VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI-590018

21CSMP67

MINI PROJECT REPORT

on

Agro-Infection Detection

Submitted in partial fulfillment of the requirements for 6th Semester in
Bachelor of Engineering in Computer Science and Engineering
of Visvesvaraya Technological University, Belagavi

Submitted by

Riya 1RN21CS124

Shilpa 1RN21CS144

Under the Guidance of:

Dr. Sudhamani M J

Associate Professor

Dept. of CSE, RNSIT

ESTD : 2001
An Institute with a Difference

Department of Computer Science and Engineering

RNS Institute of Technology

(Accredited by NBA upto 30-06-2025)

Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560098

2023-2024

RNS INSTITUTE OF TECHNOLOGY

Channasandra, Dr.Vishnuvardhan Road, Bengaluru-560098

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(Accredited by NBA upto 30-06-2025)

CERTIFICATE

Certified that the Mini Project work has been successfully carried out by **Riya** bearing USN **1RN21CS124** and **Shilpa** bearing USN **1RN21CS144** bonafide student of **RNS Institute of Technology** in partial fulfillment of the requirements of 6th Semester in **Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belagavi** during the academic year **2023-2024**. The mini project report has been approved as it satisfies the academic requirements in respect of Mini Project work for the said degree.

Dr. Sudhamani M J	**Dr. Kiran P**	**Dr. Ramesh Babu H S**
Associate Professor	Professor and HOD	Principal
Dept. of CSE	Dept. of CSE	RNSIT
RNSIT	RNSIT	

Abstract

Plant diseases represent a critical challenge to global agriculture, leading to reduced crop yield and quality, which subsequently hampers economic growth and threatens food security. With the world's population continuing to expand, there is an increasing urgency to develop advanced early detection methods for plant diseases. This paper delves into various Machine Learning (ML) and Deep Learning (DL) techniques aimed at identifying plant diseases, with a particular focus on image processing methodologies, especially Convolutional Neural Networks (CNNs).

The research underscores the efficacy of these techniques in providing accurate and efficient disease identification. Major advancements in the field are examined, highlighting the use of pre-trained models such as DenseNet-121 and ResNet-50, which have demonstrated high classification accuracy on multiple datasets. These models leverage extensive training on large-scale image data, enabling them to recognize and classify plant diseases with remarkable precision.

The findings from this study illustrate how Artificial Intelligence (AI) and ML can revolutionize the agricultural sector by enabling real-time, accurate detection of plant diseases. This capability is crucial for implementing timely interventions, which can mitigate crop losses and enhance overall productivity. Moreover, the integration of these advanced technologies supports sustainable farming practices by reducing the reliance on chemical treatments and promoting more targeted and efficient use of resources.

The application of ML and DL in plant disease detection offers significant benefits, including improved crop health monitoring, enhanced food security, and economic growth. By leveraging the power of AI, the agricultural industry can move towards more sustainable and resilient practices, ensuring that the growing global population is adequately fed.

Acknowledgement

At the very onset, I would like to place on record my gratitude to all those people who have helped me in making this Mini Project work a reality. Our Institution has played a paramount role in guiding in the right direction.

I would like to profoundly thank **Sri. Satish R Shetty**, Managing Director, RNS Group of Companies, Bengaluru for providing such a healthy environment for the successful completion of this Mini Project work.

I would like to thank our beloved Director, **Dr. M K Venkatesha**, for his constant encouragement which motivated me to complete this work.

I would also like to thank our beloved Principal, **Dr. Ramesh Babu H S** , for providing the necessary facilities to carry out this work.

I am extremely grateful to **Dr. Kiran P**, Professor and Head, Department of CSE for his constant encouragement and motivation which helped me to accomplish this work.

I would like to express my sincere thanks to our Mini Project Coordinator, **Mrs. Mamatha Jajur S**, Assistant Professor, Department of CSE, RNSIT.

My heartfelt thanks to my guide **Dr. Sudhamani M J**, Associate Professor, Department of CSE, for her continuous guidance and constructive suggestions for this work.

I am thankful to all the teaching and non-teaching staff members of the Computer Science and Engineering Department for their encouragement and support throughout this work.

Riya 1RN21CS124

Shilpa 1RN21CS144

EVALUATION SHEET

SL. NO	PARTICULARS	MAX MARKS	MARKS AWARDED
1	**PHASE 1** Title Scrutiny	10	
2	**PHASE 2** Implementation (50% Completion)	20	
3	**PHASE 3** Complete Implementation	50	
4	**REPORT SUBMISSION**	20	
	TOTAL	100	

Signature of Guide

Contents

Chapter 1

Introduction

1.1 Background

Early detection of plant diseases is essential for maintaining crop yields and mitigating economic losses, especially as global food demand is expected to rise by over 50% by 2050. The integration of AI and IoT technologies is transforming plant disease management, offering advanced solutions that surpass traditional methods. Convolutional Neural Networks (CNNs) and architectures like ResNet enhance disease classification accuracy and are increasingly accessible through smartphones, facilitating early detection. Transfer learning, leveraging pre-trained models from large datasets like ImageNet, improves performance with smaller datasets, while advanced imaging techniques such as chlorophyll fluorescence and infrared thermography hold promise for early disease detection. However, their high costs currently limit their adoption, particularly in remote farming regions.

Project is focusing on combining image processing, machine learning to optimize disease detection and management. Despite significant advancements, challenges such as the high cost of technology and limited horticultural expertise in some areas persist. Efforts are ongoing to make these technologies more accessible and effective. Funding from agricultural innovation agencies could be pivotal in driving progress, ensuring that these technologies benefit farmers globally and contribute to enhanced food security.

1.2 Problem Statement

Plant diseases pose significant challenges to agriculture, impacting crop yield and quality, and consequently affecting economic growth and food security. As the global population continues to rise, the need for advanced and efficient early detection methods for plant diseases has become increasingly urgent. Traditional visual methods for disease identification are often costly, labor-intensive, and ineffective, particularly in large-scale farming operations or remote regions with limited resources.

The problem addressed by this project is the need for a reliable, scalable, and accessible solution for detecting plant diseases using modern technology. Conventional methods fall short in terms of accuracy and speed, and there is a pressing requirement for a solution that can provide real-time disease diagnosis to support timely interventions and decision-making in agriculture.

1.3 Objectives

The primary objectives of this project are:

1. **Develop a Web-Based Application:** Create a web-based tool that leverages deep learning techniques for real-time plant disease detection and classification.

2. **Utilize Convolutional Neural Networks (CNNs):** Implement a pre-trained CNN to accurately classify plant images into various disease categories, improving diagnostic precision and speed.

3. **Enhance Accessibility:** Design the application with a user-friendly interface to ensure it is accessible to farmers and agronomists with minimal technical expertise.

4. **Support Sustainable Farming:** Provide a tool that not only improves disease management but also promotes sustainable farming practices by reducing reliance on chemical treatments and minimizing economic losses from crop diseases.

5. **Showcase AI Integration:** Demonstrate the effective application of artificial intelligence in solving practical agricultural challenges and paving the way for further research and development in agricultural technology.

1.4 Scope

The scope of this project includes:

1. **Dataset and Training:** Utilizing a dataset of approximately 87,000 RGB images of plant leaves categorized into 38 distinct classes. The model will be trained using this dataset to classify images accurately.

2. **Model Implementation:** Employing Convolutional Neural Networks (CNNs) and techniques such as transfer learning to enhance model performance and efficiency in plant disease detection.

3. **Web Application Development:** Integrating the trained model into a web-based application using frameworks like Flask, allowing users to upload images and receive real-time disease predictions.

4. **Evaluation and Testing:** Assessing the model's performance through metrics such as accuracy, precision, recall, and F1-score. Testing the model with real-world scenarios to validate its effectiveness in practical agricultural applications.

5. **User Interface Design:** Creating a user-friendly interface that simplifies interaction for users and provides clear, actionable diagnostic feedback.

6. **Deployment:** Enabling deployment of the application on edge devices or cloud platforms to ensure broad accessibility and practical use in diverse agricultural settings.

Chapter 2

Methodology

2.1 Software Requirements

The software requirements are description of features and functionalities of the system.Table 2.1 gives details of software requirements.

Software Component	**Description**
Operating System	Windows
Programming Language	Python 3.x
Frameworks	Flask (for web application development)
Deep Learning Libraries	TensorFlow, Keras
Image Processing Libraries	OpenCV
Development Tools	Integrated Development Environment (IDE) like Jupyter Notebook, VS Code
Browser	Modern web browsers (e.g., Chrome, Firefox)

Table 2.1: Software Requirements for Agro-Infection Detection Application

2.2 Algorithms

The algorithm for training a Convolutional Neural Network (CNN) for plant disease detection is as follows:

1. **Initialize the CNN:** Start by initializing the CNN with random weights and setting key hyperparameters including the learning rate, batch size, and number of epochs. This step prepares the network for training.

2. **Load and Preprocess Data:** Load the training and validation datasets, normalize the image data to ensure consistent input, and split the data into training and validation sets. Proper preprocessing is crucial for effective training and evaluation.

3. **Build the CNN Architecture:** Construct the CNN by adding an input layer with dimensions matching the images, followed by convolutional layers with specified filters, kernel sizes, and activation functions like ReLU. Include pooling layers to reduce dimensionality, dropout layers to mitigate overfitting, fully connected (dense) layers, and an output layer with softmax activation for classification.

4. **Compile the CNN:** Compile the model by defining the loss function (e.g., categorical cross-entropy), choosing an optimizer (e.g., Adam), and specifying metrics to monitor performance, such as accuracy. This prepares the model for training.

5. **Train the CNN:** Train the CNN using the training data, validating the model with the validation data at the end of each epoch. Save the best-performing model based on validation metrics to ensure the best possible results.

6. **Evaluate the CNN:** After training, evaluate the saved model on test data to calculate performance metrics including accuracy, precision, recall, and F1 score. This step assesses how well the model generalizes to new data.

7. **Make Predictions:** Preprocess new images for prediction, use the trained CNN to classify these images, and output the predicted class along with a confidence score. This final step enables practical application of the trained model for plant disease detection.

2.3 Data Collection and Analysis Procedure

2.3.1 Data Collection

The success of any machine learning model, particularly in the context of plant disease detection, hinges significantly on the quality and scope of the dataset used. In this study, the dataset for training and testing the Convolutional Neural Network (CNN) was meticulously collected and prepared as follows:

- **Source of Data:** The dataset consists of approximately 87,000 RGB images of plant leaves, sourced from multiple agricultural and botanical databases. These images include both healthy and diseased plant leaves, categorized into 38 distinct classes, representing a diverse range of plant species and diseases.

- **Data Annotation:** Each image in the dataset was manually annotated to label the presence and type of disease, or to indicate a healthy condition. This process involved subject-matter experts to ensure accuracy in disease classification.

- **Data Splitting:** The dataset was divided into training, validation, and testing subsets. Specifically, 80

- **Image Preprocessing:** Images were preprocessed to ensure consistency and suitability for model training. This included resizing images to 128x128 pixels, normalizing pixel values, and converting color formats when necessary (e.g., from BGR to RGB) to match model input requirements.

2.3.2 Data Analysis

Training and Validation: The collected dataset was used to train the CNN model, employing techniques to optimize learning and prevent overfitting:

- **Training Strategy:** The training process involved iterative epochs where the CNN learned to identify patterns in plant images. The dataset was used to adjust model weights and biases through backpropagation and gradient descent techniques.

- **Validation:** During training, a separate validation set was used to evaluate the model's performance on data it had not seen before. This allowed for monitoring and adjusting hyperparameters to avoid overfitting and ensure robust performance.

- **Data Augmentation:** To enhance the model's ability to generalize, data augmentation techniques such as rotation, flipping, and scaling were applied to the training images, effectively increasing the diversity of the dataset.

Evaluation Metrics: The performance of the trained model was assessed using several key metrics:

- **Accuracy:** Measures the proportion of correctly identified instances among the total number of instances.
- **Precision:** Indicates the ratio of true positive predictions to the total number of positive predictions made by the model.
- **Recall:** Represents the ratio of true positive predictions to the total number of actual positive instances in the dataset.
- **F1 Score:** Provides a balanced measure combining precision and recall, especially useful in scenarios with imbalanced classes.

The results from these metrics were used to evaluate the model's effectiveness in classifying plant diseases and to guide further refinement and optimization of the model.

2.3.3 Testing and Deployment

- **Testing Phase:** The model was tested using a set of new, unseen images to evaluate its real-world performance. This phase involved loading the pre-trained model, processing the test images, and generating predictions to assess accuracy and reliability.
- **Deployment:** Following successful testing, the model was integrated into a web-based application. This application allows users to upload plant images for real-time disease prediction, providing immediate feedback and facilitating practical use in agricultural settings.

Through this systematic approach to data collection and analysis, the study aimed to develop a robust and effective tool for plant disease detection, leveraging deep learning techniques to support agricultural practices and enhance crop management.

2.4 Code

2.4.1 Model Training and Evaluation

Listing 2.1: Model Training and Evaluation

```
import tensorflow as tf
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

# Loading training set from the directory 'train'
training_set = tf.keras.utils.image_dataset_from_directory(
    'train',
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    interpolation="bilinear"
)

# Loading validation set from the directory 'valid'
validation_set = tf.keras.utils.image_dataset_from_directory(
    'valid',
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=True,
    interpolation="bilinear"
)

# Importing necessary layers from Keras
```

```
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten,
    Dropout
from tensorflow.keras.models import Sequential

# Creating the model
model = Sequential()

# Adding layers to the model
model.add(Conv2D(filters=32, kernel_size=3, padding='same', activation='relu',
    input_shape=[128, 128, 3]))
model.add(Conv2D(filters=32, kernel_size=3, activation='relu'))  # No padding to
     boost training speed
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(filters=64, kernel_size=3, padding='same', activation='relu'))
model.add(Conv2D(filters=64, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(filters=128, kernel_size=3, padding='same', activation='relu'))
model.add(Conv2D(filters=128, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(filters=256, kernel_size=3, padding='same', activation='relu'))
model.add(Conv2D(filters=256, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Conv2D(filters=512, kernel_size=3, padding='same', activation='relu'))
model.add(Conv2D(filters=512, kernel_size=3, activation='relu'))
model.add(MaxPooling2D(pool_size=2, strides=2))
model.add(Dropout(0.25))  # To avoid overfitting

# Flattening the layers
model.add(Flatten())

# Adding dense layers
model.add(Dense(units=1500, activation='relu'))
model.add(Dropout(0.45))  # To avoid overfitting
model.add(Dense(units=38, activation='softmax'))  # Output layer for 38 classes
    with softmax activation
```

```
# Compiling the model
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001), loss='
    categorical_crossentropy', metrics=['accuracy'])

# Displaying the model summary
model.summary()  # To see all the parameters of the model

# Training the model
training_h = model.fit(x=training_set, validation_data=validation_set, epochs
    =10)

# Evaluating the model on the training set
train_loss, train_acc = model.evaluate(training_set)

# Evaluating the model on the validation set
val_loss, val_acc = model.evaluate(validation_set)

print(val_loss, val_acc)
print(train_loss, train_acc)

# Saving the trained model
model.save("trained_model.keras")

# Recording the training history
import json
with open("training_history.json", "w") as f:
    json.dump(training_h.history, f)

# Plotting the training and validation accuracy
epochs = [i for i in range(1, 11)]
plt.plot(epochs, training_h.history['accuracy'], color='green', label='Training
    Accuracy')
plt.plot(epochs, training_h.history['val_accuracy'], color='blue', label='
    Validation Accuracy')
plt.xlabel('Number of Epoch')
plt.ylabel('Accuracy')
```

```
plt.title('Accuracy Visualization')
plt.legend()
plt.show()

# Getting the class names from the validation set
class_name = validation_set.class_names

# Loading test set from the directory 'valid'
test_set = tf.keras.utils.image_dataset_from_directory(
    'valid',
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=32,
    image_size=(128, 128),
    shuffle=False,
    interpolation="bilinear"
)

# Making predictions on the test set
y_pred = model.predict(test_set)

# Getting the predicted categories
predicted_categories = tf.argmax(y_pred, axis=1)

# Getting the true categories from the test set
true_categories = tf.concat([y for x, y in test_set], axis=0)
y_true = tf.argmax(true_categories, axis=1)

# Importing classification report and confusion matrix from sklearn
from sklearn.metrics import classification_report, confusion_matrix

# Printing the classification report
print(classification_report(y_true, predicted_categories, target_names=
    class_name))
```

2.4.2 Flask Application

Listing 2.2: Flask Application Setup

```
from flask import Flask, request, render_template, url_for
import numpy as np
import tensorflow as tf
import cv2
import os
from werkzeug.utils import secure_filename

# Load the trained Keras model from the specified path
model = tf.keras.models.load_model('model/trained_model.keras')



# List of class names corresponding to the model's output classes
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'
]
```

2.4.3 Flask Application Routes

Listing 2.3: Flask Application Setup

```
# Create Flask app
app = Flask(__name__, static_url_path='/static')

# Configuration for file uploads
app.config['UPLOAD_FOLDER'] = 'static/uploads'  # Directory where uploaded files
    will be stored
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024  # Max file size allowed (16
    MB)
```

Listing 2.4: Home Route

```
# Route for rendering the index page
@app.route("/")
def home():
    return render_template("index.html")  # Render the HTML template for the
        home page
```

Listing 2.5: Prediction Route

```
# Route to handle image upload and prediction
@app.route("/predict", methods=["POST"])
def predict():
    if request.method == "POST":
        # Check if the request contains a file part
        if 'file' not in request.files:
            return "No file part"

        file = request.files['file']
```

```
# Check if the file is selected
if file.filename == '':
    return "No selected file"

if file:
    # Save the uploaded file
    filename = secure_filename(file.filename)  # Secure the filename
    filepath = os.path.join(app.config['UPLOAD_FOLDER'], filename)  #
       Define the path to save the file
    file.save(filepath)  # Save the file to the upload folder

    # Read and preprocess the image
    img = cv2.imread(filepath)  # Read the image using OpenCV
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)  # Convert the image from
        BGR to RGB

    # Resize and preprocess image for prediction
    img = cv2.resize(img, (128, 128))  # Resize image to match the model
       's input size
    img_array = tf.keras.preprocessing.image.img_to_array(img)  #
       Convert the image to a numpy array
    img_array = np.expand_dims(img_array, axis=0)  # Expand dimensions
       to match the model's expected input shape

    # Perform prediction
    predictions = model.predict(img_array)  # Get model predictions
    predicted_class_idx = np.argmax(predictions)  # Get the index of the
        highest probability
    predicted_class = class_names[predicted_class_idx]  # Map index to
       class name

    # Render result in HTML
    return render_template("index.html", prediction=predicted_class,
       image_url=url_for('static', filename='uploads/' + filename))
```

```
    return "Prediction failed"
```

Listing 2.6: Running the Flask Application

```
if __name__ == '__main__':
    app.run(debug=True)  # Run the Flask application in debug mode
```

Chapter 3

Results

3.1 Project results

We created a web-based application that harnesses deep learning algorithms to detect plant diseases, designed to help farmers and agronomists quickly diagnose and make treatment decisions. The application incorporates a pre-trained Convolutional Neural Network (CNN) built with TensorFlow and Keras, which can categorize plant images into different disease types. Key features and functions of the system include:

3.1.1 System Architecture and Implementation

- Model Integration: We employed a TensorFlow/Keras model trained on a dataset encompassing multiple plant diseases. The model, loaded into the Flask web framework, accepts image uploads from users for real-time disease prediction.
- Class Definition: The application supports prediction across multiple classes of plant diseases, encompassing common issues such as fungal infections, bacterial spots, and viral diseases specific to various crop types (e.g., tomatoes, apples, grapes).

3.1.2 Workflow and User Interaction

- Image Upload and Processing: Users upload images through a web interface, processed server-side using OpenCV for image manipulation tasks such as resizing and color space conversion

(BGR to RGB).

- Prediction and Display: Uploaded images are analyzed by the model to determine the most likely disease class. Predictions are returned to the user interface alongside the original image, providing a visual diagnosis aid.

3.1.3 Practical Application and Benefits

- User Interface: The web interface simplifies interaction, allowing users to quickly upload images and receive real-time diagnostic feedback without requiring specialized software or extensive technical knowledge.

- Decision Support:Utilizing deep learning, the application improves both the speed and precision of disease detection, enabling agricultural professionals to quickly apply specific treatment measures. This helps reduce crop losses and maximize yield.

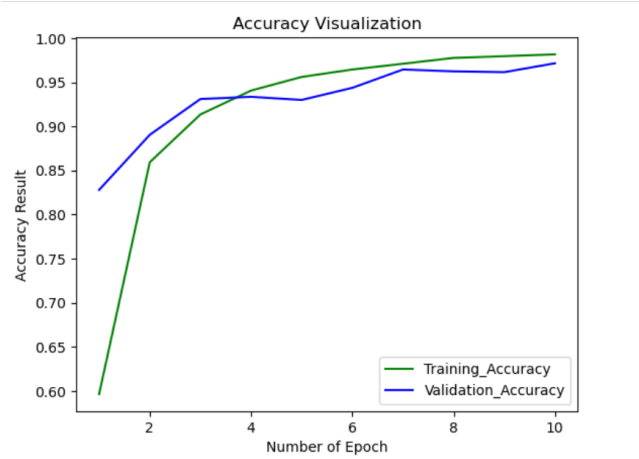
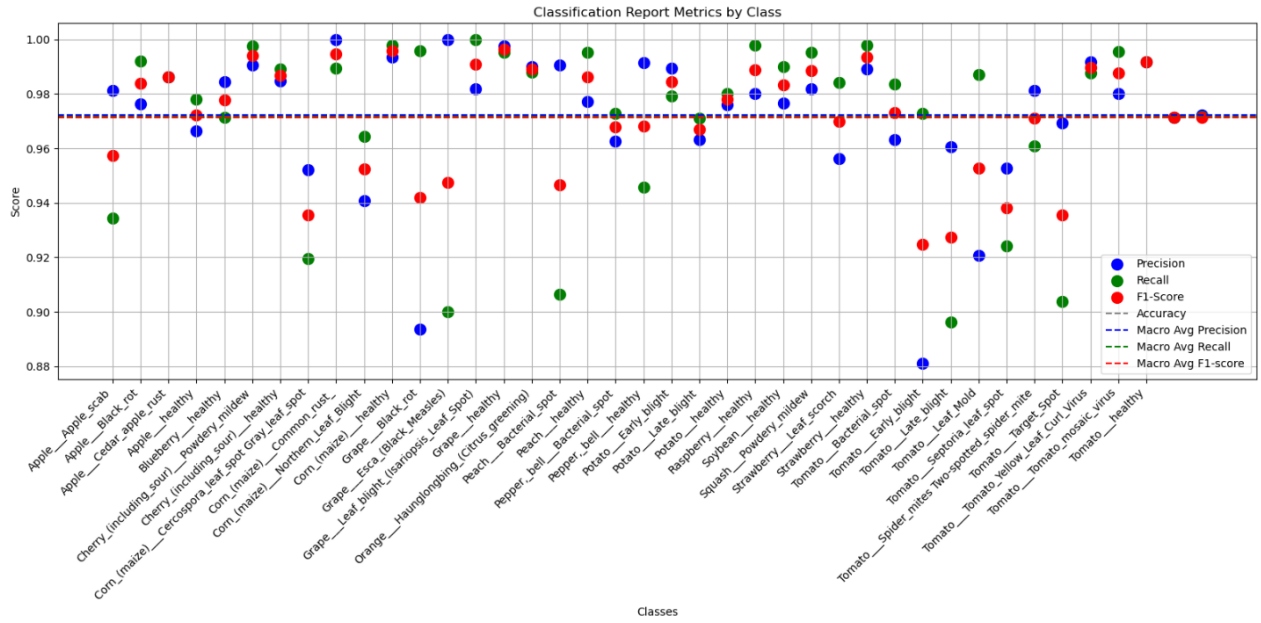


Figure 3.1: Accuracy Visualization

Post-training evaluation includes metrics like accuracy, precision, recall, and F1-score (see Fig. 3.1 and Fig. 3.2). These metrics evaluate how well the model classifies each disease category and offer insights into its strengths and weaknesses in performance.

The application of the trained model extends beyond the lab environment to real-world agricultural scenarios. By deploying the model on edge devices or cloud platforms, farmers can benefit from timely and accurate disease detection, enabling early intervention and targeted management practices. This not only enhances crop productivity and quality but also supports sustainable agriculture by




```
Accuracy: 0.9714887320737537
Macro Average Precision: 0.9722798566821991
Macro Average Recall: 0.9715647972017978
Macro Average F1-Score: 0.9715058427317667
```

Figure 3.2: Classification Report Metrics by Class

decreasing the dependence on chemical treatments and minimizing economic losses caused by crop diseases.

The developed application marks a major progress in precision agriculture, connecting sophisticated machine learning methods with practical agricultural requirements. By combining deep learning with web technology, we offer a scalable and accessible tool for diagnosing plant diseases, supporting sustainable farming practices and global food security efforts.

Chapter 4

Conclusion

Creating a web-based application for plant disease detection using Deep Learning marks a significant step forward in agricultural technology. By leveraging a pre-trained Convolutional Neural Network (CNN), this application can accurately classify plant images into different disease categories. This capability helps farmers and agronomists quickly diagnose and treat plant diseases, thereby improving disease management and reducing crop loss.

The application's accurate and timely disease detection supports sustainable farming practices and enhances food security. Its user-friendly interface ensures that even those with limited technical skills can benefit from this advanced technology.

Overall, this project demonstrates the transformative potential of AI in agriculture, showcasing how machine learning can revolutionize traditional farming methods for better efficiency and sustainability.

References

[1] OpenCV Documentation, “Image Processing and Analysis for Plant Disease Detection.” OpenCV, 2024.

[2] Goodfellow, I., Bengio, Y., Courville, A., “Deep Learning.” MIT Press, 2016.

[3] Kumar, V., Kaur, A., Kumar, D., “Plant Disease Detection Using Convolutional Neural Networks: A Review.” 2024.

[4] Simonyan, K., Zisserman, A., “Very Deep Convolutional Networks for Large-Scale Image Recognition.” arXiv preprint arXiv:1409.1556, 2014.

[5] TensorFlow Documentation, “Convolutional Neural Networks (CNNs) in TensorFlow.” Google, 2024.