaf disease detection with a precise irrigation and fertilizer management strategy. The system uses a Raspberry Pi 4 as the processing unit to execute the model and manage data from various sensors. Soil moisture sensors provide real-time data to control irrigation levels precisely, ensuring optimal water usage based on soil conditions. Simultaneously, the disease classification results from ResNet50 inform the fertilizer management system, enabling the application of appropriate nutrients to mitigate the impact of identified diseases [1].

The adoption of advanced computing techniques in agriculture is transforming traditional practices. Devices such as soil moisture sensors, combined with powerful processors like Raspberry Pi 4, facilitate real-time monitoring and control of farming activities. The integration of these technologies not only enhances crop health monitoring but also contributes to resource conservation by preventing excessive water and fertilizer use [1], [2].

The proposed system's architecture consists of three main modules: disease detection, irrigation control, and fertilizer management. The disease detection module captures and preprocesses leaf images, which are then fed into the ResNet50 model for classification. Based on the detected disease, the system suggests specific fertilizers to address

nutrient deficiencies caused by the disease. The irrigation control module utilizes data from soil moisture sensors to adjust water levels precisely, reducing waste and ensuring that crops receive adequate hydration. The entire process is managed by the Raspberry Pi 4, which coordinates sensor data collection, image processing, and actuator control [1].

The use of Raspberry Pi 4 is particularly advantageous due to its low cost, energy efficiency, and compatibility with machine learning frameworks like TensorFlow and PyTorch. It serves as a bridge between sensors and the deep learning model, enabling efficient execution of tasks in a resource-constrained environment. Moreover, the proposed system's reliance on open-source software and affordable hardware makes it accessible to small-scale farmers, promoting the adoption of smart farming practices in developing regions [2].

In conclusion, the integration of ResNet50 for leaf disease detection with precise irrigation and fertilizer management presents a comprehensive solution for enhancing agricultural productivity. By leveraging deep learning and edge computing on the Raspberry Pi 4 platform, the system addresses critical challenges in agriculture, including disease management, water conservation, and optimal fertilizer usage. Future work will focus on expanding the model's capabilities to detect a broader range of diseases and incorporating additional sensors for enhanced monitoring [1], [2].

II. Related works

Recent advancements in smart agriculture have focused on integrating IoT, artificial intelligence, and automation to enhance farming efficiency. Various studies have explored automated irrigation, disease detection, and robotic systems, highlighting the transformative impact of technology in modern agriculture.

In [1], an IoT-enabled agricultural robot is presented for smart cultivation. This system employs multiple sensors to monitor real-time soil and environmental conditions, ensuring precise water and fertilizer distribution. The research emphasizes the benefits of data-driven decision-making for improved crop management.

The study in [2] introduces an automated irrigation system utilizing wireless sensor networks. The system leverages soil moisture sensors to regulate water supply dynamically, reducing wastage and improving irrigation efficiency. The findings indicate that automation significantly enhances water resource management in agriculture.Deep learning-based plant disease detection has gained prominence in recent research. In [3], a hybrid method for palm leaf disease detection is proposed, integrating image processing with machine learning techniques. The approach facilitates early disease identification, preventing potential crop losses and improving yield outcomes.

Furthermore, [4] explores plant disease detection using ResNet50, a deep learning model renowned for image classification. The authors illustrate how convolutional neural networks (CNNs) significantly boost disease identification accuracy, enabling timely and precise agricultural interventions.Another significant work in [5] discusses the integration of automated agricultural mechanisms with relay modules for precise actuation. The study presents an intelligent spraying system that activates upon disease detection, ensuring targeted pesticide and fertilizer application while minimizing resource wastage.

Collectively, these studies underscore the vital role of automation, sensor networks, and AI-driven disease detection in smart farming. The convergence of these technologies paves the way for a more sustainable and efficient agricultural ecosystem, serving as the foundation for our proposed system.

III. materials and methods

The smart farming system presented in this study integrates deep learning and automation to enhance agricultural efficiency by enabling real-time disease detection, soil moisture assessment, and precision-based interventions. The system is designed to function autonomously, reducing human dependency while improving the accuracy and effectiveness of farming practices. At its core, the system consists of a mobile rover that follows a predefined track, scanning plant leaves for potential diseases using a deep learning model. Upon detecting a disease, the system initiates a sequence of automated actions, including soil moisture measurement, water sprinkling, and fertilizer application.

To achieve accurate disease classification, the system employs a dual-model deep learning approach, leveraging DenseNet-121 and ResNet-50. The datasets used for training and evaluating the plant disease detection model were collected from publicly available sources such as Kaggle and Mendeley. These datasets comprise a diverse range of leaf images from crops like tomato, chili, spinach, and others, covering both healthy and diseased samples. Each image in the dataset is labeled with its corresponding class, enabling supervised learning for disease classification. The inclusion of multiple crop types and various disease categories ensures that the model is trained on a wide variety of patterns and symptoms, enhancing its robustness and accuracy in real-world farming environments. These rich and well-curated datasets played a crucial role in building a reliable and effective deep learning model for smart agriculture applications.

The rover is equipped with a high-resolution camera for continuous image acquisition, a motor driver for controlled movement, and an intelligent stopping mechanism that halts operations when a diseased plant is detected. This allows for precise application of treatments and targeted analysis of affected areas. The system also features an automated irrigation mechanism, triggered based on real-time soil moisture readings. If the soil is dry, water sprinkling is activated to optimize hydration. Similarly, a fertilizer spraying mechanism ensures timely treatment for diseased plants. Additionally, continuous environmental monitoring is integrated through sensors that measure temperature and humidity every two seconds, providing valuable insights into farm conditions. By combining machine learning with automation, this methodology ensures resource efficiency,

minimizes wastage, and enhances disease detection accuracy. The system's real-time processing capabilities and autonomous decision-making offer a scalable and innovative solution for modern agriculture, paving the way for improved crop health management and sustainable farming practices.

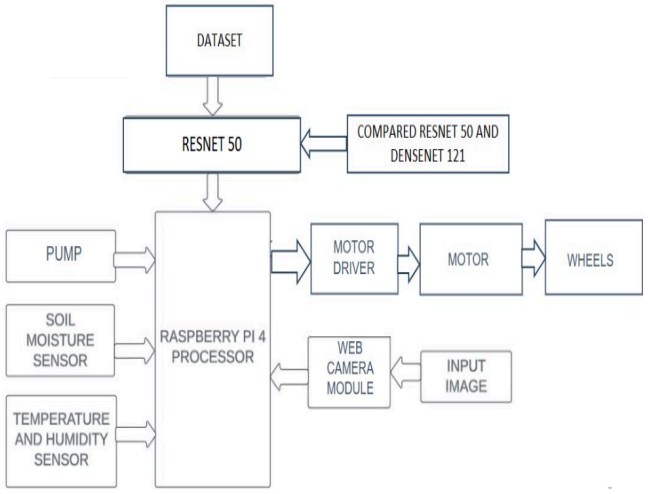


Fig 1: Block Diagram of the Smart Farming System

A. Leaf Disease classification

The smart farming system employs deep learning-based classification to detect leaf diseases with high accuracy. Two advanced convolutional neural networks (CNNs), DenseNet-121 and ResNet-50, are used to analyze plant leaf images and classify them as either healthy or affected by diseases. These models were chosen for their strong feature extraction capabilities, which enhance the precision of disease detection.

a. Densenet121 model

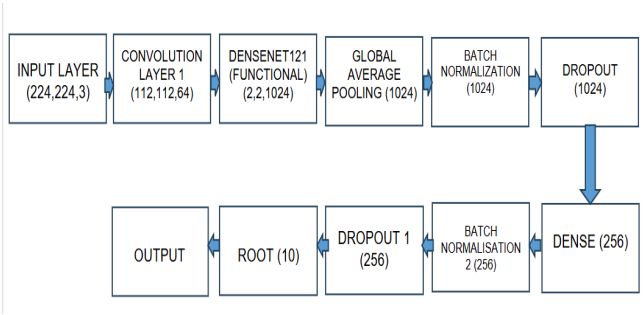


Fig 2: DenseNet121 Architecture

Fig 2 utilizes the DenseNet121 model as a feature extractor within a custom deep learning classification pipeline. The model begins with an input layer that accepts RGB images of size 224x224x3, followed by an initial convolutional layer that reduces the spatial dimensions and extracts low-level features. The output is then passed into a pre-trained DenseNet121 model (excluding its top classification layers), which outputs a rich feature map of size 2x2x1024. This output undergoes Global Average Pooling to flatten it into a 1024-dimensional feature vector. To enhance learning stability and prevent overfitting, Batch Normalization and Dropout layers are applied. The resulting features are further passed through a dense (fully connected) layer with 256 neurons, followed by another batch normalization and dropout layer. Finally, the network ends with a dense layer named "Root" with 10 units, which likely corresponds to the number of target classes. The final output layer provides the predicted class probabilities. This structure leverages DenseNet121's robust feature extraction capabilities while adapting the final layers for a specific classification task through transfer learning.

b. ResNet50 model

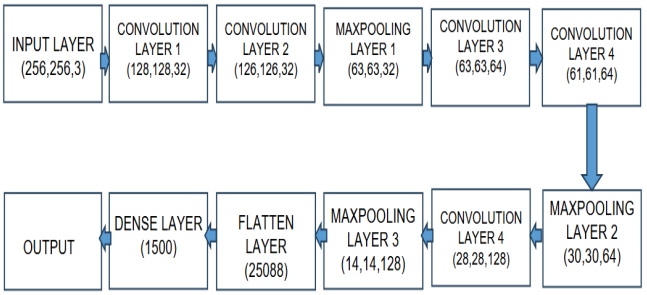


Fig 3: ResNet50 Architecture

Fig 3 represents a custom convolutional neural network (CNN) inspired by ResNet50 principles but does not directly depict the standard ResNet50 structure. It starts with an input layer for RGB images of size 256x256x3, followed by two convolutional layers that progressively extract low-level features, reducing the spatial dimensions. MaxPooling is applied to downsample the feature maps, after which deeper convolutional layers (with increased filter depth) further refine feature extraction. Another MaxPooling layer reduces the spatial size, followed by two more convolutional layers with 128 filters, extracting more complex patterns. The final MaxPooling operation downsamples the feature maps to 14x14x128. These are then flattened into a 25,088-dimensional vector and passed through a dense layer with 1500 neurons. The model ends with an output layer, likely used for classification. Though titled ResNet50, this architecture does not include residual connections, which are the hallmark of ResNet. Instead, it appears to be a simplified, custom CNN influenced by ResNet's deep structure design philosophy.

In our project, we chose the ResNet-50 model over DenseNet-121 primarily due to performance limitations observed during real-time implementation on the Raspberry Pi 4. While both models delivered good accuracy, DenseNet-121 caused significant delays when processing real-time images from the web camera module. This lag hindered the responsiveness required for timely disease detection in smart farming applications. In contrast, ResNet-50 offered a more efficient balance between accuracy and inference speed, making it better suited for deployment on resource-constrained devices like the Raspberry Pi. Its faster processing enabled smoother real-time performance, ensuring prompt and reliable detection in field conditions.

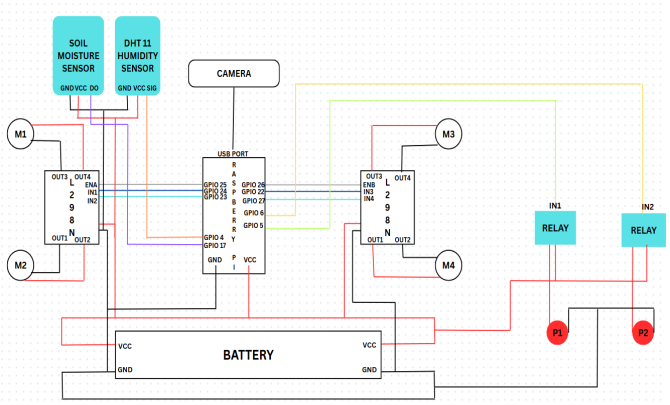


Fig 4: Circuit Diagram of the Smart Farming System

B. Rover Navigation and Automated Stopping Mechanism

The rover is a vital part of the smart farming system, designed to autonomously navigate a predefined track while scanning crops for diseases. Controlled by L298N motor driver modules, the rover uses four DC motors (M1–M4) to drive its wheels, enabling systematic movement across the field. A web camera module connected via the USB port to the Raspberry Pi 4 captures real-time images of the crops. These images are processed by the ResNet-50 deep learning model, which runs efficiently on the Raspberry Pi due to its balance between performance and computational load, making it ideal for embedded systems.

A key feature of this system is the automated stopping mechanism, triggered when the ResNet-50 model detects a diseased plant. Upon detection, the rover halts, and a soil moisture sensor checks the moisture level in the soil. If the soil is dry, the Raspberry Pi activates a relay module, which controls Pump 1 (P1) to sprinkle water. Simultaneously, another relay controls Pump 2 (P2) to spray fertilizer on the diseased plant. A DHT11 temperature and humidity sensor is also connected to monitor environmental conditions, which are displayed or used for data logging. The entire system is powered by a battery pack that ensures portability and uninterrupted field operation. All sensors and actuators are connected through appropriate GPIO pins of the Raspberry Pi, chosen for their availability and ease of control in Python. This integrated use of components ensures real-time decision-making, efficient resource utilization, and automation in precision agriculture, reducing manual labor and increasing productivity.

C. Soil Moisture Detection and Water Sprinkling System

The smart farming system includes a soil moisture detection module to ensure efficient irrigation. A soil moisture sensor connected to the Raspberry Pi 4 GPIO pins is used to measure the water content in the soil specifically at the location where a disease is detected by the system. This sensor was chosen for its low power consumption, simple interfacing, and real-time analog output, making it ideal for embedded applications like ours. If the sensor detects dry soil, the Raspberry Pi processes this information and triggers a relay module connected to Pump 1 (P1), which activates the water sprinkling mechanism. The relay acts as a switch that safely controls the high-power pump from the low-power GPIO output of the Raspberry Pi. This setup ensures water is used only when necessary, preventing overwatering and conserving valuable resources.

The decision-making process is entirely automated, relying on real-time sensor readings and the microcontroller's logic. By integrating these components—soil moisture sensor, Raspberry Pi, relay module, and water pump—the system effectively optimizes irrigation management. This not only improves crop health by maintaining ideal soil moisture levels but also promotes sustainable water usage in agriculture, aligning with the goals of precision farming.

D. Fertilizer Spraying for Disease Treatment

The smart farming system integrates an automated fertilizer spraying mechanism to ensure timely and targeted treatment of diseased crops. Upon disease detection by the ResNet-50 model, a control signal is sent from the Raspberry Pi 4 to Relay Module 2, which activates Pump 2 (P2) connected to the fertilizer sprayer. The use of a relay module allows the system to safely switch high-power components like pumps using low-power GPIO signals. This design ensures that fertilizer or pesticide is applied only when required, improving accuracy and reducing excessive chemical usage. The controlled and responsive mechanism minimizes manual effort, optimizes chemical consumption, and supports sustainable farming by ensuring precise intervention at the right time.

E. Continuous Environmental Monitoring

The smart farming system features continuous environmental monitoring to track key parameters like temperature and humidity in real time. These critical factors influence plant health and disease vulnerability. A DHT11 temperature and humidity sensor is used to collect environmental data at regular two-second intervals. This sensor was chosen for its simplicity, reliability, and compatibility with the Raspberry Pi 4 GPIO pins, making it ideal for low-cost and efficient climate sensing. The real-time data gathered is processed by the Raspberry Pi 4, allowing immediate insights into the farm's microclimate. This information supports informed decisions for disease prevention and optimized irrigation. By automating climate monitoring, the system minimizes manual intervention, ensures optimal growing conditions, and enhances overall crop productivity. The integration of the DHT11 sensor ensures proactive responses to environmental changes, making farming more efficient and sustainable.

V. RESULTS AND DISCUSSION

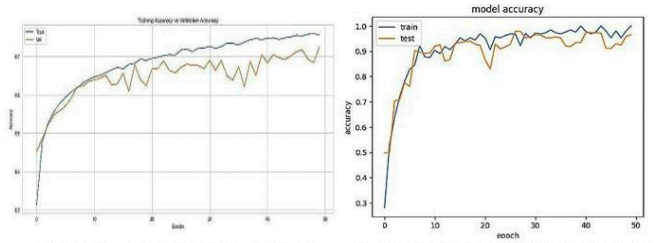
This section presents the outcomes of implementing the smart farming system, highlighting the performance of disease detection, automation accuracy, and real-time responsiveness. It also discusses the effectiveness of the chosen components and models in achieving the project's objectives.

A. Model Accuracy and Loss evaluation

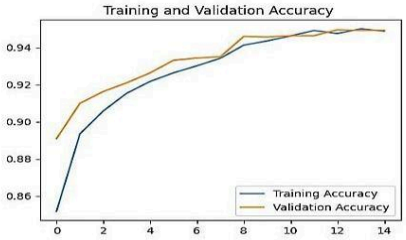
a. DenseNet121 vs Resnet50 Model Accuracy Curve

The three graphs presented illustrate the training and validation accuracy trends of two deep learning models—ResNet50 and DenseNet121—under different class settings. Each graph provides insight into how the models perform in terms of learning capacity, generalization, and stability, helping guide the decision for the best model to use in the final implementation.

The first graph (a) shows the accuracy curve of the ResNet50 model trained on a dataset with 10 classes. Both training and validation accuracy rise steadily across the epochs. Training accuracy surpasses 90% and continues to improve gradually, while validation accuracy also trends upward with minor fluctuations. The close alignment between training and validation lines suggests that the model is learning effectively without significant overfitting. This indicates that ResNet50 generalizes well on unseen data while maintaining high accuracy, making it a reliable choice for classification tasks involving 10 categories.



(a) ResNet50 Model Accuracy Graph in 10 Classes (b) DenseNet121Model Accuracy Graph in 10 Classes



(c) ResNet50 Model Accuracy Graph in 30 Classes

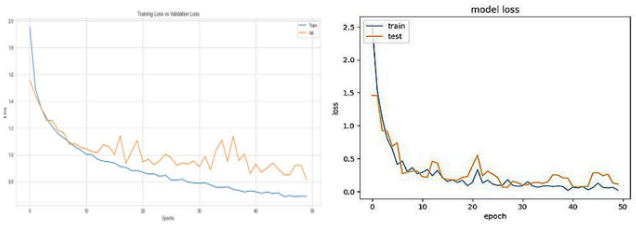
Fig 4: Accuracy Curves

In contrast, graph (b) displays the DenseNet121 model accuracy when trained on the same 10-class dataset. The model shows a rapid increase in both training and test accuracy during the early epochs, quickly reaching over 90%. However, the validation accuracy fluctuates more noticeably than ResNet50, especially around the 20–30 epoch range. These inconsistencies imply that DenseNet121, while powerful, may be more sensitive to the training dynamics, and could be prone to minor overfitting or instability in validation performance. Though it performs well in terms of raw accuracy, this lack of consistency might be a limitation in real-world applications. The third graph (c) presents the ResNet50 model accuracy on a more complex dataset with 30 classes. Remarkably, the model still performs strongly, with training and validation accuracy approaching 94%. The curves are smooth and closely follow each other throughout the training process, showing that ResNet50 scales well to increased class complexity without sacrificing performance. The very small gap between training and validation accuracy further supports its excellent generalization capability.

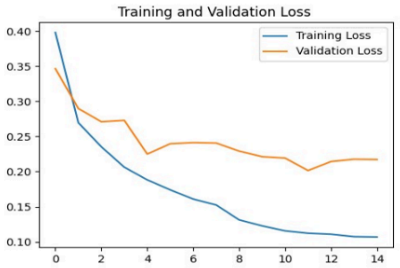
Based on these observations, ResNet50 was chosen over DenseNet121 for the final implementation. ResNet50 offers smoother and more stable learning curves, demonstrating consistent performance even when the number of classes increases. It generalizes well, showing little to no overfitting across different tasks. Furthermore, ResNet50 is relatively lightweight compared to DenseNet121, making it more suitable for real-time applications or deployment on resource-constrained devices like the Raspberry Pi. While DenseNet121 is also a strong model, its slightly unstable validation performance and higher computational demands made ResNet50 the more practical and robust choice for the system.

b. Densenet121 vs Resnet50 Model Loss Curve

The graphs presented above depict the training and validation loss curves for the ResNet50 and DenseNet121 models, each trained on datasets with 10 and 30 classes respectively. These loss curves are essential for evaluating how well a model learns from data, and they also offer deeper insight into the model's generalization ability and risk of overfitting.



(a) ResNet50 Model Loss Graph in 10 Classes (b) DenseNet121Model Loss Graph in 10 Classes



(c) ResNet50 Model Loss Graph in 30 Classes

Fig 5: Loss Curves

The first graph (a) shows the loss curve of the ResNet50 model trained on a 10-class dataset. The training loss decreases steadily and significantly over the epochs, indicating that the model is effectively minimizing error on the training data. Although the validation loss follows a similar downward trend, it is more erratic, with frequent fluctuations. This variance may be attributed to either noise in the validation data or sensitivity to certain samples, but overall, the validation loss still trends downward. Importantly, the gap between training and validation loss is not extreme, suggesting that ResNet50 maintains reasonable generalization on unseen data. Graph (b) displays the DenseNet121 loss curve on the same 10-class dataset. Both training and validation losses start high and then rapidly decrease within the first few epochs, which is typical of deep learning models. The validation loss closely follows

the training loss throughout the entire training period, indicating tight coupling between the two curves. However, small spikes are visible at several points in the validation curve, hinting at potential overfitting or unstable learning behavior. Despite this, the average loss remains low, and the model performs well. Still, the instability observed during training could pose reliability concerns in deployment environments. The third graph (c) illustrates the loss behavior of ResNet50 on a more complex dataset with 30 classes. Here, both training and validation losses decrease consistently, with the training loss reaching very low levels. While the validation loss flattens and slightly fluctuates after epoch 6, it still continues its downward trend overall. The gap between the training and validation losses increases slightly as training progresses, which is expected in more complex classification tasks. Nevertheless, there is no sharp rise in validation loss, indicating that overfitting is well-controlled and the model remains stable across the training period.

Based on these loss graphs, ResNet50 again proves to be the better choice compared to DenseNet121. It demonstrates smooth and steady reduction in training loss and shows reliable validation loss trends even when the number of classes increases. DenseNet121, while showing good initial performance, exhibits minor instability in validation loss that could translate to inconsistent results in real-time applications. Additionally, ResNet50 appears more robust and adaptable to datasets of increasing complexity, as shown in the 30-class loss curve. These results further support the decision to select ResNet50 for the final implementation, balancing high accuracy with stable and predictable training behavior.

B. Farmbot

Fig 6: Structure of the Farmbot

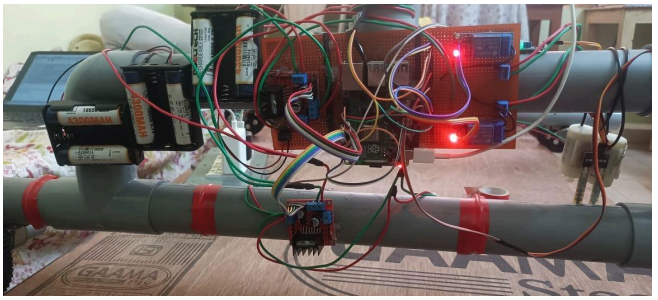
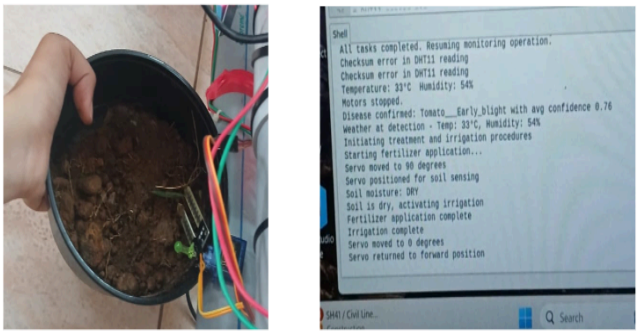


Fig 7: Connections To Raspberry Pi 4

Fig 6 displays the physical structure of the farmbot, constructed using PVC pipes to form a sturdy rectangular frame mounted on wheels for mobility. At the center, a camera module is attached to capture real-time images of crops for disease detection. The simple yet functional design allows the farmbot to navigate along predefined tracks in the field. Fig 7 presents the intricate circuitry and wiring connected to the Raspberry Pi processor. This includes multiple components such as battery packs for power supply, motor drivers, relay modules, and various sensors. The connections are mounted on a custom PCB board, demonstrating a compact and efficient hardware setup to support autonomous operations like image processing, soil moisture sensing, and actuation of sprinklers or fertilizer sprayers. Together, these images illustrate the blend of mechanical design and embedded electronics driving the functionality of the smart farming system.



(a) Dry Soil (b)Soil Moisture detecting the soil as dry

Fig 8: The soil moisture sensor detecting the soil as dry

Fig 8 illustrates the dry soil detection and treatment process in the smart farming system. In sub-image (a), a soil moisture sensor is inserted into a plant pot containing dry soil. This sensor is connected to the automated system powered by a Raspberry Pi. Sub-image (b) shows the terminal output from the Raspberry Pi, displaying the detection and decision-making process in real time.

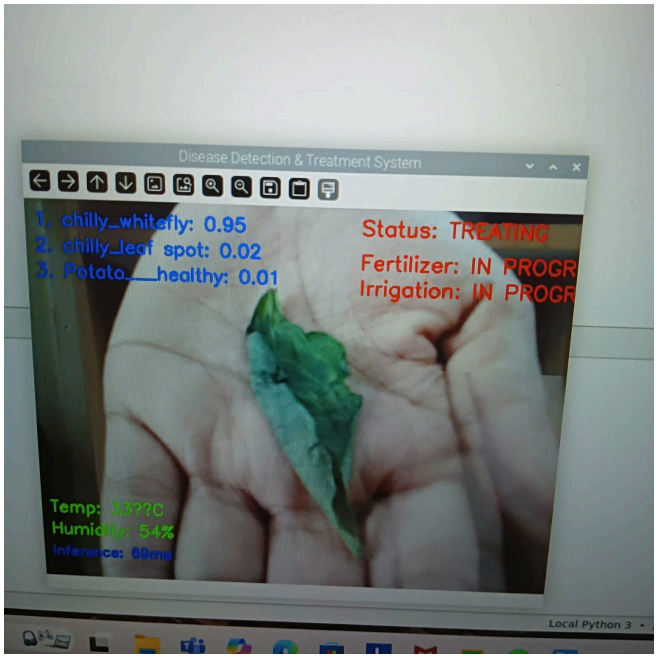


Fig 9: The output from the Smart Farming System

Fig 9 showcases the real-time output of the plant disease detection and treatment module. On the top-left, in blue text, the system displays the prediction results from the machine learning model: “chilly_whitefly” with a confidence of 95%, followed by “chilly_leaf spot” (2%), and “Potato_healthy” (1%). This clearly indicates that the leaf being analyzed is most likely infected by the whitefly pest. On the right side, in red, the Status is marked as “TREATING”, while both “Fertilizer” and “Irrigation” are indicated as “IN PROGRESS”, signifying that the system has autonomously initiated both processes. These operations are controlled based on real-time sensor readings, also displayed on the screen. At the bottom-left corner, in green, the system shows the current temperature (33°C) and humidity level (54%), captured through environmental sensors. The inference time is shown as 89 milliseconds, reflecting the model's efficiency in detecting disease. All of this data including prediction results, sensor values, and system statuses is transmitted wirelessly via Wi-Fi from the Raspberry Pi, which acts as the central processing unit of the smart farming system. Once the irrigation and fertilizer spraying tasks are completed, the respective labels will update from “IN PROGRESS” to “COMPLETED”, indicating the end of the treatment cycle.

V. Conclusion

The smart farmbot developed in this project successfully automates essential farming tasks such as disease detection, irrigation, and fertilizer application. By autonomously moving across the farm, detecting leaf diseases, and responding with targeted fertilizer spraying, the system reduces manual labor and optimizes resource usage. Additionally, it monitors soil moisture levels and activates irrigation when necessary, ensuring efficient water management. Despite its success, the project faced challenges such as maintaining accurate disease detection under varying environmental conditions, optimizing the movement of the farmbot, and managing power consumption for continuous operation. However, the farmbot demonstrates a scalable and efficient approach to modernizing agriculture through automation, helping farmers improve productivity while minimizing resource wastage.

Future advancements can further enhance the Farmbot’s capabilities and sustainability. A moving camera module can be integrated to scan larger areas for better disease detection, while a weed removal mechanism using robotic arms can help eliminate unwanted plants, improving crop yield. AI-driven pesticide prevention can optimize chemical usage by applying pesticides only when necessary. Additionally, integrating solar panels will make the system energy-efficient, reducing dependency on external power sources. LoRa-based wireless communication can enable real-time tracking of soil conditions and farmbot performance over large farmlands. Further improvements, such as AI-driven decision support for predictive farming and multi-crop support, will make the system more adaptable and intelligent. With these advancements, the farmbot has the potential to revolutionize precision agriculture, making farming more automated, sustainable, and resource-efficient.

References

[1] Senthil Kumar Swami Durai, Mary Divya Shamili, Smart farming using Machine Learning and Deep Learning techniques, Decision Analytics Journal, Volume 3, 2022,100041,ISSN 2772-6622, https://doi.org/10.1016/j.dajour.2022.100041.

[2] U. Archana, A. Khan, A. Sudarshanam, C. Sathya, A. K. Koshariya and R. Krishnamoorthy, "Plant Disease Detection using ResNet," *2023 International Conference on Inventive Computation Technologies (ICICT)*, Lalitpur, Nepal, 2023, pp. 614-618, doi: 10.1109/ICICT57646.2023.10133938.

[3] S. M. N. Nobel *et al*., "Palm Leaf Health Management: A Hybrid Approach for Automated Disease Detection and Therapy Enhancement," in *IEEE Access*, vol. 12, pp. 9097-9111, 2024, doi: 10.1109/ACCESS.2024.3351912.

[4] A. Salam, M. Naznine, N. Jahan, E. Nahid, M. Nahiduzzaman and M. E. H. Chowdhury, "Mulberry Leaf Disease Detection Using CNN-Based Smart Android Application," in *IEEE Access*, vol. 12, pp. 83575-83588, 2024, doi: 10.1109/ACCESS.2024.3407153.

[5] R. Rani, J. Sahoo, S. Bellamkonda, S. Kumar and S. K. Pippal, "Role of Artificial Intelligence in Agriculture: An Analysis and Advancements With Focus on Plant Diseases," in *IEEE Access*, vol. 11, pp. 137999-138019, 2023, doi: 10.1109/ACCESS.2023.3339375.

[6] M. S. Chaitanya, U. K. Reddy B, S. K and S. Amaran, "MFC Based Smart Irrigation System Using IoT and Machine Learning," *2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*, Goathgaun, Nepal, 2025, pp. 368-373, doi: 10.1109/ICMCSI64620.2025.10883570.

[7] J. Gutiérrez, J. F. Villa-Medina, A. Nieto-Garibay and M. Á. Porta-Gándara, "Automated Irrigation System Using a Wireless Sensor Network and GPRS Module," in *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 1, pp. 166-176, Jan. 2014, doi: 10.1109/TIM.2013.2276487

[8] M. N. Mowla, N. Mowla, A. F. M. S. Shah, K. M. Rabie and T. Shongwe, "Internet of Things and Wireless Sensor

Networks for Smart Agriculture Applications: A Survey," in *IEEE Access*, vol. 11, pp. 145813-145852, 2023, doi: 10.1109/ACCESS.2023.3346299.

[9] E. -T. Bouali, M. R. Abid, E. -M. Boufounas, T. A. Hamed and D. Benhaddou, "Renewable Energy Integration Into Cloud & IoT-Based Smart Agriculture," in *IEEE Access*, vol. 10, pp. 1175-1191, 2022, doi: 10.1109/ACCESS.2021.3138160.

[10] S. Kar, R. Mohammad, M. T. Zaman, S. M. Reza, O. Talukder and M. Hasan, "Farmbot: an IoT-Based Wireless Agricultural Robot for Smart Cultivation," *2023 26th International Conference on Computer and Information Technology (ICCIT)*, Cox's Bazar, Bangladesh, 2023, pp. 1-6, doi: 10.1109/ICCIT60459.2023.10441291.

Appendix B: Vision, Mission, Programme Outcomes and Course Outcomes

Vision, Mission, Programme Outcomes and Course Outcomes

Institute Vision

To evolve into a premier technological institution, moulding eminent professionals with creative minds, innovative ideas and sound practical skill, and to shape a future where technology works for the enrichment of mankind.

Institute Mission

To impart state-of-the-art knowledge to individuals in various technological disciplines and to inculcate in them a high degree of social consciousness and human values, thereby enabling them to face the challenges of life with courage and conviction.

Department Vision

To evolve into a centre of excellence in electronics and communication engineering, moulding professionals having inquisitive,innovative and creative minds with sound practical skills who can strive for the betterment of mankind.

Department Mission

To impart state-of-the-art knowledge to students in Electronics and Communication Engineering and to inculcate in them a high degree of social consciousness and a sense of human values, thereby enabling them to face challenges with courage and conviction.

Programme Outcomes (PO)

Engineering Graduates will be able to:

1. Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

4. Conduct investigations of complex problems: Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

5. Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

9. Individual and Team work: Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.

10. Communication: Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports documentation. Make effective presentations, and give and receive clear instructions.

11. Project management and finance: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments.

12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of technological change.

Programme Specific Outcomes (PSO)

Engineering students will be able to:

PSO 1: Demonstrate their skills in designing, implementing and testing analogue and digital electronic circuits, including microprocessor systems, for signal processing, communication, networking, VLSI, and embedded systems applications.

PSO 2: Apply their knowledge and skills to conduct experiments and develop applications using electronic design automation (EDA) tools.

PSO 3: Demonstrate a sense of professional ethics, recognize the importance of continued learning, and be able to carry out their professional and entrepreneurial responsibilities in the electronics engineering field, giving due consideration to environmental protection and sustainability.

Course Outcomes (CO)

After successful completion of the course, the students will be able to:

Course Outcome 1: Be able to practice acquired knowledge within the selected area of technology for project development.

Course Outcome 2: Identify, discuss, and justify the technical aspects and design aspects of the project with a systematic approach.

Course Outcome 3: Reproduce, improve, and refine technical aspects for engineering projects.

Course Outcome 4: Work as a team in the development of technical projects.

Course Outcome 5: Communicate and report effectively project-related activities and findings.

Project Objectives

Project Objective 1: Design and develop an AI-based plant disease classification system using deep learning models like DenseNet121 and ResNet50 to improve agricultural productivity.(Cognitive knowledge level: Apply)

Project Objective 2: Implement a real-time image acquisition and processing system using an web-Cam module and Raspberry Pi to monitor plant health and automate

farming decisions.(Cognitive knowledge level: Apply)

Project Objective 3: Automate essential agricultural tasks such as irrigation, fertilizer spraying, moisture sensing, and weather monitoring using a FarmBot system integrated with machine learning models.(Cognitive knowledge level: Apply)

Project Objective 4: Analyze and compare the performance of different deep learning models (DenseNet121 vs. ResNet50) on varying dataset sizes to optimize classification accuracy.(Cognitive knowledge level: Analyze)

Project Objective 5: Integrate AI, and embedded systems to create a sustainable and scalable smart farming solution that minimizes human intervention and maximizes efficiency.(Cognitive knowledge level: Apply)

Project Objective 6: Document and present research findings, model performance comparisons, and technical implementations effectively through structured reports and presentations.(Cognitive knowledge level: Apply).

Project Outcomes

Project Outcome 1: Demonstrate the ability to analyze agricultural challenges and apply AI-based solutions for plant disease classification and precision farming.

Project Outcome 2: Design and develop an automated farming system integrating deep learning, IoT, and embedded systems for real-time monitoring and decision-making.

Project Outcome 3: Optimize the accuracy of plant disease classification by evaluating and selecting the best-performing deep learning models for different dataset sizes.

Project Outcome 4: Collaborate effectively in an interdisciplinary team to design, implement, and evaluate a smart agricultural system with automation capabilities.

Project Outcome 5: Present technical methodologies, system architecture, and experimental findings effectively through structured documentation and professional presentations.

Appendix C: CO-PO-PSO Mapping,PRO-PO and PRO-PSO Mapping

Table 7.1: CO - PO - PSO Mapping

CO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
1	3	2	2	-	-	-	-	-	-	-	-	-	3	2	-
2	3	3	2	2	3	-	-	-	-	-	-	-	3	3	-
3	2	2	3	-	3	-	2	2	-	-	-	2	3	2	2
4	3	3	3	2	3	-	-	-	-	-	-	2	3	2	2
5	-	-	-	-	-	-	-	-	3	3	2	2	-	-	2

Table 7.2: Justification for CO - PO - PSO Mapping Table

Mapping	Justification
CO1 - PO1	Uses engineering fundamentals to implement deep learning models for disease detection.
CO1 - PO2	Analyzes plant health problems using model accuracy and confusion matrices.
CO1 - PO3	Designs and implements plant disease detection pipelines suited for real-world scenarios.
CO1 - PSO1	Involves creating digital classification logic that mimics electronic circuit behavior.
CO1 - PSO2	Uses simulation tools and code editors to implement and debug AI models.
CO2 - PO1	Applies domain-specific AI knowledge to build robust disease detection architectures.
CO2 - PO2	Justifies dataset selection, preprocessing, and model design from a technical perspective.
CO2 - PO3	Uses systematic approach in pipeline creation, from input to classification.
CO2 - PO4	Compares DenseNet and ResNet model accuracy and draws conclusions based on metrics.
CO2 - PO5	Uses TensorFlow, OpenCV, and Raspberry Pi — all modern engineering tools.
CO2 - PSO1	Applies knowledge of signal processing (image) in an embedded environment.
CO2 - PSO2	Involves programming and testing using IDEs and data visualization libraries.
CO3 - PO3	Integrates software (model output) with hardware (FarmBot actuators).
CO3 - PO5	Implements real-time control logic using microcontroller interfacing.
CO3 - PO7	Ensures minimal water/fertilizer usage via data-driven automation.
CO3 - PO8	Maintains ethical practices in avoiding excessive automation or harming ecology.
CO3 - PO12	Encourages continual exploration of AI-agriculture integration.
CO3 - PSO1	Connects ML inference with physical components, like pumps and sensors.
CO3 - PSO2	Uses tools to debug, log, and improve real-time control programs.
CO3 - PSO3	Focuses on sustainable farm practices and professional responsibilities.
CO3 - PO1	Applies embedded systems knowledge and deep learning integration to translate model predictions into real-world actions via FarmBot.
CO3 - PO2	Analyzes timing delays, control logic, and sensor feedback to ensure precise irrigation and spraying.
CO4 - PO1	Applies multidisciplinary knowledge to synchronize sensors, data, and decisions.
CO4 - PO2	Analyzes environmental data to decide irrigation/fertilization triggers.
CO4 - PO3	Automates responses to data inputs through integrated control logic.
CO4 - PO4	Tests hardware responses, refines timing, and measures success via experiments.
CO4 - PO5	Uses real-time dashboards and automation tools for debugging and execution.
CO4 - PO12	Encourages upskilling in AIoT and hardware interfacing.

Table 7.3: Justification for CO - PO - PSO Mapping Table

Mapping	**Justification**
CO4 - PSO1	Coordinates digital decisions with embedded controls in FarmBot.
CO4 - PSO2	Modifies automation rules and tests responsiveness using tools.
CO4 - PSO3	Advocates smart water management aligned with environmental goals.
CO5 - PO9	Develops teamwork through division of tasks: hardware, software, documentation.
CO5 - PO10	Presents outcomes, graphs, and live demos in structured formats.
CO5 - PO11	Manages code versions, project plans, and task dependencies.
CO5 - PO12	Learns to use presentation tools, report formats, and document pipelines.
CO5 - PSO3	Emphasizes ethical responsibility in sharing accurate and useful knowledge.

Table 7.4: PRO - PO - PSO Mapping

PRO	**PO 1**	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**	**PSO 1**	**PSO 2**	**PSO 3**
1	3	3	3	2	3	-	2	2	-	-	-	-	3	3	2
2	3	2	3	2	3	-	-	-	-	-	-	2	3	2	3
3	2	2	3	-	3	-	3	2	-	-	-	2	3	2	2
4	3	3	2	3	3	-	-	-	-	-	-	2	3	2	-
5	3	3	3	2	3	-	3	3	-	-	2	3	3	2	3
5	2	2	2	-	2	-	-	-	2	3	3	2	2	-	2

Table 7.5: Justification for PRO - PO - PSO Mapping Table

Mapping	**Justification**
PRO1 - PO1	Applies foundational AI concepts like convolution, feature maps, and classification.
PRO1 - PO2	Frames disease detection as a technical problem needing optimization and accuracy.
PRO1 - PO3	Designs classification systems that are practical, scalable, and user-friendly.
PRO1 - PO4	Benchmarks DL model performance through precision, recall, and confusion matrix.
PRO1 - PO5	Implements using advanced tools: Jupyter, Python libraries, Raspberry Pi.
PRO1 - PO7	Reduces pesticide overuse by accurately targeting diseased plants.
PRO1 - PO8	Upholds AI model transparency and fair use in agriculture.
PRO1 - PSO1	Implements AI logic mimicking signal processing in digital systems.
PRO1 - PSO2	Trains, tests, and tunes the classifier using custom scripts and datasets.
PRO1 - PSO3	Advances sustainable agriculture using automation and accurate diagnosis.
PRO2 - PO1	Uses engineering fundamentals to control camera feed and analyze frames.
PRO2 - PO2	Frames problem as a design challenge — balancing performance and resource use.
PRO2 - PO3	Builds a real-time monitoring system suited to open field environments.
PRO2 - PO4	Verifies detection timing and image clarity through iterative testing.
PRO2 - PO5	Integrates OpenCV with Raspberry Pi for live detection.
PRO2 - PO12	Develops skill in integrating low-level (Pi) and high-level (Python) codebases.
PRO2 - PSO2	Calibrates resolution, framerate, and brightness using EDA techniques.
PRO2 - PSO3	Encourages minimal waste and optimal resource use through smart tech.
PRO3 - PO3	Automates tasks based on health status — disease triggers irrigation/fertilizer.
PRO3 - PO5	Uses GPIO and FarmBot API to trigger physical actions.
PRO3- PO7	Saves water and avoids over-irrigation via data-driven control.
PRO3 - PO8	Follows ethical design: alerts before action, avoids blind automation.
PRO3- PO12	Learns about IoT automation and MQTT/cloud integrations.
PRO3 - PSO1	Combines control algorithms with electrical output signals.
PRO3 - PSO2	Uses IDEs to debug automation flow and sensor readings.
PRO3 - PSO3	Reinforces environmental goals with smart agriculture.
PRO4 - PO1	Uses core statistics and ML knowledge to evaluate model success.
PRO4 - PO2	Analyzes training history, loss curves, and performance trade-offs.

Table 7.6: Justification for PRO - PO - PSO Mapping Table

Mapping	**Justification**
PRO4 - PO3	Designs experiments to benchmark DenseNet vs ResNet.
PRO4 - PO4	Interprets test results to guide final model selection.
PRO4 - PO5	Uses tools like Matplotlib, Pandas, TensorBoard.
PRO4 - PO12	Promotes critical thinking and experimentation.
PRO4 - PSO1	Ties model accuracy to real-time impact on embedded systems.
PRO4 - PSO2	. Refines and tests different versions of the models.
PRO5 - PO1	Fuses AI, electronics, and control logic into a single solution.
PRO5 - PO2	Identifies gaps in traditional farming and justifies smart system development.
PRO5 - PO3	Designs scalable layout for deployment beyond lab setup.
PRO5 - PO4	Validates performance under varying crop/lighting conditions.
PRO5 - PO5	Implements dashboards, sensor interfaces, mobile alerts.
PRO5 - PO7	Emphasizes low environmental impact with precise automation.
PRO5 - PO8	Avoids misuse of AI, respects farmer decisions and transparency.
PRO5 - PO11	Uses task boards, cost analysis, and workload distribution.
PRO5 - PO12	Motivates students to follow evolving agri-tech trends.
PRO5 - PSO1	Fully integrates technical skills, experimentation, and ethical values.
PRO6 - PO9	Encourages team planning, division of roles, and group collaboration.
PRO6 - PO10	Documents outcomes using charts, photos, diagrams, and clear language.
PRO6 - PO11	Applies planning, scheduling, and timeline tracking for presentations.
PRO6 - PO12	Uses new documentation tools and improves technical writing.
PRO6 - PSO3	Demonstrates professional integrity through high-quality documentation.

Appendix D: Code

Complete Smart Farming System Python Script

```
import RPi.GPIO as GPIO
import cv2
import numpy as np
import tensorflow as tf
import time
import threading

# Set GPIO mode
GPIO.setmode(GPIO.BCM)

# Define motor control pins
ENA = 25  # Enable A (PWM)
ENB = 26  # Enable B (PWM)
IN1 = 24  # Motor A Forward
IN2 = 23  # Motor A Reverse
IN3 = 22  # Motor B Forward
IN4 = 27  # Motor B Reverse

# DHT11 sensor pin
DHT_PIN = 4

# Soil moisture sensor pin (digital input)
MOISTURE_PIN = 17

# Relay pins
RELAY1_PIN = 5  # Fertilizer pump
RELAY2_PIN = 6  # Water pump

# Servo motor pin
SERVO_PIN = 18

# Timing constants
FERTILIZER_TIME = 8.0  # Seconds to run fertilizer pump
WATER_PUMP_TIME = 8.0  # Seconds to run water pump

# Variables to store temperature and humidity readings
current_temperature = 0
current_humidity = 0
last_dht_reading_time = 0
dht_reading_interval = 2.0  # Read DHT11 every 2 seconds

# Set up GPIO pins
GPIO.setup(ENA, GPIO.OUT)
GPIO.setup(ENB, GPIO.OUT)
GPIO.setup(IN1, GPIO.OUT)
GPIO.setup(IN2, GPIO.OUT)
```

```
GPIO.setup(IN3, GPIO.OUT)
GPIO.setup(IN4, GPIO.OUT)
GPIO.setup(MOISTURE_PIN, GPIO.IN)
GPIO.setup(RELAY1_PIN, GPIO.OUT)
GPIO.setup(RELAY2_PIN, GPIO.OUT)
GPIO.setup(SERVO_PIN, GPIO.OUT)

# Initialize relay pins to LOW (off)
GPIO.output(RELAY1_PIN, GPIO.LOW)
GPIO.output(RELAY2_PIN, GPIO.LOW)

# Create PWM objects for motor speed control
motor_pwm_a = GPIO.PWM(ENA, 1000)  # Frequency = 1 kHz
motor_pwm_b = GPIO.PWM(ENB, 1000)  # Frequency = 1 kHz

# Create PWM object for servo motor
servo_pwm = GPIO.PWM(SERVO_PIN, 50)  # 50 Hz (standard for servos)
servo_pwm.start(7.5)  # Initialize to middle position (90 degrees)

# Task status flags
task1_complete = False  # Fertilizer task
task2_complete = False  # Soil moisture sensing and irrigation task

# Motor control functions
def start_motors(speed=70):
    """Start motors with specified speed"""
    motor_pwm_a.start(speed)
    motor_pwm_b.start(speed)
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)
    print("Motors moving forward...")

def stop_motors():
    """Stop motors"""
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.LOW)
    print("Motors stopped.")

# Servo control function
def set_servo_angle(angle):
    """Set servo angle (0-180 degrees)"""
    duty = angle / 18 + 2.5  # Convert angle to duty cycle (2.5-12.5)
    servo_pwm.ChangeDutyCycle(duty)
    time.sleep(0.5)  # Give servo time to move
```

```
    print(f"Servo moved to {angle} degrees")

# Soil moisture reading function
def read_soil_moisture():
    """Read soil moisture sensor"""
    # For digital moisture sensor - returns 0 when wet, 1 when dry
    # Adjust logic as needed based on your specific sensor
    is_dry = GPIO.input(MOISTURE_PIN)
    moisture_status = "DRY" if is_dry else "WET"
    print(f"Soil moisture: {moisture_status}")
    return is_dry  # True if dry, False if wet

# Task 1: Run fertilizer pump for specified time
def run_fertilizer_task():
    global task1_complete

    print("Starting fertilizer application...")
    GPIO.output(RELAY1_PIN, GPIO.HIGH)  # Turn on fertilizer pump
    time.sleep(FERTILIZER_TIME)  # Run for specified time
    GPIO.output(RELAY1_PIN, GPIO.LOW)   # Turn off fertilizer pump
    print("Fertilizer application complete")

    task1_complete = True

# Task 2: Servo positioning, soil moisture sensing and irrigation
def run_irrigation_task():
    global task2_complete

    # Move servo to 90 degrees
    set_servo_angle(90)
    print("Servo positioned for soil sensing")

    # Check soil moisture
    is_soil_dry = read_soil_moisture()

    # If soil is dry, activate water pump
    if is_soil_dry:
        print("Soil is dry, activating irrigation")
        GPIO.output(RELAY2_PIN, GPIO.HIGH)  # Turn on water pump
        time.sleep(WATER_PUMP_TIME)  # Run for specified time
        GPIO.output(RELAY2_PIN, GPIO.LOW)   # Turn off water pump
        print("Irrigation complete")
    else:
        print("Soil moisture adequate, no irrigation needed")

    # Return servo to forward position
    set_servo_angle(0)
    print("Servo returned to forward position")
```

```
143
144     task2_complete = True
145
146 # Direct DHT11 sensor reading function
147 def read_dht11_data():
148     global current_temperature, current_humidity
149
150     # Use the standard Raspberry Pi GPIO library method
151     data = []
152     j = 0
153
154     # Set GPIO pin to output and set to low
155     GPIO.setup(DHT_PIN, GPIO.OUT)
156     GPIO.output(DHT_PIN, GPIO.LOW)
157     time.sleep(0.018)  # Wait 18ms (minimum pull-down time for DHT11)
158
159     # Set to input with pull-up
160     GPIO.setup(DHT_PIN, GPIO.IN, pull_up_down=GPIO.PUD_UP)
161
162     # Wait for response from DHT11 (low, then high, then data)
163     # First expect LOW
164     count = 0
165     while GPIO.input(DHT_PIN) == GPIO.HIGH:
166         count += 1
167         if count > 100:
168             return False
169
170     # Then expect HIGH
171     count = 0
172     while GPIO.input(DHT_PIN) == GPIO.LOW:
173         count += 1
174         if count > 100:
175             return False
176
177     count = 0
178     while GPIO.input(DHT_PIN) == GPIO.HIGH:
179         count += 1
180         if count > 100:
181             return False
182
183     # Read 40 bits of data (5 bytes)
184     for i in range(40):
185         count = 0
186         while GPIO.input(DHT_PIN) == GPIO.LOW:
187             count += 1
188             if count > 100:
189                 return False
190
```

```
        # Measure high time to determine bit value
        count = 0
        start = time.time()
        while GPIO.input(DHT_PIN) == GPIO.HIGH:
            count += 1
            if count > 100:
                break

        # Determine bit value based on pulse width
        if (time.time() - start) > 0.00004:  # 40  s   threshold
            data.append(1)
        else:
            data.append(0)

    # Convert bits to bytes
    humidity_int = 0
    for i in range(8):
        humidity_int += data[i] << (7-i)

    humidity_decimal = 0
    for i in range(8, 16):
        humidity_decimal += data[i] << (15-i)

    temperature_int = 0
    for i in range(16, 24):
        temperature_int += data[i] << (23-i)

    temperature_decimal = 0
    for i in range(24, 32):
        temperature_decimal += data[i] << (31-i)

    checksum = 0
    for i in range(32, 40):
        checksum += data[i] << (39-i)

    # Verify checksum
    calculated_checksum = (humidity_int + humidity_decimal +
temperature_int + temperature_decimal) & 0xFF

    if checksum == calculated_checksum:
        humidity = humidity_int + humidity_decimal * 0.1
        temperature = temperature_int + temperature_decimal * 0.1

        # DHT11 typically doesn't use decimal part, so just use integer
part
        current_humidity = humidity_int
        current_temperature = temperature_int

```

```
        print(f"Temperature: {current_temperature}  C    Humidity: {current_humidity}%")
        return True
    else:
        print("Checksum error in DHT11 reading")
        return False

# Load the TFLite model
interpreter = tf.lite.Interpreter(model_path="/home/ras6/plant-detection-env/prediction_disease_raspberrypi/model1.tflite")
interpreter.allocate_tensors()

# Get input/output tensor details
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Initialize the webcam
cap = cv2.VideoCapture(0)

# Define class names (same as used in training)
class_names = [
    'Apple___Apple_scab', 'Apple___Black_rot', 'Apple___Cedar_apple_rust', 'Apple___healthy',
    'Blueberry___healthy', 'Cherry_(including_sour)___Powdery_mildew', 'Cherry_(including_sour)___healthy',
    'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot', 'Corn_(maize)___Common_rust_',
    'Corn_(maize)___Northern_Leaf_Blight', 'Corn_(maize)___healthy', 'Grape___Black_rot',
    'Grape___Esca_(Black_Measles)', 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)', 'Grape___healthy',
    'Orange___Haunglongbing_(Citrus_greening)', 'Peach___Bacterial_spot', 'Peach___healthy',
    'Pepper,_bell___Bacterial_spot', 'Pepper,_bell___healthy', 'Potato___Early_blight',
    'Potato___Late_blight', 'Potato___healthy', 'Raspberry___healthy', 'Soybean___healthy',
    'Squash___Powdery_mildew', 'Strawberry___Leaf_scorch', 'Strawberry___healthy',
    'Tomato___Bacterial_spot', 'Tomato___Early_blight', 'Tomato___Late_blight', 'Tomato___Leaf_Mold',
    'Tomato___Septoria_leaf_spot', 'Tomato___Spider_mites Two-spotted_spider_mite',
    'Tomato___Target_Spot', 'Tomato___Tomato_Yellow_Leaf_Curl_Virus', 'Tomato___Tomato_mosaic_virus',
    'Tomato___healthy'
]

```

```
# Define classes to ignore
ignored_classes = [
    'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
    'Corn_(maize)___Common_rust_',
    'Corn_(maize)___Northern_Leaf_Blight',
    'Corn_(maize)___healthy',
    'Strawberry___healthy',
    'Strawberry___Leaf_scorch'
]

# Lower the initial confidence threshold to catch more potential diseases
initial_confidence_threshold = 0.65

# Define confirmation parameters
required_consecutive_detections = 4  # Number of consecutive frames needed
    to confirm a disease
confirmation_threshold = 0.75  # Higher threshold for confirmation

try:
    # Start motors initially
    start_motors()

    # Disease detection variables
    disease_detected = False
    tasks_started = False
    detection_count = 0
    last_detected_class = None
    detection_history = []  # Store last few predictions

    # Initial servo position (0 degrees - forward)
    set_servo_angle(0)

    # Start real-time video capture
    while True:
        current_time = time.time()

        # Read from DHT11 sensor at specified intervals
        if current_time - last_dht_reading_time >= dht_reading_interval:
            try:
                # Make several attempts to read the DHT11
                for _ in range(3):  # Try up to 3 times
                    if read_dht11_data():
                        break
                    time.sleep(0.5)
            except Exception as e:
                print(f"Error reading DHT11: {e}")

            last_dht_reading_time = current_time
```

```
318
319         # Capture and process frame
320         ret, frame = cap.read()
321         if not ret:
322             break
323
324         # Process image even during waiting period (for display purposes)
325         # Preprocess the frame for prediction (resize to 128x128)
326         image = cv2.resize(frame, (128, 128))
327         image = np.expand_dims(image, axis=0)
328         image = image.astype(np.float32)  # Convert to float32 as required
    by TFLite
329
330         # Set the input tensor
331         interpreter.set_tensor(input_details[0]['index'], image)
332
333         # Run inference
334         interpreter.invoke()
335
336         # Get the prediction output
337         output = interpreter.get_tensor(output_details[0]['index'])
338
339         # Get top 3 predictions for more robust detection
340         top_indices = np.argsort(output[0])[-3:][::-1]
341         predictions = [(class_names[i], output[0][i]) for i in top_indices]
342
343         predicted_class = class_names[np.argmax(output)]
344         confidence_score = np.max(output)
345
346         # Display weather information on the frame
347         cv2.putText(frame, f"Temp: {current_temperature}  C  ",
348                     (10, frame.shape[0] - 90), cv2.FONT_HERSHEY_SIMPLEX,
    0.7, (0, 255, 0), 2)
349         cv2.putText(frame, f"Humidity: {current_humidity}%",
350                     (10, frame.shape[0] - 60), cv2.FONT_HERSHEY_SIMPLEX,
    0.7, (0, 255, 0), 2)
351
352         # Display the top 3 predictions
353         y_pos = 30
354         for i, (pred_class, pred_score) in enumerate(predictions):
355             cv2.putText(frame, f"{i+1}. {pred_class}: {pred_score:.2f}",
356                         (10, y_pos + i*30), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
    (255, 0, 0), 2)
357
358         # Handle disease detection workflow
359         if not disease_detected:
360             # Check if prediction is valid (not in ignored classes and
    above threshold)
```

```
            is_valid_prediction = (predicted_class not in ignored_classes
and
                                   confidence_score >=
initial_confidence_threshold)

            # Update detection history
            if is_valid_prediction:
                # Add current prediction to history
                detection_history.append((predicted_class, confidence_score
))
                # Keep only the last 'required_consecutive_detections'
entries
                if len(detection_history) > required_consecutive_detections
:
                    detection_history.pop(0)

                # Check if we have enough consistent detections
                if len(detection_history) ==
required_consecutive_detections:
                    # Count occurrences of each class in history
                    class_counts = {}
                    for cls, score in detection_history:
                        if cls not in class_counts:
                            class_counts[cls] = []
                        class_counts[cls].append(score)

                    # Find most common class and its average score
                    most_common_class = None
                    max_count = 0
                    avg_score = 0

                    for cls, scores in class_counts.items():
                        if len(scores) > max_count:
                            max_count = len(scores)
                            most_common_class = cls
                            avg_score = sum(scores) / len(scores)

                    # If we have consistent detection above confirmation
threshold
                    if (max_count >= required_consecutive_detections * 0.8
and
                        avg_score >= confirmation_threshold):
                        # Stop motors when disease is confirmed
                        stop_motors()
                        disease_detected = True
                        tasks_started = False
                        task1_complete = False
                        task2_complete = False
```

```
401
402                     print(f"Disease confirmed: {most_common_class} with
     avg confidence {avg_score:.2f}")
403                     print(f"Weather at detection - Temp: {
    current_temperature}  C  , Humidity: {current_humidity}%")
404                     print("Initiating treatment and irrigation
    procedures")
405
406                     # Add a border to indicate disease detected
407                     cv2.rectangle(frame, (0, 0), (frame.shape[1], frame
    .shape[0]), (0, 0, 255), 10)
408             else:
409                 # Clear history if nothing detected
410                 detection_history = []
411
412             # Display detection status
413             if len(detection_history) > 0:
414                 cv2.putText(frame, f"Potential disease: {len(
    detection_history)}/{required_consecutive_detections}",
415                             (10, frame.shape[0] - 30), cv2.
    FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2)
416         else:
417             # When disease is detected, start both tasks in parallel
418             if not tasks_started:
419                 # Start both tasks in separate threads
420                 fertilizer_thread = threading.Thread(target=
    run_fertilizer_task)
421                 irrigation_thread = threading.Thread(target=
    run_irrigation_task)
422
423                 fertilizer_thread.start()
424                 irrigation_thread.start()
425
426                 tasks_started = True
427
428                 # Add task status to the frame
429                 cv2.putText(frame, "TASKS STARTED: Fertilizer & Irrigation"
    ,
430                             (10, frame.shape[0] - 30), cv2.
    FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255), 2)
431
432             # Display task status
433             task1_status = "COMPLETE" if task1_complete else "IN PROGRESS"
434             task2_status = "COMPLETE" if task2_complete else "IN PROGRESS"
435
436             cv2.putText(frame, f"Fertilizer: {task1_status}",
437                         (frame.shape[1] - 250, 70), cv2.FONT_HERSHEY_SIMPLEX
    , 0.8,
```

```
                        (0, 255, 0) if task1_complete else (0, 0, 255), 2)

            cv2.putText(frame, f"Irrigation: {task2_status}",
                        (frame.shape[1] - 250, 100), cv2.FONT_HERSHEY_SIMPLEX, 0.8,
                        (0, 255, 0) if task2_complete else (0, 0, 255), 2)

            # Check if both tasks are complete
            if task1_complete and task2_complete:
                # Resume operation
                disease_detected = False
                tasks_started = False
                start_motors()
                detection_count = 0
                detection_history = []
                print("All tasks completed. Resuming monitoring operation.")

        # Add status indicator
        if disease_detected:
            status_text = "Status: TREATING"
        else:
            status_text = "Status: MONITORING"

        cv2.putText(frame, status_text, (frame.shape[1] - 250, 30),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 0, 255) if disease_detected else (0, 255, 0), 2)

        # Show the frame with the prediction
        cv2.imshow("Disease Detection & Treatment System", frame)

        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

except KeyboardInterrupt:
    print("Program terminated by user")

finally:
    # Clean up resources
    motor_pwm_a.stop()
    motor_pwm_b.stop()
    servo_pwm.stop()
    GPIO.output(RELAY1_PIN, GPIO.LOW)  # Ensure pumps are off
    GPIO.output(RELAY2_PIN, GPIO.LOW)  # Ensure pumps are off
    GPIO.cleanup()
    cap.release()
    cv2.destroyAllWindows()
```

```
483     print("Resources released. Program ended.")
```

Listing 7.1: Smart Farming Python Code

Appendix E: Presentation

SMART FARMING SYSTEM

GUIDED BY : MS.S.SANTHI JABARANI

DONE BY :

ALKA DENNY(U2101023)
ANITHRA ROSS AJITH(U2101035)
ANNA JOJU(U2101039)
ANU XAVIER(U2101042)

CONTENTS

INTRODUCTION

- Farming is entering a new era where technology works hand in hand with nature to create more efficient and sustainable practices.
- This smart farming system combines advanced tools to monitor plant health and automate care.
- By detecting leaf diseases through image analysis and delivering precise irrigation and fertilizers, it ensures crops receive the exact attention they need to thrive.
- With its ability to take real-time actions based on plant conditions, this system not only improves productivity but also reduces resource usage, paving the way for a smarter and greener approach to agriculture.

LITERATURE SURVEY

PAPER TITLE	AUTHORS	INFERENCE	DRAWBACKS
Smart farming using Machine Learning and Deep Learning techniques(2022)	Senthil Kumar Swami Durai, Mary Divya Shamili	The paper explores machine learning and deep learning for smart farming, enhancing crop health monitoring, disease detection, and resource optimization to improve productivity and sustainability through automation and precision farming.	Lacks real-time implementation challenges, high computational costs, and scalability issues for large-scale farms with diverse environmental conditions.
A comparative study of fine-tuning deep learning models for plant disease identification.(2019)	Edna Chebet Tooa, Li Yujian, Sam Njuki, Liu Yingchun	This study compares fine-tuning techniques on deep learning models for identifying plant diseases, finding that tailored fine-tuning significantly improves model accuracy across various plant disease datasets.	The study is limited by a lack of diversity in disease types and environmental conditions. Additionally, it doesn't assess model performance under real-time conditions, which can differ significantly from the controlled settings used in training.

LITERATURE SURVEY

PAPER TITLE	AUTHORS	INFERENCE	DRAWBACKS
Multiple plant leaf disease classification using densenet-121 architecture (2021)	Swaminathan, Akshay, C. Varun, and S. Kalaivani	Using the DenseNet-121 architecture, this paper demonstrates effective classification of multiple plant leaf diseases, showing DenseNet's efficiency in extracting features from complex disease patterns.	The study's primary limitation lies in the high computational resources required for DenseNet-121, which may not be practical for resource-constrained environments. It also doesn't evaluate the model's robustness against noise or varied lighting conditions in field scenarios.
Transfer learning based plant diseases detection using ResNet50(2019)	Mukti, Ishrat Zahan, and Dipayan Biswas	This paper explores transfer learning using ResNet-50 for detecting plant diseases, indicating that transfer learning boosts model performance, especially when data availability is limited.	The model's accuracy might decline in diverse field conditions due to the limited dataset diversity. Additionally, ResNet-50's memory and computation demands could restrict its use on low-resource devices in agricultural applications.

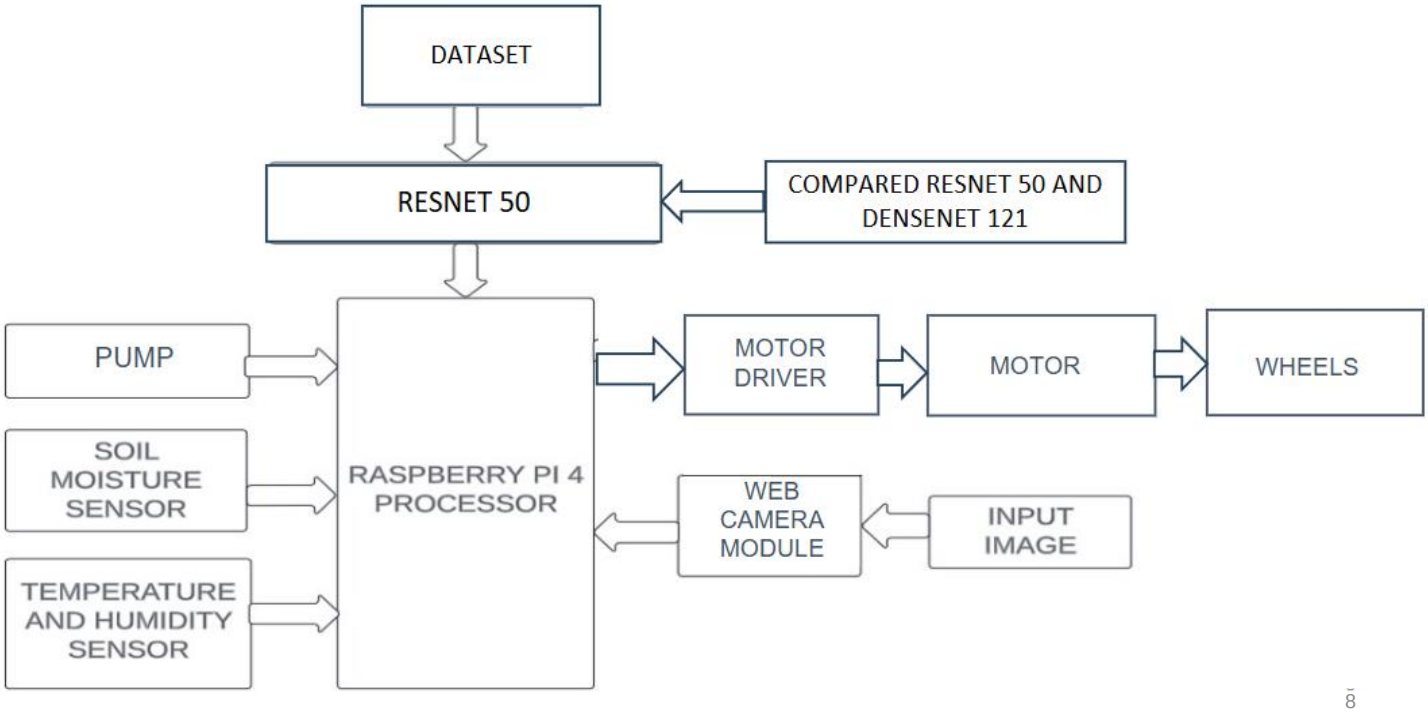
PROBLEM STATEMENT

- **Inefficient Disease Detection** – Traditional methods are labor-intensive, error-prone, and slow, making timely disease management difficult.
- **Resource Optimization Challenge** – Inefficient use of water and fertilizers increases costs and reduces crop yields, highlighting the need for an automated solution.

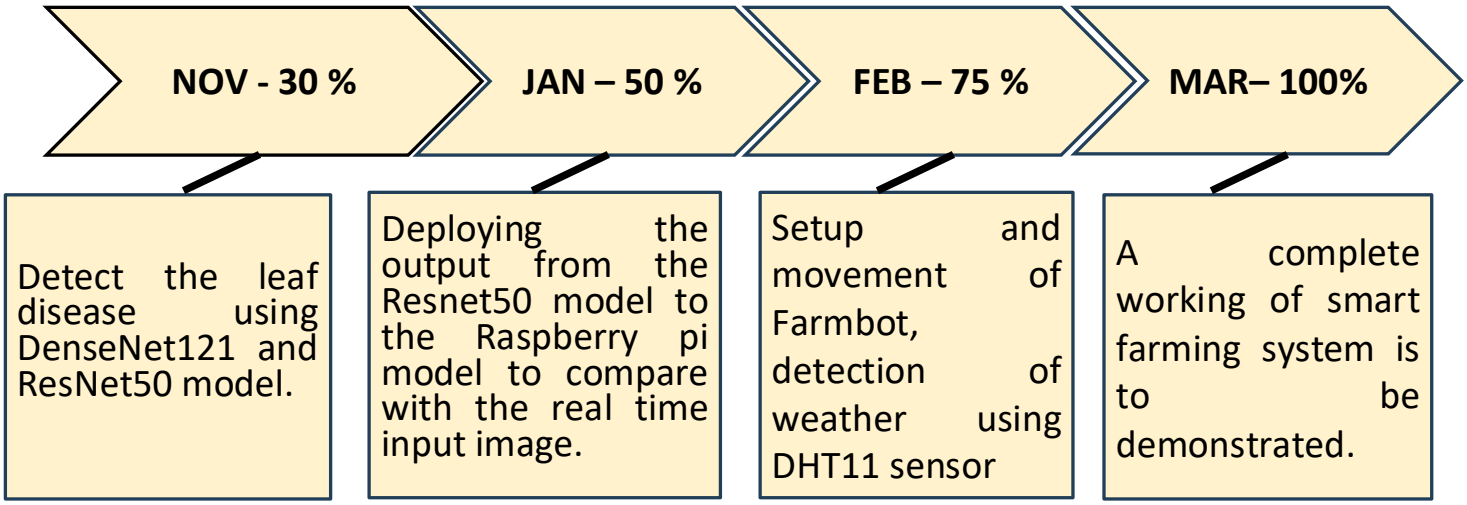
OBJECTIVES

- **Advanced Leaf Disease Detection** – Develop a robust system using DenseNet-121 and ResNet-50 for accurate identification of leaf diseases.
- **Automated Irrigation System** – Implement a smart irrigation system that optimizes water usage by adjusting supply based on plant needs.
- **Smart Resource Management & Environmental Monitoring** – Automate fertilizer application based on plant health assessments from the disease detection model, while integrating temperature and weather sensing to provide real-time updates and take necessary actions for improved crop management.

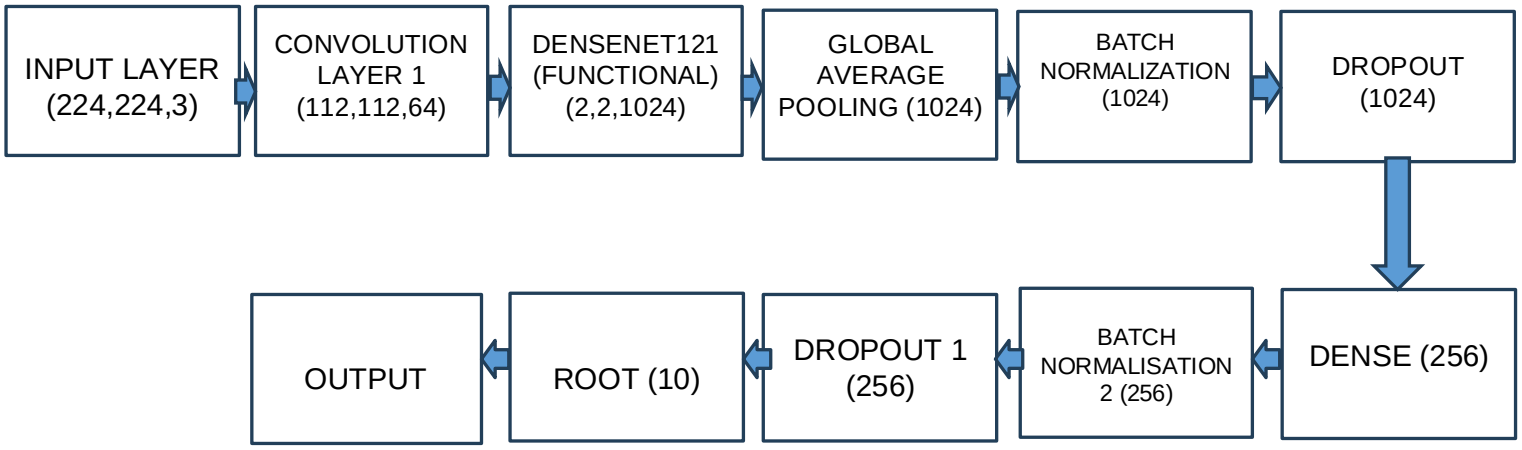
BLOCK DIAGRAM



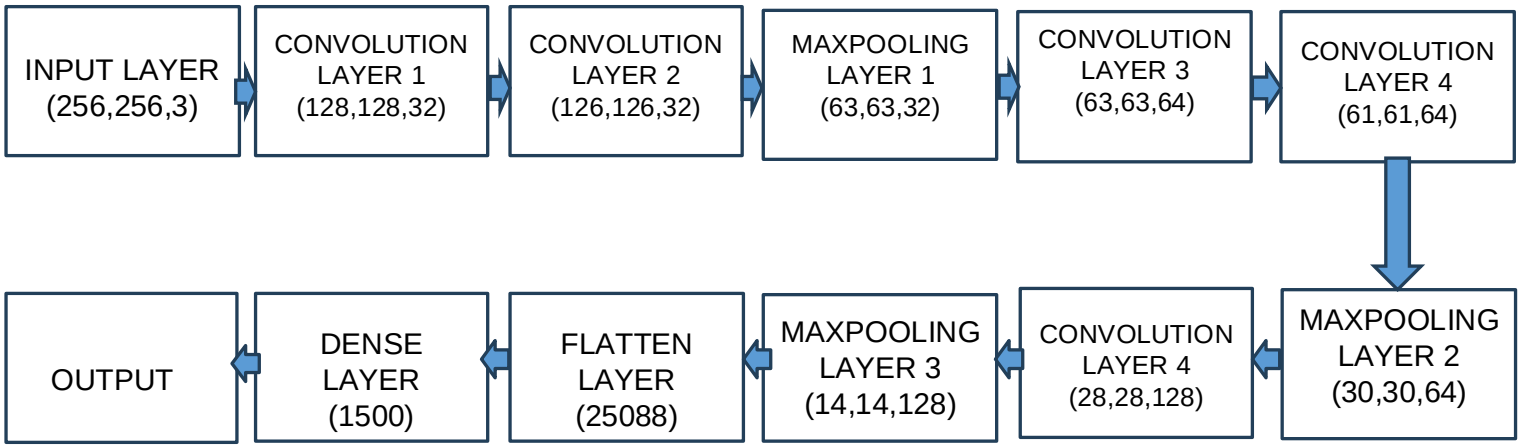
WORK SPLIT - UP



DENSENET121 ARCHITECTURE



RESNET50 ARCHITECTURE



COMPARISON

Aspect	DenseNet-121	ResNet-50
Architecture	Consists of 121 layers with *dense connections* between layers, where each layer takes inputs from all preceding layers.	Consists of 50 layers with *residual connections*, where layers have skip connections, allowing gradients to flow more smoothly.
Number of Parameters	Approximately 8 million parameters due to efficient feature reuse.	Approximately 25 million parameters, as residual connections lead to fewer feature reuse opportunities.
Computational Efficiency	Slightly higher memory requirements due to dense connections, but fewer parameters improve parameter efficiency.	Typically requires more memory due to larger parameter count, though computations are simpler with residual connections.
Accuracy	Accuracy on ImageNet is slightly lower, generally around 74.9% top-1 accuracy.	Higher accuracy on ImageNet, typically around 76.2% top-1 accuracy, making it slightly more effective for large-scale datasets.

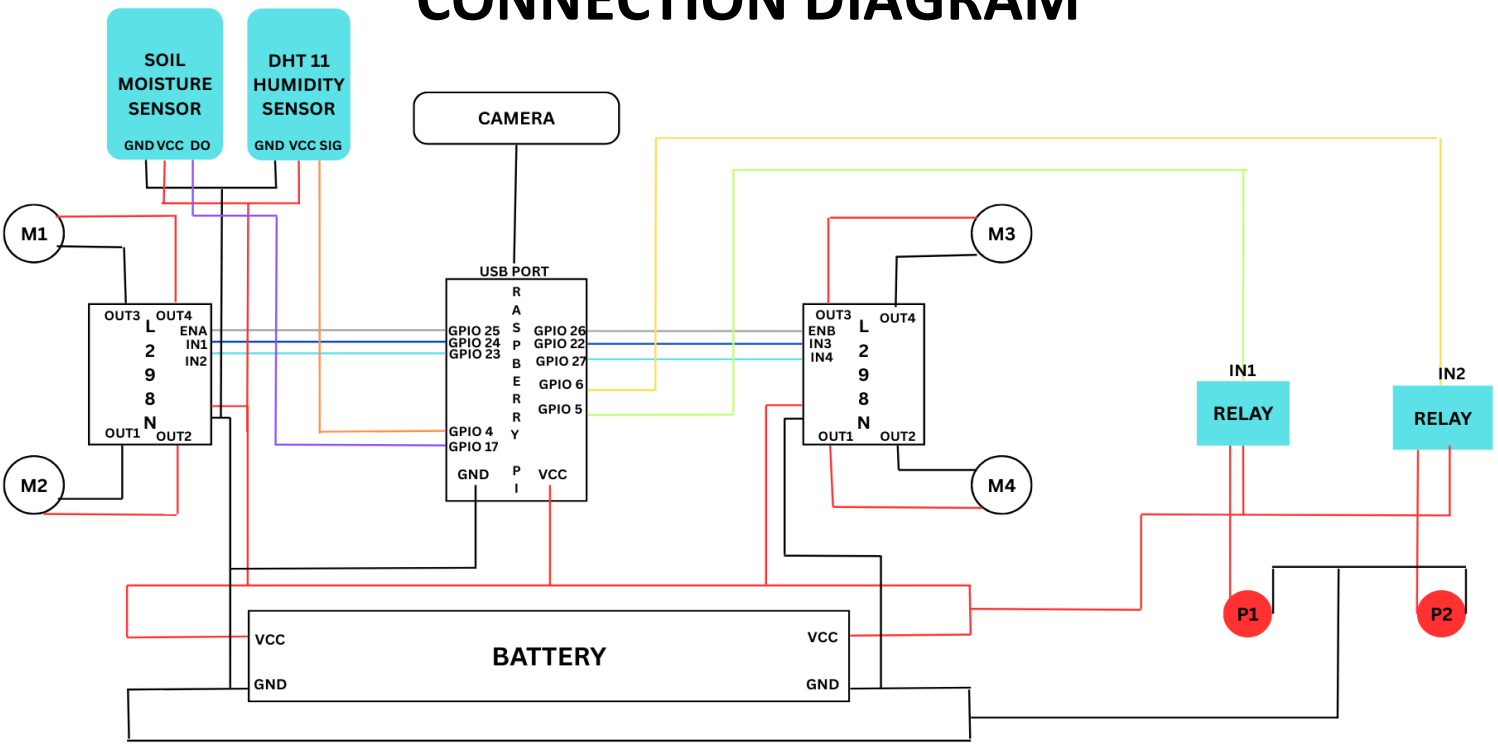
Why we choose Resnet50 over Densenet121

- **Faster Training and Inference**:ResNet50 is less complex and faster than DenseNet121.
- **Lower Memory Usage**: Requires less GPU memory, making it suitable for limited resources.
- **Widely Used**: Pretrained models are widely available with excellent community support.
- **Scalability**: Easy to scale to larger models (e.g., ResNet101, ResNet152).
- **Better for Large Datasets**: Performs well on large-scale tasks due to its depth and efficiency.

HARDWARE DESCRIPTION

SL NO.	COMPONENTS LIST	COST
1	DC Motor(12 V,30 rpm,10 kg Torque)	1200 X 4= 4800
2	3.7 V Battery (4300MAH)	225 X 9= 2025
3	JQC-3FF- S- Z Relay	150 X 2 = 300
4	LM393 Soil Sensor	300
5	Jumper wires	150 x 4 = 600
6	PCB	85 x 2 = 370
7	Holder	200
8	Motor Driver L298N	350 x 2 = 700
9	DHT 11 sensor	225
10	Raspberry pi 4B 4GB	6000
11	Tire	51 x 4 = 204
12	Pipe and accessories	1500
13	12 V Immersed Water pump	250 x 2 = 500

CONNECTION DIAGRAM



METHODOLOGY

RASPBERRY PI 4B :

1. Sensor Data Processing

- Reads data from the Soil Moisture Sensor to determine soil dryness.
- Receives environmental readings from the DHT Humidity Sensor for monitoring.

2. Automated Motor & Relay Control

- M1,M2,M3,M4 are used for rotating the wheels.
- Sends signals to relay modules to spray fertilizer and irrigation when a disease is detected.

3. Camera-Based Disease Detection

- Captures leaf images using the **camera module**.
- Processes the images using Resnet50 to detect plant diseases.
- Triggers the relay system to spray fertilizer when disease is identified.

4. Communication & Decision Making

- Uses GPIO pins to send commands to motors, sensors, and relays.
- Acts as the brain of the system, making real-time decisions based on sensor inputs.

METHODOLOGY

DC MOTOR :

- **Purpose** : Enables the FarmBot to move along the track for plant monitoring, leaf detection, and spraying.
- **Wiring Connection :**
  - OUT1 → Motor A Terminal 1
  - OUT2 → Motor A Terminal 2
  - OUT3 → Motor B Terminal 1
  - OUT4 → Motor B Terminal 2
  - OUT1 → Motor C Terminal 1
  - OUT2 → Motor C Terminal 2
  - OUT3 → Motor D Terminal 1
  - OUT4 → Motor D Terminal 2
- Moves the Rover (12V DC),Supports Disease Detection ,Ensures Controlled Navigation, Handles Heavy Load ,Automates Farming Tasks, Enhances Efficiency.

METHODOLOGY

3.7 V BATTERY :

- Voltage: 3.7V DC – Powers the Raspberry Pi, sensors, and motor driver efficiently.
- Capacity: 4300mAh – Ensures long-lasting operation, reducing frequent recharging.
- Boost Converter Support: Steps up voltage to 12V for running the DC motor (30 RPM, 10 kg.cm torque).
- Sensor & Camera Power: Supports DHT11 sensor, web camera module, and motor driver for automation.
- Compact & Lightweight: Ideal for smooth mobility of the smart farming rover without adding excessive weight.
- Rechargeable & Cost-Effective: Lithium-ion technology allows multiple charge cycles, making it sustainable.

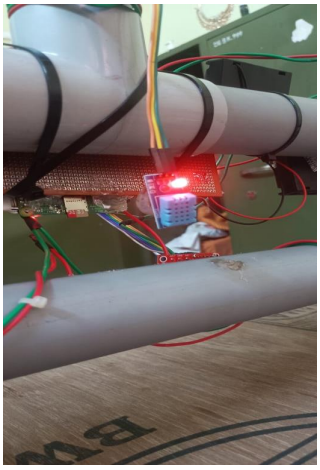
METHODOLOGY

JQC-3FF- S- Z Relay

- **Purpose :** Activates pumps P1 (fertilizer) and P2 (water) for 8 seconds each when needed.
- **Wiring connections :**
  - VCC → 5V (Raspberry Pi)
  - GND → GND (Raspberry Pi)
  - IN1 → GPIO5 (Controls Fertilizer Pump P1)
  - IN2 → GPIO6 (Controls Water Pump P2)
- **Working Principle**
  - A relay is an electromechanical switch that allows low-power GPIO signals to control high-power loads like pumps.
  - When GPIO5 or GPIO6 is HIGH, the relay closes the circuit, allowing current to flow to the pumps.
  - When the GPIO is LOW, the relay opens and stops the pumps.

METHODOLOGY

DHT11 SENSOR :



- **Purpose :** Monitors humidity and temperature for farm conditions.
- **Wiring connections:**
  - VCC → 5V (Raspberry Pi)
  - GND → GND (Raspberry Pi)
  - Data → GPIO4 (Temperature & Humidity Data)
- **Working Principle:**

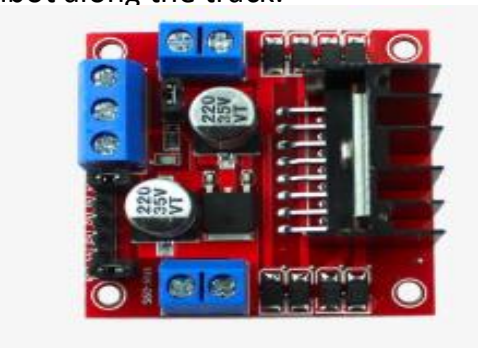
The sensor contains a thermistor and a humidity-sensitive capacitor.

- It measures temperature by detecting changes in resistance with temperature.
- It measures humidity based on electrical conductivity changes in moisture-sensitive material.
- Data is sent digitally to the Raspberry Pi via GPIO4.

METHODOLOGY

MOTOR DRIVER :

- **Purpose:** Controls four motors (M1, M2, M3, M4) to move the Farmbot along the track.
- **Wiring Connection :**
  - ENA → GPIO25 (PWM speed control for Motor A)
  - ENB → GPIO26(PWM speed control for Motor B)
  - IN1 → GPIO24 (Motor A Forward)
  - IN2 → GPIO2 (Motor A Reverse)
  - IN3 → GPIO22 (Motor B Forward)
  - IN4 → GPIO27 (Motor B Reverse)

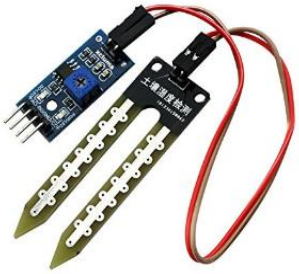


- **Working Principle :**
  - L298N is an H-Bridge motor driver that allows the Raspberry Pi to control the direction and speed of two DC motors.
  - The IN1, IN2, IN3, IN4 pins control the direction of the motors.
  - The ENA and ENB pins control the speed using Pulse Width Modulation (PWM) signals from the Raspberry Pi.

METHODOLOGY

LM393 SOIL MOISTURE SENSOR :

- **Purpose** : Detects if the soil is dry. If it is dry, triggers the water pump for 8 seconds.
- **Wiring Connections:**
  - VCC → 3.3V/5V (Raspberry Pi)
  - GND → GND (Raspberry Pi)
  - D0 (Digital Output) → Any GPIO 17(Raspberry Pi)
- **Working Principle:**
  - Dry Soil → D0 goes HIGH (1), triggers irrigation.
  - Moist Soil → D0 goes LOW (0), pump stays OFF.



RESULT AND DISSCUSSION

- IDE setup and installation of libraries
  - Python Libraries: numpy, keras, tensorflow, opencv, pandas, sklearn
- Downloading datasets
  - Dataset: Kaggle
  - Image: Jpg format
  - Classes:30 classes
- Resnet 50 model
  - 87,000 RGB samples were taken as dataset
  - Learning rate :0.0001
  - Epoch:10
  - Leaf disease detection using ResNet50 with the accuracy of 96.8%

RESULT AND DISCUSSION

- **Model Architecture**: Trained using ResNet-50 for robust image recognition.
- **Model Conversion**: Optimized the model by converting it to TensorFlow Lite for deployment on edge devices.
- **Deployment Platform**: Implemented the model on Raspberry Pi 4 for real-time image detection.
- **Hardware Integration**: Achieved image detection through a camera module.
- **Connectivity**: Enabled communication between Raspberry Pi 4 and laptop via Wi-Fi.

RESULT AND DISCUSSION

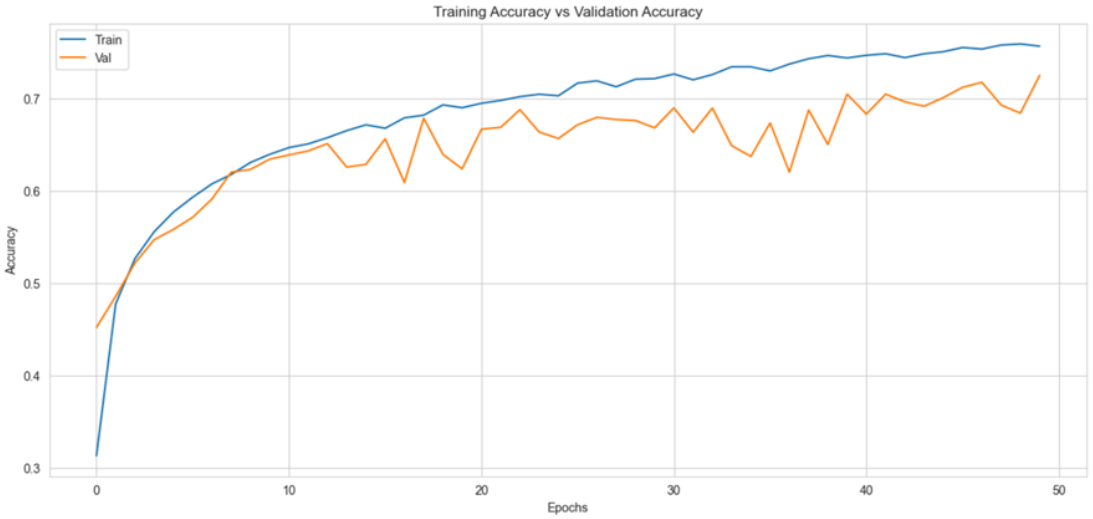
- **Camera & Raspberry Pi 4** – Camera is fixed on a rover, connected to Raspberry Pi 4, mounted on a PVC structure.
- **Movement & Detection** – The rover moves through the field to scan plants.
- **Disease Detection** – When a diseased leaf is detected, Captures and analyzes using the ResNet50 model
- **Action Based on Detection** – Uses analysis results for further processing.
- **DHT11 Sensor** – Measures real-time temperature and humidity.
- **Outcome** – Provides accurate environmental monitoring, aiding in better crop management

RESULT AND DISCUSSION

- **Soil Moisture Detection & Automated Irrigation**
  - Implemented soil moisture sensors to detect dryness.
  - Automated water sprinkling system activates when soil is dry.
- **Disease Detection & Fertilizer Spraying**
  - Used ResNet50 model to detect leaf diseases.
  - Automated fertilizer spraying system activates when disease is detected.
- **Fully Functional Smart Farming Model**
  - Integrated all components into a complete working model.
  - Used Raspberry Pi 4 with a web camera module for real-time monitoring.

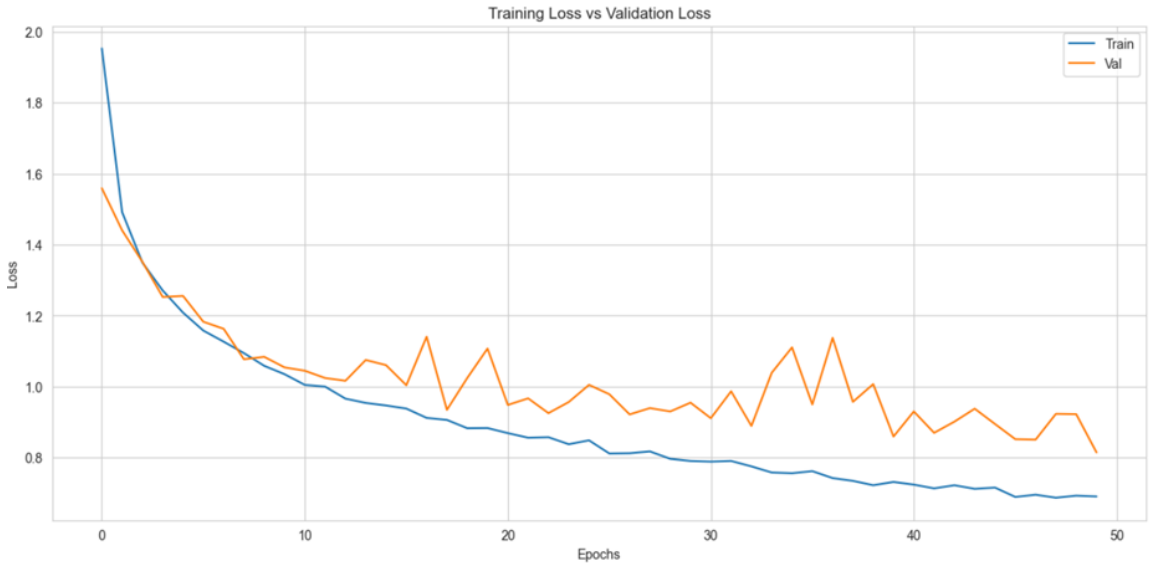
RESULT

RESNET50 ACCURACY CURVE



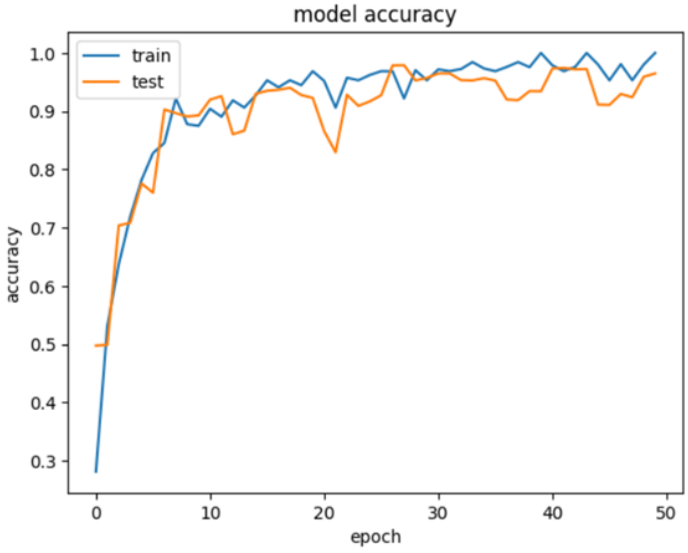
RESULT

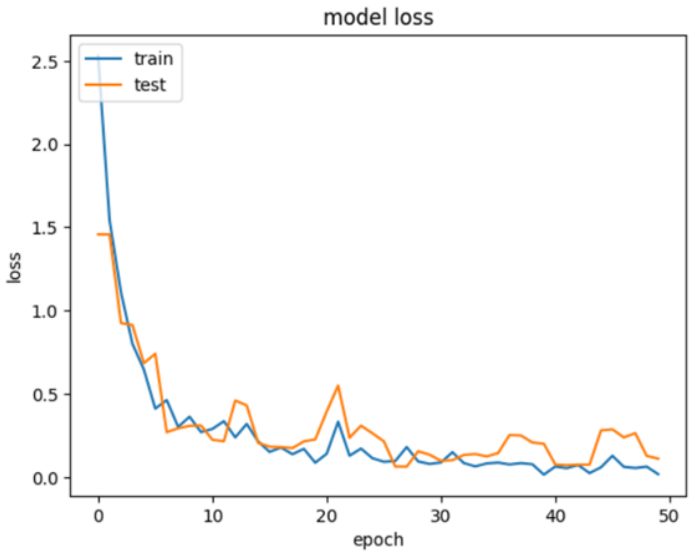
RESNET50 LOSS CURVE



RESULT

OUTPUT OF DENSENET121

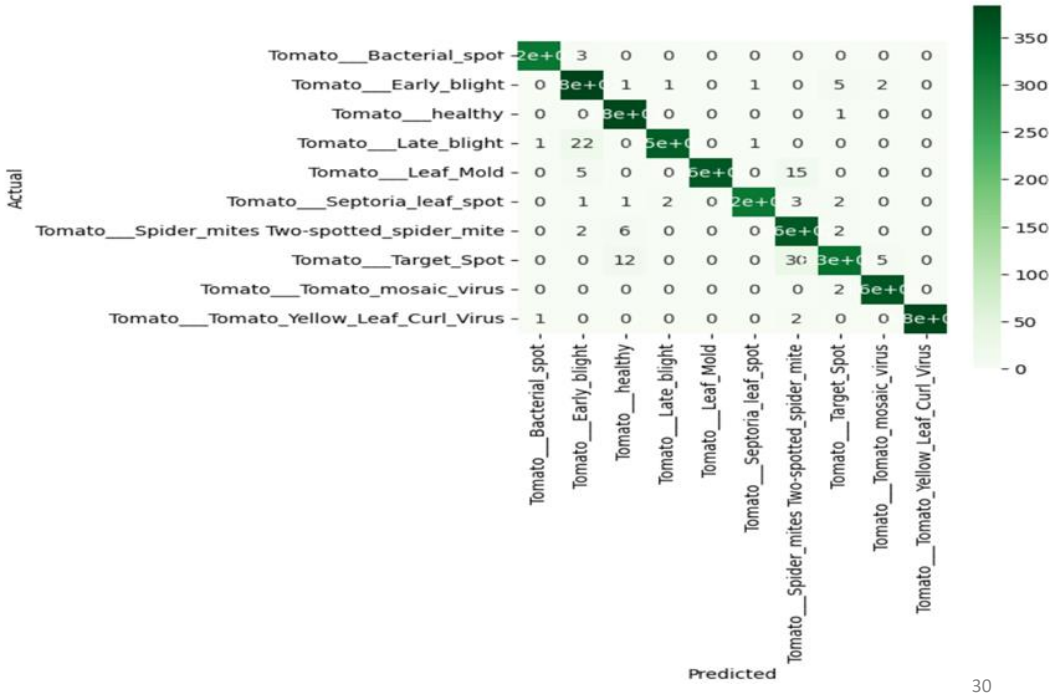




RESULT

Confusion matrix

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FN + FP)}$$



RESULT

F1 score and precision

F1 Score (weighted): 0.9599332150846099

Classification Report:

	precision	recall	f1-score	support
0	0.96	0.99	0.98	323
1	0.92	0.98	0.95	393
2	0.96	0.98	0.97	378
3	0.98	0.94	0.96	377
4	0.99	0.98	0.99	379
5	1.00	0.93	0.96	329
6	0.88	0.98	0.93	368
7	0.95	0.83	0.89	376
8	1.00	0.99	0.99	364
9	0.97	0.99	0.98	382
accuracy			0.96	3669
macro avg	0.96	0.96	0.96	3669
weighted avg	0.96	0.96	0.96	3669

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

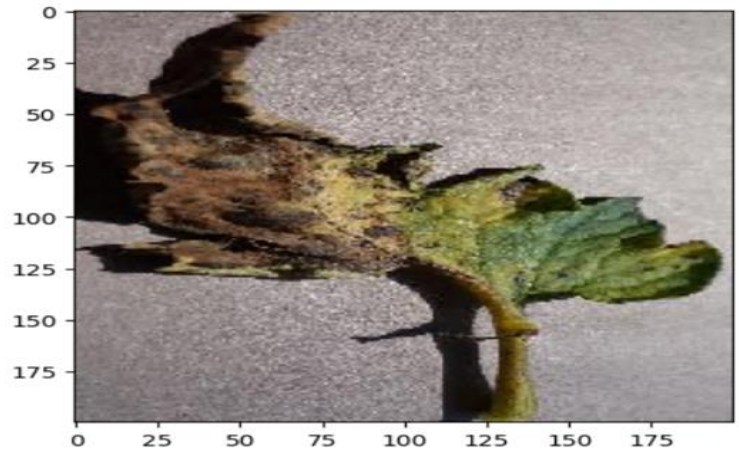
$$Precision = \frac{TP}{TP + FP}$$

$$Recall(R) = \frac{TP}{TP + FN}$$

RESULT

Predicted image(output)

```
Image loaded successfully.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
[2.9105140e-04 8.6968213e-01 6.2681079e-02 7.0700841e-03 4.1625514e-03
 2.1784984e-02 1.5737068e-04 1.2782962e-02 2.1355869e-02 3.1793857e-05]
```



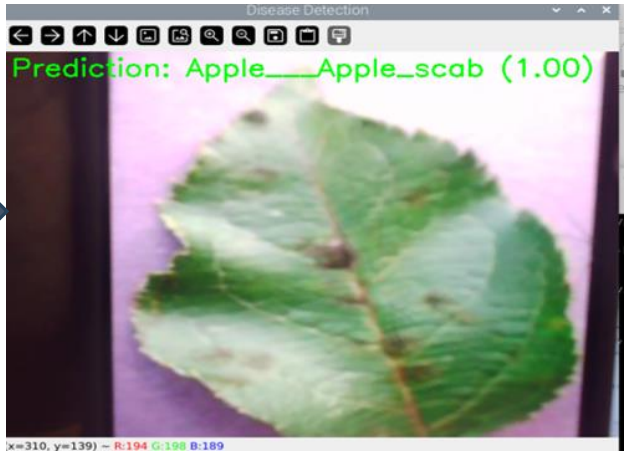

```
Prediction: Tomato___Early_blight
```

RESULT

The camera efficiently captures images of leaves, and the system accurately analyzes them using ResNet-50 to correctly identify disease present in the input image.

INPUT IMAGE

PREDICTED IMAGE



RESULT

Classes	Precision	Recall	F1-Score	Support
Apple___Apple_scab	0.86	0.99	0.92	504
Apple___Black_rot	0.97	0.98	0.98	497
Apple___Cedar_apple_rust	0.88	1.00	0.94	440
Apple___healthy	0.96	0.97	0.97	502
Blueberry___healthy	0.98	0.97	0.98	454
Cherry_(including_sour)___Powdery_mildew	0.99	0.99	0.99	421
Cherry_(including_sour)___healthy	0.99	0.96	0.98	456
Corn_(maize)_Cercospora_leaf_spot Gray_leaf_spot	0.95	0.89	0.92	410
Corn_(maize)Common_rust	1.00	1.00	1.00	477
Corn_(maize)_Northern_Leaf_Blight	0.87	0.98	0.92	477

Classes	Precision	Recall	F1-Score	Support
Corn_(maize)_healthy	0.98	1.00	0.99	465
Corn_(maize)_healthy	0.90	1.00	0.94	472
Grape__Esca(Black_Measles)	0.99	0.97	0.98	480
Grape__Leaf_blight(Isariopsis_Leaf_Spot)	1.00	0.91	0.95	430
Grape___healthy	0.97	0.98	0.97	423
Orange__Haunglongbing(Citrus_greening)	0.94	0.99	0.97	503
Peach___Bacterial_spot	0.96	0.97	0.97	459
Peach___healthy	0.99	0.99	0.99	432
pepper,bell__Bacterial_spot	0.96	0.95	0.95	478
Pepper,bell__healthy	0.95	0.97	0.96	497

RESULT

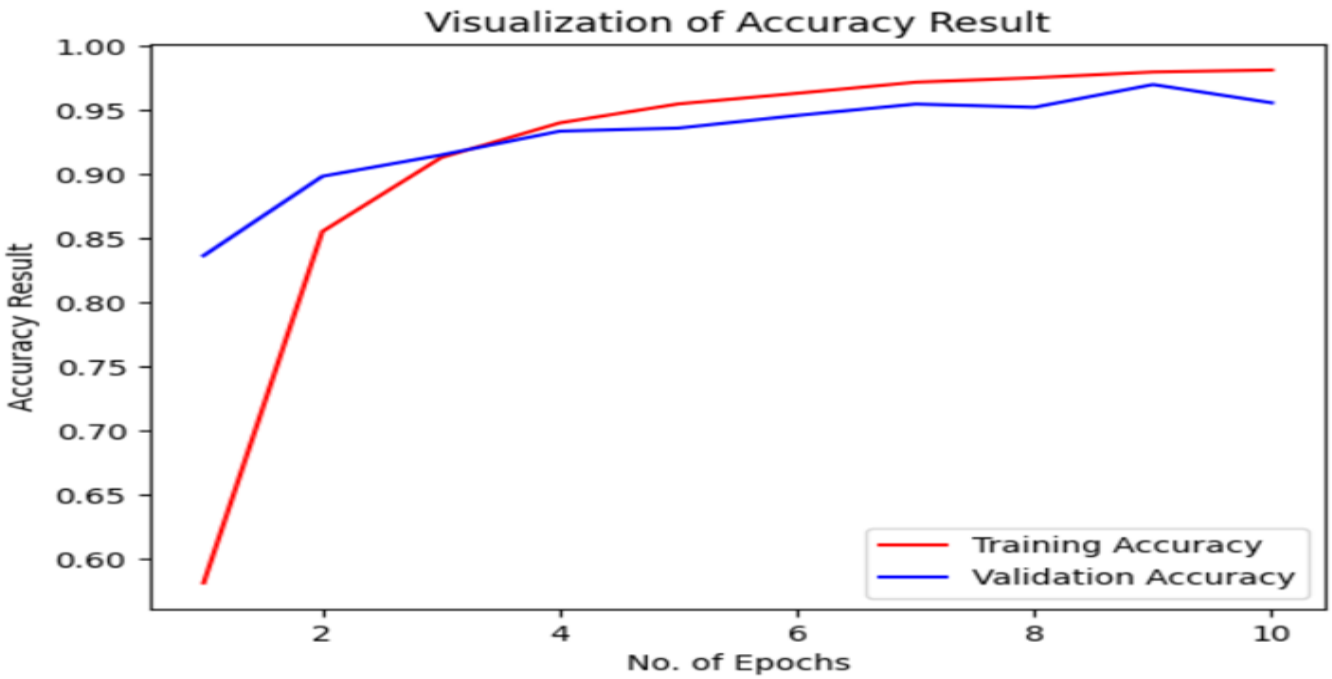
	Precision	Recall	F1-Score	Support
Potato___Early_blight	1.00	0.94	0.97	485
Potato___Late_blight	0.93	0.97	0.95	485
Potato___healthy	0.98	0.96	0.97	456
Accuracy			0.96	17572
Macro avg	0.96	0.95	0.96	17572
Weighted avg	0.96	0.96	0.96	17572

RESULT

Plant disease Prediction Confusion Matrix

0	**5e+02**	1	0	1	3	0	0	0	0	0	0	0
1	4	**4.9e+02**	3	0	0	0	0	0	0	2	0	0
2	0	0	**4.4e +02**	0	0	0	0	0	1	0	0	0
3	8	1	0	**4.9e+02**	0	0	2	0	0	0	0	0
4	1	0	3	4	**4.4e+02**	0	0	0	0	0	0	0
5	2	0	2	0	0	**4.2e+02**	0	0	0	0	0	0
6	0	0	0	1	0	1	**4.4e+02**	0	0	0	0	0
7	1	0	0	0	0	0	0	**3.6e+02**	1	43	0	0
8	0	0	0	0	0	0	0	1	**4.8e+02**	0	0	0
9	1	0	1	0	0	0	0	5	0	**4.7e+02**	1	0
10	0	0	0	0	0	0	0	0	0	0	**4.6e+02**	0
11	0	0	0	0	0	0	0	0	0	0	0	**4.7e+02**

RESULT



RESULT

WEATHER DETECTION:

Temperature (°C) Classification for Plant Growth

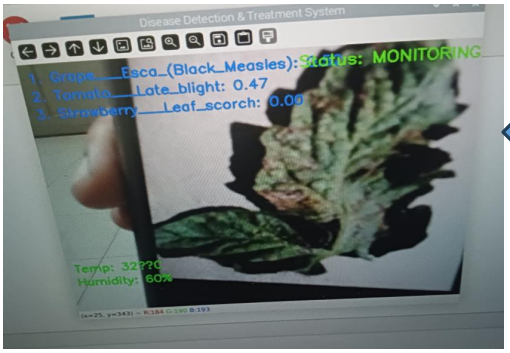
- **Normal (Optimal Growth):** 20–30°C
- **Good (Slightly Favorable):** 15–20°C or 30–35°C
- **Mild (Tolerable with Reduced Growth):** 10–15°C or 35–40°C
- **Poor (Stressful for Most Plants):** 5–10°C or 40–45°C
- **Worst (Severe Damage or Death Likely):** Below 5°C or above 45°C

Humidity (%) Classification for Plant Growth

- **Normal (Optimal Growth):** 50–70%
- **Good (Slightly Favorable):** 40–50% or 70–80%
- **Mild (Tolerable with Some Stress):** 30–40% or 80–90%
- **Poor (Stressful for Most Plants):** 20–30% or 90–95%
- **Worst (Severe Damage or Death Likely):** Below 20% or above 95%

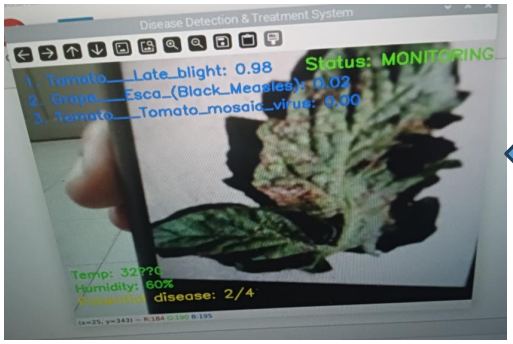
RESULT

1



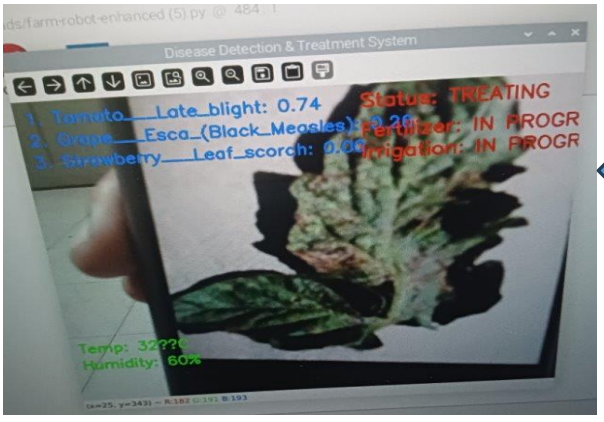
MONITORING THE LEAF FOR DISEASE

2



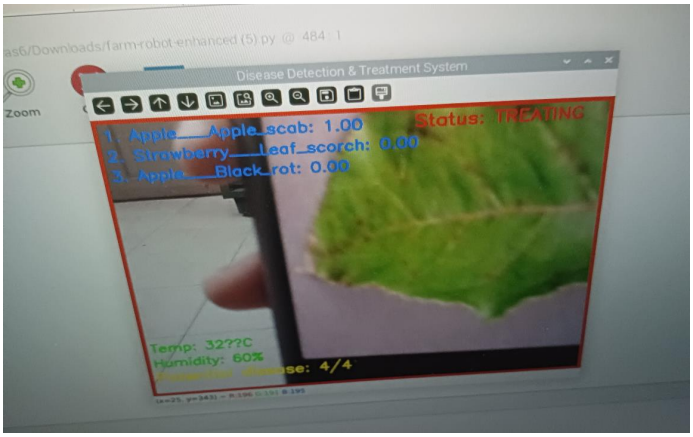
DETECTS DISEASE ON LEAF AND PREDICTS IT CORRECTLY.

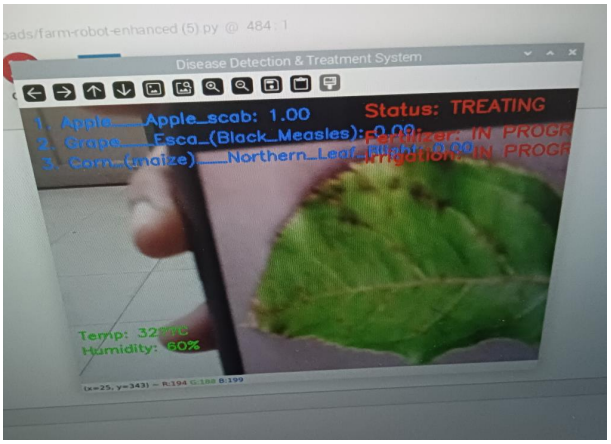
RESULT



AFTER THE DISEASE IS DETECTED, ACTIONS SUCH AS IRRIGATION AND FERTILIZER SPRAYING ARE INITIATED.TEMPERATURE AND HUMIDITY READINGS ARE ALSO DISPLAYED ON THE SCREEN.

RESULT





THE DISEASE IS PREDICTED CORRECTLY AFTER MONITORING THE LEAF.THE TEMPERATURE AND HUMIDITY READINGS ARE DISPLAYED ON THE SCREEN.

RESULT

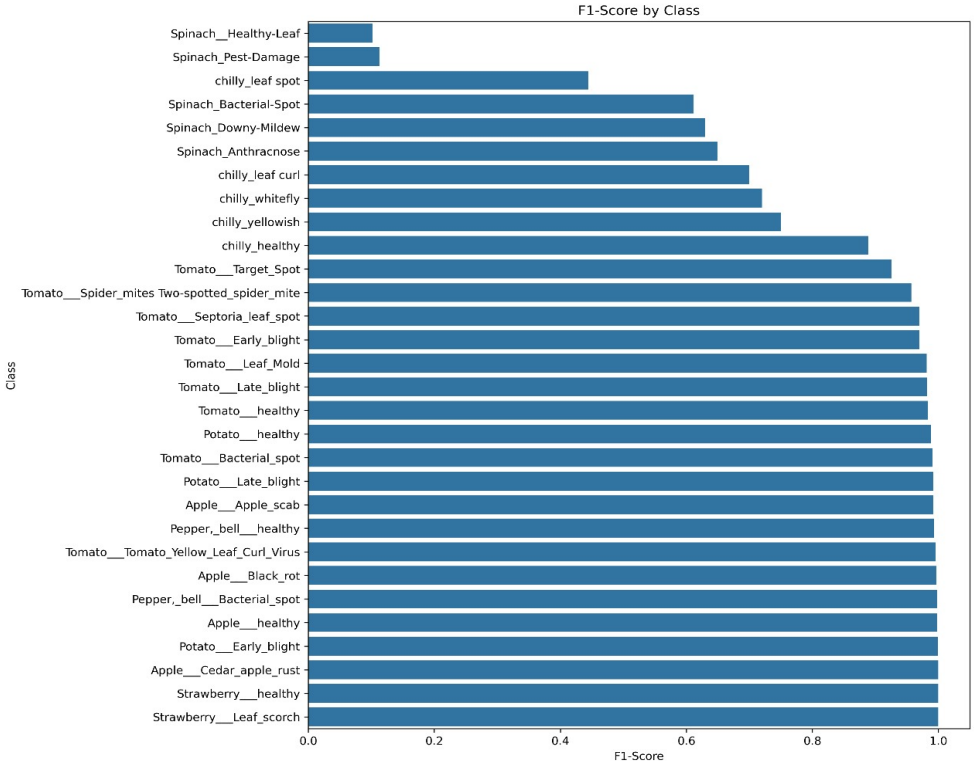
CLASSES	PRECISION	RECALL	F1-SCORE	SUPPORT
Apple___Apple_scab	0.992063	0.992063	0.992063	504
Apple___Black_rot	0.994	1	0.996991	497
Apple___Cedar_apple_rust	1	1	1	440
Apple___healthy	0.996032	1	0.998012	502
Pepper,_bell___Bacterial_spot	0.995833	1	0.997912	478
Pepper,_bell___healthy	0.986111	1	0.993007	497
Potato___Early_blight	0.997942	1	0.99897	485
Potato___Late_blight	1	0.983505	0.991684	485
Potato___healthy	0.997763	0.97807	0.987818	456
Spinach_Anthracnose	0.65	0.65	0.65	20
Spinach_Bacterial-Spot	0.694915	0.546667	0.61194	150
Spinach_Downy-Mildew	0.506329	0.833333	0.629921	48
Spinach_Pest-Damage	0.077778	0.205882	0.112903	102
Spinach__Healthy-Leaf	0.178571	0.071685	0.102302	279
Strawberry___Leaf_scorch	1	1	1	444
Strawberry___healthy	1	1	1	456

CLASSES	PRECISION	RECALL	F1-SCORE	SUPPORT
Tomato___Bacterial_spot	0.986014	0.995294	0.990632	425
Tomato___Early_blight	0.989177	0.952083	0.970276	480
Tomato___Late_blight	0.978541	0.984881	0.981701	463
Tomato___Leaf_Mold	0.997802	0.965957	0.981622	470
Tomato___Septoria_leaf_spot	0.992788	0.947248	0.969484	436
Tomato___Spider_mites Two-spotted_spider_mite	0.992593	0.924138	0.957143	435
Tomato___Target_Spot	0.865019	0.995624	0.925738	457
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1	0.991837	0.995902	490
Tomato___healthy	0.967807	1	0.98364	481
chilly_healthy	1	0.8	0.888889	10
chilly_leaf curl	0.7	0.7	0.7	10
chilly_leaf spot	0.5	0.4	0.444444	10
chilly_whitefly	0.6	0.9	0.72	10
chilly_yellowish	1	0.6	0.75	10

RESULT

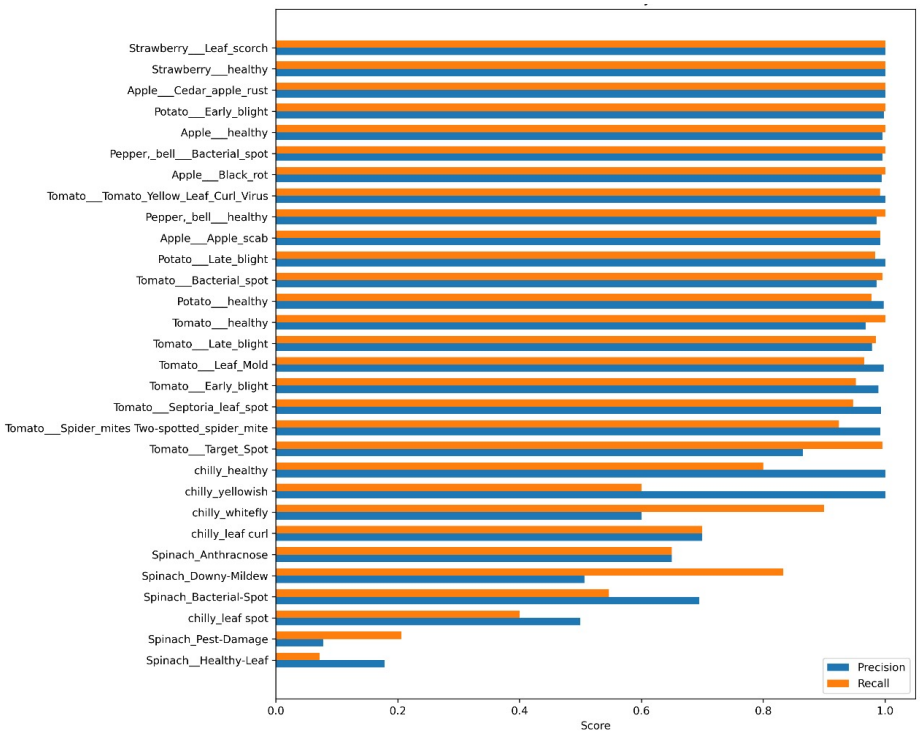
accuracy	0.943071	0.943071	0.943071	0.943071
macro avg	0.854569	0.847276	0.8441	10030
weighted avg	0.946437	0.943071	0.943065	10030

RESULT



The chart clearly visualizes model performance across all classes using F1-scores. Most plant disease classes achieve high F1-scores, indicating strong overall classification accuracy and balanced precision-recall performance.

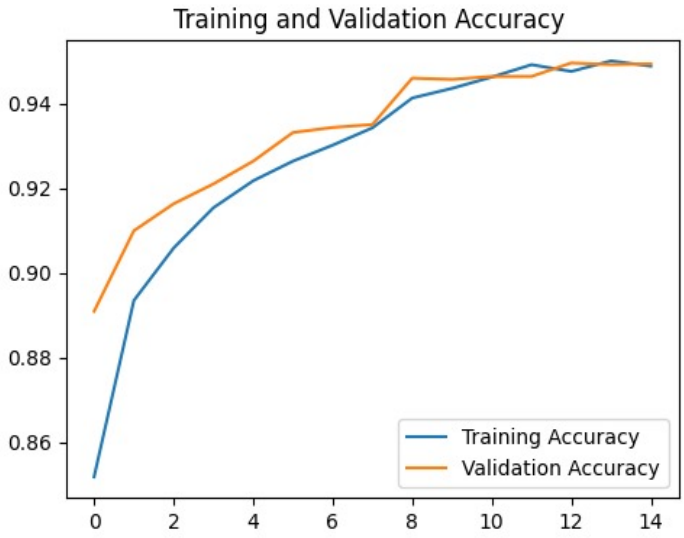
RESULT

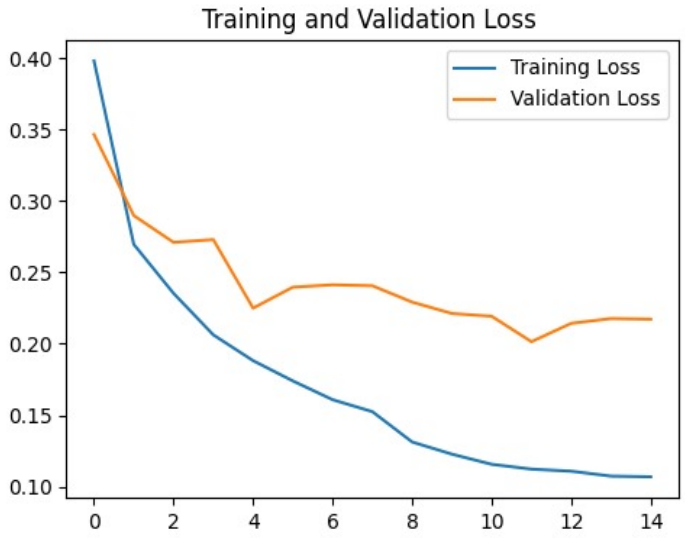


This figure effectively compares precision and recall for each class, showcasing detailed insights. Many classes exhibit near-perfect scores, reflecting the model's strong ability to a detect and identify plant diseases.

RESULT

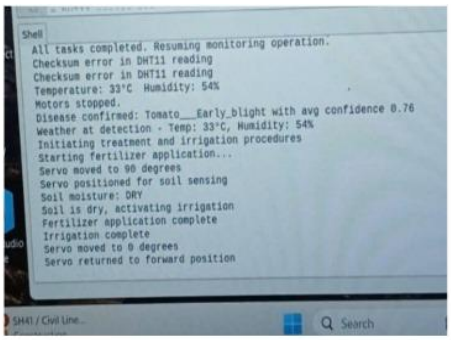
RESNET50 MODEL ACCURACY AND LOSS FOR 30 CLASSES





RESULT

(a) Dry Soil



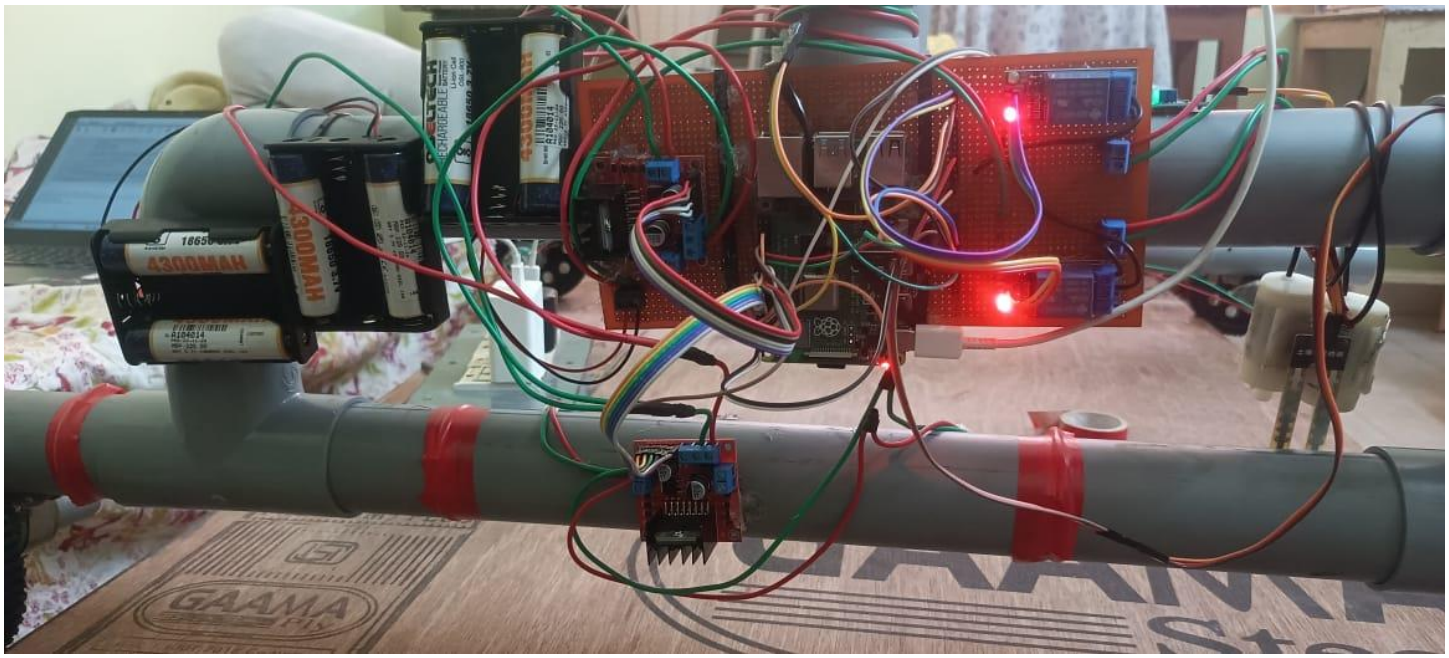
(b)Soil Moisture detecting the soil as dry

The soil moisture level was detected, and based on whether it was dry or wet, the irrigation system was automatically activated.

RESULT

STRUCTURE OF THE FARMBOT

CONNECTONS TO RASPERRY PI



WORK DIVISION

ALKA DENNY	• Dataset training and preprocessing. • Splitting data and implementing ResNet50 model.
ANITHRA ROSS AJITH	• Setting up Raspberry Pi 4 and camera module. • Connecting and configuring FarmBot sensors.
ANNA JOJU	• Transfer learning and model optimization. • Deploying the model on Raspberry Pi 4.
ANU XAVIER	• Building the FarmBot structure and mounting components. • Implementing rover movement and testing.

CONCLUSION

- The smart farming system successfully automates irrigation and disease management.
- Soil moisture sensors effectively trigger water sprinkling, ensuring optimal hydration.
- The camera-based disease detection system enables timely fertilizer spraying, reducing manual intervention.
- The project demonstrates a fully functional model integrating sensors, motors, and Raspberry Pi for precision farming.
- However, challenges such as hardware compatibility issues, optimizing deep learning models for real-time predictions, and ensuring stable wireless communication were encountered and addressed during development.

FUTURE SCOPE

- **Moving Camera Integration:** Instead of a stationary camera, a moving camera module can be used to scan a larger area for disease detection, improving accuracy and coverage.
- **Weed Removal Mechanism:** The system can be enhanced to detect and remove weeds using robotic arms or mechanical tools.
- **Pesticide Prevention:** AI-based analysis can identify pest infestations and trigger automated pesticide spraying only when necessary, reducing excessive chemical use.
- **Solar Panel Integration:** Using solar energy can make the system more sustainable and reduce dependence on external power sources.
- **IoT-Based Remote Monitoring:** Implementing IoT technology can allow farmers to monitor soil moisture, crop health, and system performance in real time through a mobile application.

REFERENCES

- Senthil Kumar Swami Durai, Mary Divya Shamili, Smart farming using Machine Learning and Deep Learning techniques, Decision Analytics Journal, Volume 3, 2022,100041,ISSN 2772-6622, https://doi.org/10.1016/j.dajour.2022.100041.
- U.Archana, Amanulla Khan, Appani Sudarshanam, C.Sathya, Ashok Kumar Koshariya, R.Krishnamoorthy . "Plant Disease Detection using ResNet." 2023 International Conference on Inventive Computation Technologies (ICICT). DOI : 10.1109/ICICT57646.2023.10133938
- Too, Edna Chebet, et al. "A comparative study of fine-tuning deep learning models for plant disease identification." *Computers and Electronics in Agriculture* 161 (2019): 272-279.
- Swaminathan, Akshay, C. Varun, and S. Kalaivani. "Multiple plant leaf disease classification using densenet-121 architecture." *Int. J. Electr. Eng. Technol* 12 (2021): 38-57.
- Mukti, Ishrat Zahan, and Dipayan Biswas. "Transfer learning based plant diseases detection using ResNet50." *2019 4th International conference on electrical information and communication technology (EICT)*. IEEE, 2019.
- Eunice, Jennifer, et al. "Deep learning-based leaf disease detection in crops using images for agricultural applications." *Agronomy* 12.10 (2022): 2395.

REFERENCES

- S. M. NURUZZAMAN NOBEL , MD. ASIF IMRAN, NAHIDA ZAMAN BINA, MD. MOHSIN KABIR, MEJDL SAFRAN, SULTAN ALFARHOOD, AND M. F. MRIDHA . "Palm Leaf Health Management: A Hybrid Approach for Automated Disease Detection and Therapy Enhancement." IEEE ACCESS VOLUME 12, 2024.Digital Object Identifier 10.1109/ACCESS.2024.3351912.
- Jayme Garcia Arnal Barbedo. "Plant disease identification from individual lesions and spots using deep learning" . biosystems engineering 180 (2019) 96 – 107.
- Jayme Garcia Arnal Barbedo . " A review on the main challenges in automatic plant disease identification based on visible range images ". biosystems engineering 144 (2016) 52 – 60.
- RUCHI RANI , JAYAKRUSHNA SAHOO, SIVAIAH BELLAMKONDA , SUMIT KUMAR , AND SANJEEV KUMAR PIPPAL."Role of Artificial Intelligence in Agriculture: An Analysis and Advancements With Focus on Plant Diseases". IEEE ACCESS VOLUME 11, 2023. Digital Object Identifier 10.1109/ACCESS.2023.3339375.
- Youness Tace, Mohamed Tabaa, Sanaa Elfilali, Cherkaoui Leghris, Hassna Bensag, Eric Renault."Smart irrigation system based on IoT and machine learning".TMREES22-Fr, EURACA, 09 to 11 May 2022, Metz-Grand Est, France. Energy Reports 8 (2022) 1025–1036.

THANK YOU

Appendix F: Questions and Answers

Questions and Answers

Q1: What is the primary objective of the project described in the report?

The primary objective of the project is to develop a smart farming system that enhances plant leaf disease detection using deep learning models like ResNet50 and DenseNet121. It aims to automate irrigation and fertilizer application based on disease detection to optimize resource use and crop health. The system is designed for real-time deployment using Raspberry Pi and edge computing to enable efficient, on-field decision-making.

Q2:What are the benefits of using this system in farming?

The smart farming system offers several benefits, including early and accurate detection of plant diseases, which helps prevent crop loss and improves yield. It automates irrigation and fertilization,saving water and reducing the overuse of chemicals. Overall,it enhances farming efficiency, lowers labor costs, and promotes sustainable agriculture through smart resource management.

Q3: What methodology is used to implement the project?

The project uses deep learning methodologies, primarily training ResNet50 and DenseNet 121 models for accurate tomato leaf disease detection. Data augmentation techniques are applied to enhance model generalization, and the trained model is optimized and converted to TensorFlow Lite for deployment on edge devices. The system is implemented on a Raspberry Pi 4 with a camera module for real-time image detection and automated decision-making in smart farming.

Q4: What are the key findings discussed in the results and discussion section?

The ResNet-50 model achieved a high accuracy of 95% in detecting leaf diseases from images, proving to be more efficient and faster than DenseNet121 for this application.Integration with Raspberry Pi and sensors like DHT11 and LM393 enabled real-time monitoring of environmental conditions and automated responses such as irrigation and fertilizer spraying based on detected plant health.

Q5:Why did we choose ResNet50 over DenseNet 121 in this project?

ResNet50 was chosen because it performed better in detecting plant leaf diseases, giving a higher accuracy of 96.8 percentage compared to DenseNet. It also processed images faster, which is important for real-time detection when using devices like Raspberry Pi that have limited computing power. Additionally, ResNet50 was easier to optimize and convert to a lighter format (TensorFlow Lite), making it more efficient for deployment in the smart farming system where quick and accurate responses are needed.

Anithra_Project_Report_Phase_2.pdf

ORIGINALITY REPORT

19%	13%	14%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to APJ Abdul Kalam Technological University, Thiruvananthapuram Student Paper	4%
2	huggingface.co Internet Source	1%
3	www.mdpi.com Internet Source	1%
4	"Computing Technologies for Sustainable Development", Springer Science and Business Media LLC, 2025 Publication	<1%
5	github.com Internet Source	<1%
6	Submitted to apjabdul Student Paper	<1%
7	M. A. Jabbar, Kantipudi MVV Prasad, Sheng-Lung Peng, Mamun Bin Ibne Reaz, Ana Madureira. "Machine Learning Methods for Signal, Image and Speech Processing", Routledge, 2022 Publication	<1%
8	www.alibabacloud.com Internet Source	<1%
9	www.coursehero.com Internet Source	<1%
	ijircce.com	