A Report on

Plant Leaf Disease Detection and Classification Using AI and Computer Vision Techniques

Submitted for partial fulfilment of award of

BACHELOR OF TECHNOLOGY

degree

In

Computer Science & Engineering (Data Science)

By

STUDENT
Shikhar Sharma 2100301540071
Piyush Pandey 2100301540052
Ujjawal Singh 2100301540079
Yash Srivastava 2100301540087

INDERPRASTHA ENGINEERING COLLEGE, GHAZIABAD,

Dr. A P J ABDUL KALAM TECHNICAL UNIVERSITY LUCKNOW

December 2024

Certificate

Certified that **Shikhar Sharma, Piyush Pandey, Ujjawal Singh, Yash Srivastava** has carried out the project work presented in this report entitled **"Plant Leaf Disease Detection and Classification Using AI and Computer Vision Techniques"** for the award of **Bachelor of Technology** from Inderprastha Engineering College, Ghaziabad, under my supervision. The report embodies result of original work and studies carried out by Student himself/herself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else.

Submitted to: Ms. Rati Goyal

Date: 2nd January, 2025

Acknowledgement

We would like to extend my sincere gratitude to everyone who contributed to the successful completion of this project, “Machine Learning-Based Income Classification System of Individuals.”

We would also like to thank Dr. Tripti Sharma (HOD, Computer Science and Engineering (Data Science)) for her constant support during the development of the project.

Also, we are immensely grateful to my project mentor Ms. Rati Goyal, whose guidance, expertise, and insightful feedback have been important throughout this project. Her support and encouragement inspired me to take on challenges and pursue continuous improvement.

We would also like to thank our group members, whose constructive discussions and shared knowledge provided fresh perspectives and a collaborative learning environment.

Additionally, we are grateful to the organizations and researchers who provided the publicly accessible datasets that made this project possible. Their contributions to open-source data and knowledge sharing are commendable and essential to the advancement of machine learning applications.

Thank you all for your invaluable support and guidance.

Abstract

Agriculture is a vital industry and a primary source of income for many nations. Diseases in plants caused by pathogens such as viruses, fungi, and bacteria result in significant financial losses globally in the agricultural sector. Monitoring and ensuring the quality and quantity of crops require effective plant disease management.

Disease symptoms are often visible on various plant parts, with leaves being the most affected. Researchers have utilized computer vision, deep learning, few-shot learning, and soft computing techniques to automatically identify plant diseases using leaf images. These technologies enable farmers to take timely and accurate actions to prevent declines in crop yield and quality. By automating disease detection, these methods overcome limitations such as subjective feature selection, manual feature extraction, and inefficiencies in traditional approaches, enhancing both research speed and technology effectiveness.

Additionally, molecular techniques have been developed to mitigate pathogenic threats. This review examines the use of machine learning, deep learning, and few-shot learning for automated plant disease detection, highlights diagnostic techniques to prevent disease, and explores future directions in disease classification.

TABLE of CONTENTS

INTRODUCTION

Agriculture plays a vital role in ensuring food security and economic stability worldwide. With the global population continuously growing, the demand for agricultural productivity has never been higher. However, one of the primary challenges faced by farmers is the onset of plant diseases, which can lead to reduced yields, compromised crop quality, and significant economic losses. Plants also play a vital role in maintaining environmental balance by producing oxygen through photosynthesis. However, plant diseases, particularly those affecting leaves, can severely impact plant health and disrupt food production. A historical example is the 1845 Irish Potato Famine, which caused 1.2 million deaths due to crop failure [1]

Various pathogens contribute to plant diseases and can be identified using molecular techniques such as DNA analysis, PCR, MPG (Modified Panchayat Mixture), ELISA (Enzyme-Linked Immunosorbent Assay), FISH (Fluorescence in Situ Hybridization), and IF (Immunofluorescence) methods. This review paper provides a comparative analysis of machine learning, deep learning, and few-shot learning in plant disease detection. It also examines segmentation, feature extraction, and classification techniques alongside molecular diagnostic tools [2].

This project, "Plant Leaf Disease Detection and Classification Using AI and Computer Vision Techniques," aims to develop an intelligent and scalable solution that uses cutting-edge AI and computer vision methods to identify plant diseases from images of leaves. The system's accuracy and efficiency can empower farmers with actionable insights, promote sustainable farming practices, and ultimately contribute to global food security

Problem Statement: Traditional methods of plant disease detection rely on manual inspection by agricultural experts or farmers, which can be time-consuming, subjective, and prone to errors. Moreover, in many rural and remote areas, access to expert advice may be limited, leading to delays in identifying and addressing disease outbreaks.

Additionally, certain plant diseases share similar symptoms, making manual identification challenging even for experienced professionals. This lack of accurate and timely diagnosis often results in inappropriate treatments, further exacerbating crop losses and increasing costs.

This project seeks to address these challenges by developing an AI-powered system for plant leaf disease detection and classification using computer vision techniques. The goal is to create a scalable, efficient, and user-friendly solution that can analyze leaf images, identify diseases with high accuracy, and provide actionable recommendations to farmers and agricultural stakeholders.

FEASIBILITY STUDY

1. Technical Feasibility
   - Availability of Technology: The project leverages readily available technologies, such as deep learning frameworks (e.g., TensorFlow, PyTorch), computer vision libraries (e.g., OpenCV), and cloud computing platforms for scalable deployment.
   - Data Requirements: Datasets of diseased and healthy plant leaves are widely available in agricultural research repositories. Additionally, smartphone cameras and digital imaging devices make it easy to collect further data.

Conclusion: Technically feasible with current tools, resources, and expertise.

2. Economic Feasibility
   - Development Costs: Initial costs include data acquisition, model training, and system development. These can be minimized using open-source tools and publicly available datasets.
   - Operational Costs: Cloud deployment and model updates will incur recurring expenses, but these are scalable based on user demand.

Conclusion: Economically feasible with potential for substantial ROI (Return of Investment).

3. Operational Feasibility
   - User Accessibility: The system can be deployed as a mobile application or web-based platform, making it easily accessible to farmers, agronomists, and agricultural stakeholders.
   - Ease of Use: Intuitive interfaces with image upload options and simple outputs (e.g., disease name, severity, and recommendations) will ensure widespread adoption.

Conclusion: Operationally feasible with user-friendly design and scalable deployment.

4. Legal and Ethical Feasibility
   - Data Privacy: Ensuring the anonymity and security of images uploaded by users will be crucial. Standard data protection measures will address privacy concerns.
   - Regulatory Compliance: The system must comply with regional agricultural policies and technology regulations, especially if pesticides or treatments are recommended.
   - Bias and Fairness: Ensuring the AI model is trained on diverse datasets to avoid bias is essential to ensure fair and accurate predictions.

Conclusion: Feasible with adherence to ethical AI practices and data protection laws.

Overall Feasibility: The project is technically, economically, operationally, and legally feasible. Its implementation can have a transformative impact on agriculture, offering a cost-effective, scalable, and accessible solution for plant disease management.

SOFTWARE REQUIREMENT ANALYSIS

1. **Functional Requirements**
    a. Image Upload and Input
        - Users must be able to upload images of plant leaves via a web or mobile application.
        - Support for common image formats (e.g., JPEG, PNG).
        - Option to capture images directly using a device's camera.

    b. Image Processing and Disease Detection
        - Preprocess the uploaded image (e.g., resizing, noise reduction, normalization).
        - Analyze the image using trained AI models to detect and classify diseases.
        - Return disease classification results with confidence scores.

2. **System Requirements**
    a. AI Model and Libraries
        - Model: Pretrained Convolutional Neural Networks (CNNs) such as ResNet, MobileNet, or custom-trained models.
        - Libraries: TensorFlow, PyTorch, or Keras for model development.

3. **Cloud Services**
    a. Deployment: AWS, Azure, or Google Cloud for hosting and scalability.
    b. Storage: Cloud storage for user-uploaded images and datasets.

4. **Testing Frameworks**
    a. AI Model Testing: Confusion matrix, accuracy metrics, and robustness tests.

5. **Integration Requirements**
    a. External APIs: Agricultural databases for updated disease management practices.
    b. Data Sources: Integration with research datasets for model training and validation.

6. **Hardware Requirements**
    a. Local Machine Requirements: A machine with at least 8GB RAM and a processor (Intel i3/i5/i7 or AMD Ryzen 5/7).

7. **Model Evaluation Metrics**
    a. Confusion Matrix: Visualizing model performance for classification.
    b. Accuracy Score: Primary metric to gauge model correctness, especially if classes are balanced.
    c. Precision, Recall, F1-score: Additional metrics in case of class imbalance, especially if misclassification costs vary by income category.

IMPLEMENTATION

Step 1: Project setup

- Define constants like image dimensions (img_width, img_height) and batch size (batch_size).
- Specify the dataset directory (PlantVillage) for leaf disease detection.
- Importing libraries

```
import os
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping
```

Step 2: Data Preprocessing

- Use the ImageDataGenerator class to create data generators for:
- Data Augmentation: Apply transformations like rotation, zoom, shift, etc.
- Normalization: Rescale pixel values to the range [0, 1].
- Split the data into training and validation sets using validation_split.
- Load augmented and preprocessed images from the dataset directory with flow_from_directory.

```
def load_data():
    data_dir = dataset["data_dir"]
    datagen = ImageDataGenerator(
        rescale=1.0 / 255,
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest',
        validation_split=0.2
    )
```

```
    train_generator = datagen.flow_from_directory(
        os.path.join(data_dir, 'train'),
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='categorical',
        subset='training'
    )

    validation_generator = datagen.flow_from_directory(
        os.path.join(data_dir, 'train'),
        target_size=(img_width, img_height),
        batch_size=batch_size,
        class_mode='categorical',
        subset='validation'
    )
    return train_generator, validation_generator
```

Step 3: Building the model (Convolutional Neural Network Model)

1. Define a Convolutional Neural Network (CNN) using Sequential with:
   - **Convolutional Layers**: Extract features from images.
   - **Pooling Layers**: Downsample feature maps.
   - **Fully Connected Layers**: Map features to class probabilities.
   - **Dropout Layer**: Prevent overfitting.
2. Compile the model with:
   - Optimizer: adam
   - Loss Function: categorical_crossentropy
   - Metric: accuracy

```
def build_model(num_classes):
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(img_width, img_height, 3)),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(num_classes, activation='softmax')
```

```
    ])
    model.compile(optimizer='adam',                    loss='categorical_crossentropy',
metrics=['accuracy'])
    return model
```

Step 4: Model Training

1. Check if a pre-trained model exists (plant_disease_model.h5).
   - If found, load the model using load_model.
   - If not, proceed to train a new model.
2. Train the model using the training and validation data generators with:
   - Early stopping to prevent overfitting.
   - A maximum of 25 epochs.
3. Save the trained model to a file (plant_disease_model.h5).

```
def get_model():
    model_filename = "plant_disease_model.h5"
    train_generator, validation_generator = load_data()
    num_classes = len(train_generator.class_indices)

    if os.path.exists(model_filename):
        print(f"Loading pre-trained model for {dataset['task_name']}...")
        model = load_model(model_filename)
    else:
        print(f"Training new model for {dataset['task_name']}...")
        model = build_model(num_classes)
        early_stop          =          EarlyStopping(monitor='val_loss',          patience=5,
restore_best_weights=True)
        model.fit(
            train_generator,
            epochs=25,
            validation_data=validation_generator,
            callbacks=[early_stop]
        )
        model.save(model_filename)
        print(f"Model saved as {model_filename}.")

    return model, train_generator
```

Step 5: Evaluating Validation set

The evaluate_model function computes the following metrics on the test dataset:

- Classification Report: Includes precision, recall, and F1 score for each class.
- Confusion Matrix: Visualizes the number of correct and incorrect predictions for each class.
- Overall Metrics: Accuracy, weighted precision, recall, and F1 score.

```
def evaluate_model(model, validation_generator):
    print("\nEvaluating model on validation set...")
    validation_generator.reset()
    predictions = model.predict(validation_generator)
    y_pred = np.argmax(predictions, axis=1)
    y_true = validation_generator.classes
    class_labels = list(validation_generator.class_indices.keys())

    print("Classification Report:")
    print(classification_report(y_true, y_pred, target_names=class_labels))

    print("Confusion Matrix:")
    print(confusion_matrix(y_true, y_pred))
```

Step 6: Predicting the input image

1. Implement a user-friendly prediction loop:
    - Prompt the user to input the path of an image for prediction.
    - Preprocess the input image (resize and normalize).
    - Use the trained model to predict the class label.
    - Display the image and the predicted label using matplotlib.

```
def predict_images_in_loop(model, class_indices):
    while True:
        test_image_path = input("\nEnter the path of an image to predict (or type 'exit' to
quit): ")
        if test_image_path.lower() == 'exit':
            print("Exiting prediction loop.")
            break
        try:
            img = Image.open(test_image_path).resize((img_width, img_height))
            img_array = np.expand_dims(np.array(img) / 255.0, axis=0)
            prediction = model.predict(img_array)
            predicted_label = list(class_indices.keys())[np.argmax(prediction)]
            print(f"Predicted Label: {predicted_label}")
```

```
 plt.imshow(img)
        plt.title(f"Predicted: {predicted_label}")
        plt.axis('off')
        plt.show()
    except Exception as e:
        print(f"Error processing image: {e}")
```

Step 7: Main program

1. Define a main function to:
    - Print the task name (Leaf Disease Detection).
    - Train or load the model using get_model.
    - Use predict_images_in_loop to allow interactive predictions.
2. Start the program with if __name__ == "__main__": main().

```
def main():
    print("Starting Plant Disease Detection...")
    model, train_generator = get_model()
    predict_images_in_loop(model, train_generator.class_indices)

if __name__ == "__main__":
    main()
```

RESULT & OUTPUT

Evaluation of validation set

```
Evaluating model on validation set...
129/129 ━━━━━━━━━━━━━━━━━━━━ 42s 322ms/step
Classification Report:
```

	precision	recall	f1-score	support
Pepper Bell - Bacterial Spot	0.97	0.93	0.95	199
Pepper Bell - Healthy	0.92	0.98	0.95	295
Potato - Early Blight	0.93	0.98	0.96	200
Potato - Healthy	0.00	0.00	0.00	30
Potato - Late Blight	0.00	0.00	0.00	200
Tomato - Bacterial Spot	0.95	0.95	0.95	425
Tomato - Early Blight	0.83	0.87	0.85	200
Tomato - Healthy	0.01	0.01	0.01	318
Tomato - Late Blight	0.02	0.01	0.01	381
Tomato - Leaf Mold	0.01	0.01	0.01	190
Tomato - Mosaic Virus	0.00	0.00	0.00	74
Tomato - Septoria Leaf Spot	0.00	0.00	0.00	354
Tomato - Spider Mites Two Spotted Spider Mite	0.00	0.00	0.00	335
Tomato - Target Spot	0.00	0.00	0.00	280
Tomato - Yellow Leaf Curl Virus	0.00	0.00	0.00	641
accuracy			0.31	4122
macro avg	0.31	0.32	0.31	4122
weighted avg	0.30	0.31	0.30	4122

Fig. 1

Confusion Matrix

```
Confusion Matrix:
[[185   8   0   0   0   1   1   0   0   3   0   0   1   0   0]
 [  1 290   0   0   1   1   0   2   0   0   0   0   0   0   0]
 [  0   0 197   0   0   0   1   2   0   0   0   0   0   0   0]
 [  0   6   0   0  21   0   0   0   0   0   3   0   0   0   0]
 [  0   0   1 182   0   0   5  10   0   0   0   1   0   0   1]
 [  1   0   0   1   0 405   3   0   0   1   0   2  12   0   0]
 [  2   1   5   0   0   2 174   8   0   3   0   3   2   0   0]
 [  0   0   0   0   0   0   0   3   0   0   0   2   0   0 313]
 [  0   2   2   8   0   4  14 346   3   0   1   0   0   0   1]
 [  0   0   0   0   0   0   2   1 184   2   1   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   1   0   0   0  73   0]
 [  2   3   7   0   0   6   7   3   1 324   0   0   0   1   0]
 [  0   2   0   0   0   0   1   2   1   0 313  15   1   0   0]
 [  0   2   0   0   0   2   1   1   0   6  21 242   0   0   5]
 [  0   0   0   0   0   4   0   0   0   0   3   0 634   0   0]]
```

Fig. 2

Model showing input image with its predicted leaf disease

```
Enter the path of an image to predict (or type 'exit' to quit):  Sample Images/1.JPG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 76ms/step
Predicted Label: Pepper Bell - Bacterial Spot
```

Fig. 3

```
Enter the path of an image to predict (or type 'exit' to quit):  Sample Images/2.JPG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 86ms/step
Predicted Label: Pepper Bell - Healthy
```

Fig. 4

```
Enter the path of an image to predict (or type 'exit' to quit):  Sample Images/3.JPG
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 75ms/step
Predicted Label: Tomato - Early Blight
```

Fig. 5

CONCLUSION & RECOMMENDATIONS

The emergence of plant pathogens poses a significant threat to global food security, ecosystems, and economies. Factors such as globalization, increased mobility, vectors, climate change, and pathogen evolution have accelerated the spread of invasive plant pathogens. To address agricultural losses, the development of automated approaches for plant disease detection and classification is urgently needed. This review explores the use of machine learning (ML), deep learning (DL), and few-shot learning (FSL) for automated plant disease recognition. It highlights key methodologies, including acquisition, preprocessing, segmentation, feature extraction, and classification. While many studies rely on RGB images, some have adopted hyperspectral imaging for plant leaves, which offers the advantage of detecting microscopic symptoms without requiring labeled datasets. These automated techniques have facilitated timely advancements in research.

Recommendations

1. Data Collection and Quality
   - Diverse Dataset: Ensure the dataset includes images of various plant species, diseases, and healthy leaves, captured in different lighting conditions, angles, and backgrounds to improve model generalization.
   - High-Quality Images: Use high-resolution images to enhance feature extraction and classification accuracy.

2. Model Development
   - Pretrained Models: Leverage pretrained models like ResNet, MobileNet, or EfficientNet to reduce training time and improve performance.
   - Custom Models: Consider developing a custom Convolutional Neural Network (CNN) tailored to the dataset if pretrained models do not yield satisfactory results.

3. System Design
   - User-Friendly Interface: Design an intuitive and simple interface.
   - Mobile-First Design: Prioritize a mobile-friendly design as smartphones are widely used.
   - Offline Mode: Include an offline mode for users in areas with limited internet connectivity.

4. Deployment and Scalability
   - Cloud Deployment: Use cloud platforms like AWS, Azure, or Google Cloud to ensure scalability and reliability.

5. Ethical Considerations
   - Privacy Protection: Implement strong data encryption and privacy policies to protect user-uploaded images and data.
   - Open Access: Consider providing a free basic version of the application for small-scale farmers to promote inclusivity and social impact.

REFERENCES

[1] FAO, IFAD, UNICEF, WFP, WHO, The State of Food Security and Nutrition in the World 2023. Geneva, Switzerland, Jul. 2023.

[2] J. Liu and X. Wang, "Plant diseases and pests detection based on deep learning: A review," Plant Methods, vol. 17, no. 1, pp. 1–18, Dec. 2021.

[3] K. Bashir, M. Rehman, and M. Bari, "Detection and classification of rice diseases: An automated approach using textural features," Mehran Univ. Res. J. Eng. Technol., vol. 38, no. 1, pp. 239–250, Jan. 2019.

[4] J. Parraga-Alava, K. Cusme, A. Loor, and E. Santander, "RoCoLe: A robusta coffee leaf images dataset for evaluation of machine learning-based methods in plant diseases recognition," Data Brief, vol. 25, Aug. 2019, Art. no. 104414, doi: 10.1016/j.dib.2019.104414.

[5] A. N. I. Masazhar and M. M. Kamal, "Digital image processing technique for palm oil leaf disease detection using multiclass SVM classifier," in Proc. IEEE 4th Int. Conf. Smart Instrum., Meas. Appl. (ICSIMA), Nov. 2017, pp. 1–6.

[6] S. Kaur, S. Pandey, and S. Goel, ‘‘Semi-automatic leaf disease detection and classification system for soybean culture,’’ IET Image Process., vol. 12, no. 6, pp. 1038–1048, Jun. 2018

[7] P. Goncharov, G. Ososkov, A. Nechaevskiy, A. Uzhinskiy, andI. Nestsiarenia, ‘‘Disease detection on the plant leaves by deep learning,’’ in Advances in Neural Computation, Machine Learning, and Cognitive Research II. Cham, Switzerland: Springer, 2019, pp. 151–159.

[8] M. Francisco, F. Ribeiro, J. Metrôlho, and R. Dionísio, ‘‘Algorithms and models for automatic detection and classification of diseases and pests in agricultural crops: A systematic review,’’ Appl. Sci., vol. 13, no. 8, p. 4720, Apr. 2023.

[9] M. Bhagat and D. Kumar, "Efficient feature selection using BoWs and SURF method for leaf disease identification," Multimedia Tools and Applications, vol. 82, no. 18, pp. 28187–28211, Feb. 2023, doi: 10.1007/s11042-023-14625-5.