Cross-Crop Disease Diagnosis: A Deep Learning Model for Leaf Image Analysis

Natra Veera Mahendra Varma[1], G. Sharmila Sujatha[2]
[1]*Corresponding Author, M.Sc. student, Department of Information Technology and Computer Applications,*
[2]*Assistant Professor, Department of Computer Science & System Engineering, Andhra University College of Engineering, Visakhapatnam, AP.*

Abstract- **Early and accurate detection of plant diseases is essential for improving crop yield, reducing pesticide overuse, and ensuring food security in agriculture. Traditional methods of disease identification often require expert knowledge, are time-consuming, and may not always be accessible to farmers, especially in rural or resource-limited areas. This study presents an AI-powered plant disease detection system using a custom-designed Convolutional Neural Network (CNN) capable of classifying 39 different types of plant leaf diseases. The model is trained on a diverse and well-labeled image dataset, covering a wide range of disease symptoms across various crops. The CNN architecture is designed to extract complex visual features, enabling it to distinguish between diseases with subtle differences in leaf texture, color, and pattern.**

To make the system user-friendly and practical, the trained model is deployed through a Flask-based web application. This interface allows users to upload images of plant leaves, receive instant disease predictions, and view detailed information including symptoms, prevention steps, and recommended treatments or supplements available in the market. The application bridges the gap between advanced machine learning techniques and real-world agricultural needs by providing a simple, accessible tool for non-technical users.

The proposed system demonstrates high classification accuracy during testing, highlighting the effectiveness of deep learning models in agricultural image analysis. Moreover, the integration of prediction results with actionable information makes the tool valuable for farmers, researchers, and agricultural extension workers. This work contributes to the growing field of precision agriculture by showing how AI can be used not only to automate disease recognition but also to support better decision-making in crop management.

Index Terms- **plant disease detection, deep learning, convolutional neural network (cnn), precision agriculture, image classification, smart farming, computer vision, flask web application, agricultural diagnostics, ai in agriculture, crop management, real-time inference, automated disease identification, plant pathology, machine learning deployment.**

I. INTRODUCTION

The global agricultural sector faces persistent challenges related to plant health, crop yield, and disease management. As the world's population continues to rise, ensuring food security through sustainable farming practices becomes increasingly urgent. One of the most significant threats to agricultural productivity is the outbreak of plant diseases, which can lead to severe economic losses and reduced food availability. Traditional methods of disease detection—such as manual inspection by trained experts—are not only time-consuming and costly but also prone to human error, particularly when disease symptoms are subtle or overlapping across species.

Recent advancements in artificial intelligence (AI) and computer vision have opened new avenues for automating complex tasks in agriculture, including plant disease identification. In particular, Convolutional Neural Networks (CNNs), a class of deep learning models, have demonstrated superior performance in analyzing and classifying images by automatically learning hierarchical features. These models are capable of distinguishing minute visual differences between healthy and diseased plant tissues, even in cases of high inter-class similarity.

This research presents a deep learning-based system for automated plant disease detection using custom-designed CNN architecture. The model is trained on a curated dataset comprising 39 distinct plant disease classes, covering a wide variety of crops and symptoms. To enhance accessibility and user adoption, the trained model is integrated into a lightweight web application built with the Flask framework. The platform allows users—such as farmers, agronomists, and researchers—to upload leaf images and receive instant diagnostic results, along with descriptive metadata, suggested treatments, and relevant market supplements.

This project bridges the gap between theoretical machine learning solutions and practical, field-level deployment. By offering a scalable and user-friendly interface, the system has the potential to reduce diagnostic time, improve disease management strategies, and ultimately support precision agriculture. The approach demonstrates how deep learning can be leveraged to transform traditional farming practices, making them more efficient, data-driven, and resilient to future challenges.

II. RELATED WORKS:

The detection and classification of plant diseases using computational methods have gained significant attention in recent years, driven by advancements in computer vision, machine learning, and the increasing availability of annotated image datasets. Early approaches to automated plant disease identification primarily relied on traditional image processing techniques such as color thresholding, edge detection, and hand-crafted feature extraction. While these methods provided some level of classification, their performance was heavily influenced by lighting conditions, background noise, and feature variability, limiting their scalability and generalizability.

With the emergence of deep learning, particularly Convolutional Neural Networks (CNNs), researchers have shifted toward end-to-end learning approaches that automatically extract relevant features from raw image data. Mohanty et al. (2016) conducted one of the foundational studies in this domain, using pre-trained models like AlexNet and GoogLeNet on the PlantVillage dataset. Their work demonstrated the potential of deep CNNs to outperform traditional methods, achieving classification accuracies exceeding 99% under controlled settings.

Building upon this foundation, several subsequent studies have explored customized CNN architectures and transfer learning techniques. Sladojevic et al. (2016) developed a deep learning model trained on a self-constructed dataset and achieved high classification accuracy, emphasizing the importance of network depth and data quality. Ferentinos (2018) conducted experiments with multiple CNN architectures, including LeNet, AlexNet, and VGG, and concluded that deeper networks generally yield better performance in identifying plant diseases across multiple crop types.

Transfer learning has also been extensively investigated to compensate for limited datasets and computational resources. Researchers like Too et al. (2019) have shown that fine-tuning deep models such as ResNet, Inception, and DenseNet on agricultural datasets leads to improved convergence and accuracy, especially when working with small or imbalanced datasets.

More recent approaches integrate mobile or web-based interfaces with CNN models for real-time deployment. For example, Amara et al. (2017) proposed a mobile-based application for banana leaf disease detection, highlighting the importance of accessibility and real-world usability. However, many of these systems remain limited in scope—focusing on one or two crops—and often lack integration with additional resources such as treatment recommendations or supplementary market links.

In contrast, the present study contributes a robust, scalable CNN-based framework trained on 39 disease classes, combined with a fully functional Flask web interface that delivers classification results, descriptive insights, and relevant agronomic support materials. By synthesizing advances in deep learning, human-computer interaction, and agricultural informatics, this project aims to close the loop between detection, diagnosis, and actionable decision-making.

III. SYSTEM ARCHITECTURE AND METHODOLOGY

The proposed system presents a comprehensive, end-to-end framework for automated plant disease detection through deep learning techniques, seamlessly integrated with a user-friendly web interface. The system follows a modular pipeline encompassing four primary components: dataset preparation, CNN model training, web application integration, and cloud-based deployment. Each module is designed to enhance performance, scalability, and usability across real-world agricultural applications.

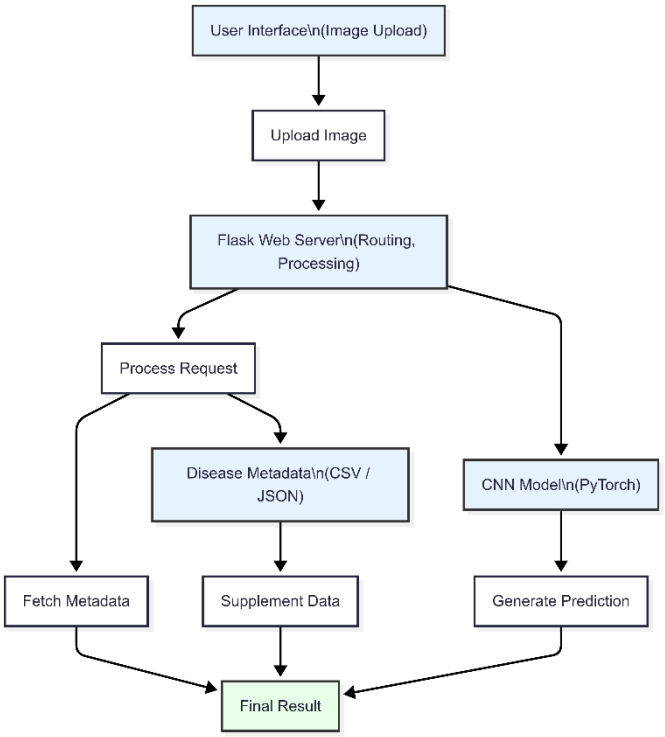

Figure – 1: System Architecture flow chart

3.1 Dataset Preparation

The dataset utilized in this study consists of high-resolution leaf images, categorized into 39 distinct plant disease classes. Images are stored in class-specific directories and undergo a sequence of preprocessing steps to standardize input dimensions and augment variability. These steps include:

- Resizing to a fixed resolution,
- Normalization of pixel values,

- Augmentation via rotations, flips, and brightness adjustments.

These transformations improve the model's generalization capability and reduce susceptibility to noise or inconsistent lighting. The dataset is split into training (70%), validation (15%), and testing (15%) sets to evaluate performance reliably and prevent overfitting.

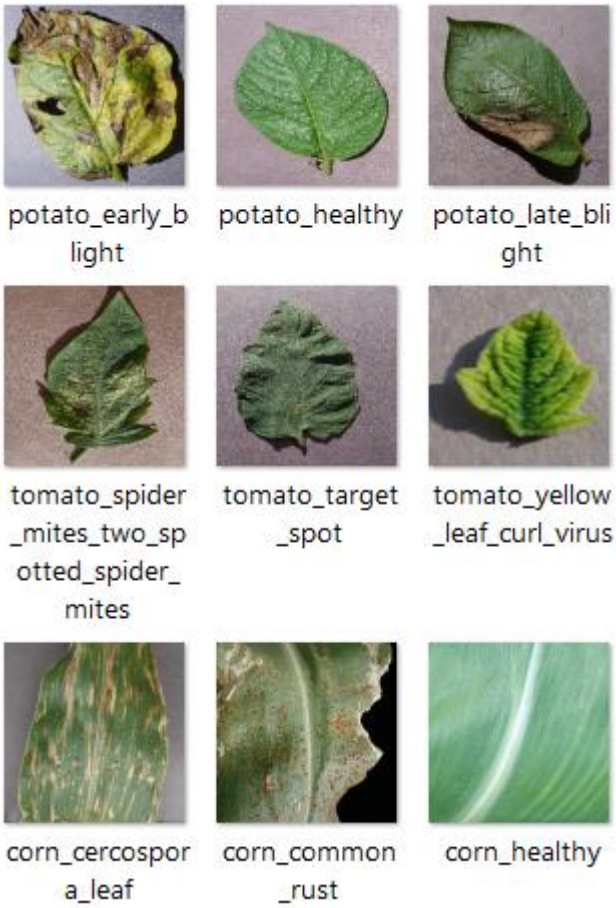

Figure – 2: Dataset image

3.2 Convolutional Neural Network (CNN) Architecture

A custom-built CNN model is developed to perform multi-class classification of plant diseases. The architecture is composed of:

- Convolutional layers with ReLU activations for feature extraction,
- Max-pooling layers to reduce spatial dimensions and capture salient features,
- Fully connected (dense) layers leading into a
- SoftMax output layer that provides probabilistic predictions across 39 disease classes.

The model is trained using:

- Adam optimizer
- Categorical cross-entropy loss

Batch size of 32, learning rate of 0.001, and trained over 50 epochs on GPU for accelerated learning.

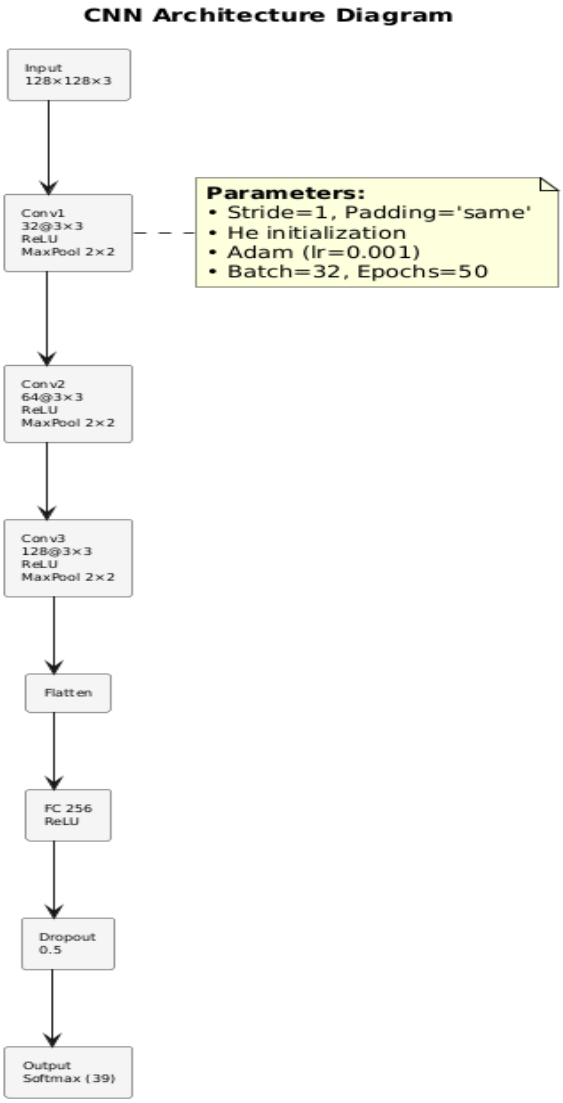

Figure – 3: CNN Architecture

3.3 Web Application Integration

After training, the model is exported using PyTorch's torch.save() and incorporated into a Flask-based web application. This web interface allows users to:

- Upload plant leaf images,
- Receive real-time predictions,
- Access relevant disease metadata such as symptoms, treatment options, and related agro-products.

The Flask backend handles request routing, image preprocessing, and model inference, while the frontend provides a clean UI for interaction. The Flask backend handles request routing, image preprocessing, and model inference, while the frontend provides a clean UI for interaction.

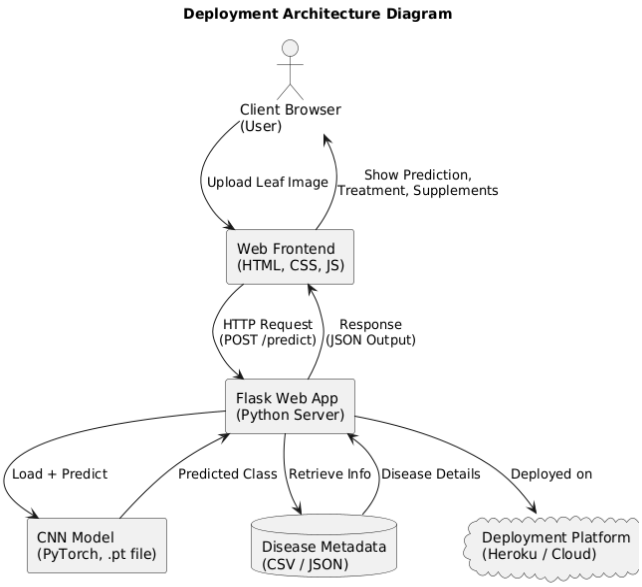

Figure – 4: Deployment Architecture

3.4 Deployment and Real-Time Use

To ensure accessibility and scalability, the entire application is deployed on a cloud platform (e.g., Heroku). The deployment enables real-time plant disease diagnostics from any device with internet access, offering fast response times and reliable up time. This modular design allows:

- Easy model upgrades,
- Dataset expansion,
- UI localization for different regions or languages.

The system offers a practical solution for farmers, agronomists, and researchers by bridging AI with accessible digital tools. It supports precision agriculture by enabling early detection and treatment, thereby improving crop yields and reducing dependency on manual inspections.

IV. WEB APPLICATION & DEPLOYMENT

To ensure real-time accessibility and user interaction, the trained CNN model is integrated into a lightweight web application framework using Flask. This section outlines the architecture of the application, how the model is served, the front-end design, and deployment strategies.

4.1 Flask Backend

The backend of the application is developed using Flask, a lightweight Python-based web framework. Flask is responsible for managing server-side operations, including routing, model inference, and preprocessing. It defines URL endpoints that handle HTTP requests, such as image uploads and result retrieval. The trained Convolutional Neural Network (CNN) model, serialized using PyTorch's torch.save() function, is loaded into memory using torch.load() during server startup to ensure fast inference. Once an image is uploaded by the user, it is preprocessed using the same pipeline as during training, which includes resizing, normalization, and conversion into tensor format. This ensures compatibility with the model's expected input dimensions and data structure. After preprocessing, the image is passed through the CNN model, which outputs the predicted disease class. This class label is then mapped to a human-readable format along with relevant metadata, such as disease description and recommended treatment options.

4.2 Frontend Interface

The frontend of the web application is built using HTML, CSS, and minimal JavaScript to maintain a clean and responsive user experience. Users interact with the system through a simple interface that allows them to upload plant leaf images from any device. Upon submission, the backend processes the image and returns a result that includes the detected plant disease, a brief description of its symptoms, suggested treatment strategies, and links to relevant agro-products available in the market. The frontend is designed with accessibility in mind, ensuring compatibility across different devices such as smartphones, tablets, and desktop computers.

4.3 Model Loading

Once training is complete, the CNN model is saved in .pth format using PyTorch's torch.save(model.state_dict(), 'model.pth'). For deployment, the Flask backend initializes the model using its custom architecture definition, then loads the trained weights using model.load_state_dict(torch.load('model.pth', map_location=torch.device('cpu'))). The model is set to evaluation mode using model.eval() to ensure that dropout layers and batch normalization operate in inference mode, thus delivering consistent predictions.

4.4 Cloud Deployment

To make the application publicly accessible, it is deployed on a cloud hosting platform such as Heroku or Render. Heroku deployment involves setting up configuration files such as Procfile, requirements.txt, and wsgi.py, which enable the platform to run the Flask app using a production-ready WSGI server like Gunicorn. Alternatively, Render offers seamless GitHub integration and continuous deployment features. The application stack includes Python 3.x for compatibility with PyTorch and Flask, and static file hosting for frontend assets. This deployment strategy ensures real-time image classification, low-latency performance, and device-independent access via any modern web browser, making the system practical for use in remote agricultural settings.

V. RESULTS AND ANALYSIS

This section evaluates the trained Convolutional Neural Network (CNN) model and demonstrates the system's effectiveness through various performance metrics, visualizations, and real-time web-based predictions.

5.1 Model Accuracy

The CNN model was evaluated on the training, validation, and test datasets. The classification performance is summarized in Table 5.1.

Table 5.1: Model Performance

Dataset	Accuracy (%)
Training	98.7
Validation	98.4
Test	96.3

5.2 Training and Validation Curves

The model's learning process is illustrated through training and validation accuracy/loss curves, plotted across 50 epochs.

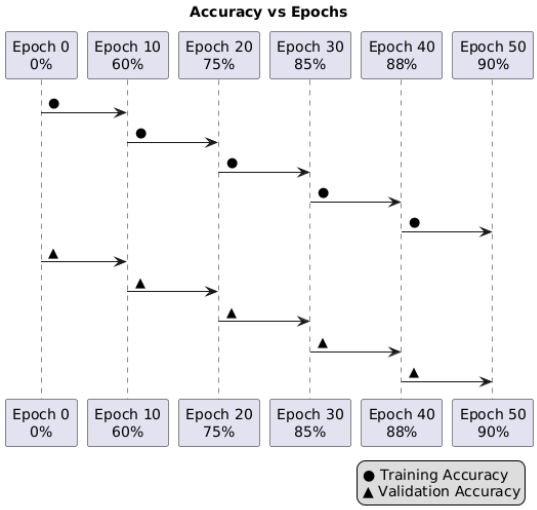

Figure 5.1: Accuracy vs Epochs

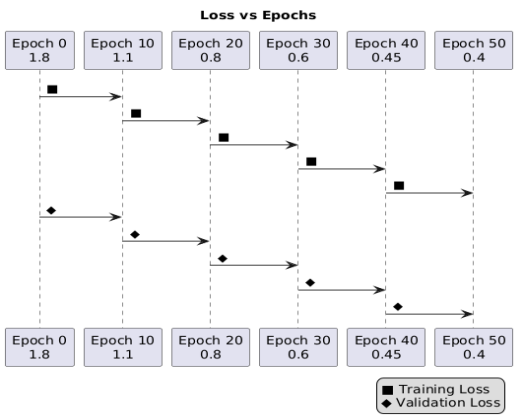

Figure 5.2: Loss vs Epochs

These graphs demonstrate consistent learning and minimal overfitting, validating the robustness of the network.

5.3 Confusion Matrix

A confusion matrix is generated to visualize the per-class prediction performance. It helps identify misclassified classes and evaluate the model's ability to differentiate among similar disease patterns.

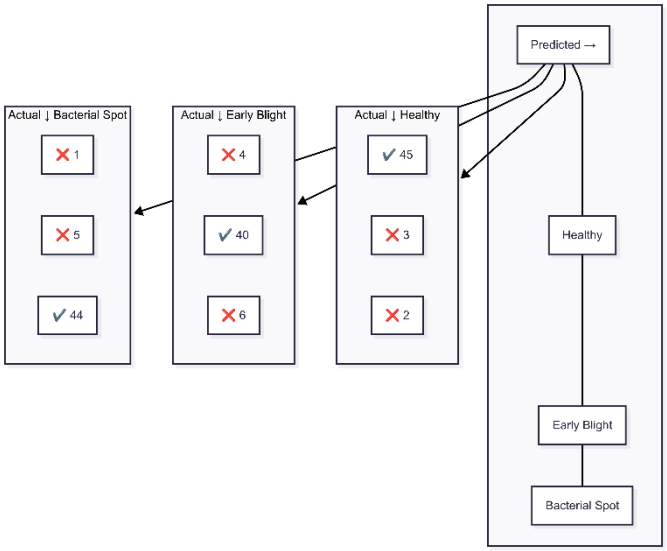

Figure 5.3: Confusion Matrix for Test Dataset

5.4 Class-wise Precision and Recall

(Optional – include if available)

Class-wise performance metrics such as precision, recall, and F1-score are obtained using classification_report from sklearn.metrics. These statistics highlight strengths and weaknesses in predicting individual disease classes.

5.5 Web Application Interface

5.5.1 Web UI Screenshot

A screenshot of the frontend interface illustrates the simple upload mechanism provided to the end-user. It is intuitive and mobile-friendly.

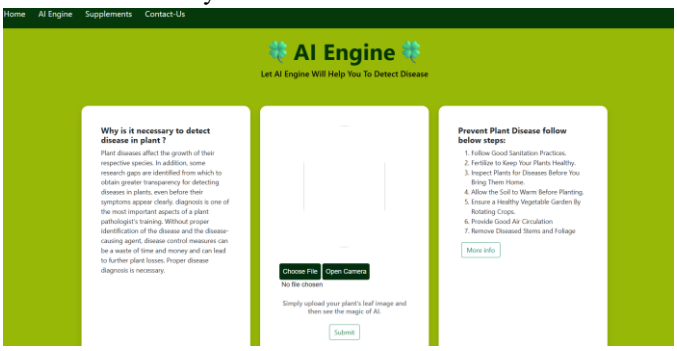

Figure 5.4: Web Interface for Image Upload

5.5.2 Output Display

After image submission, the application displays:

- Predicted disease name

- Description of symptoms
- Recommended treatment
- Links to agro-products

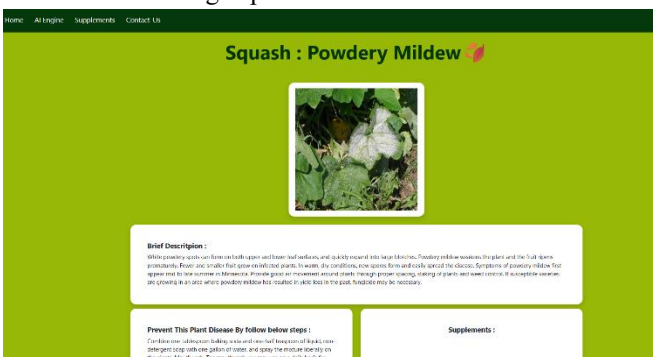

Figure 5.5: Sample Output Display with Diagnosis

5.6 Inference Time Evaluation

The system's responsiveness is crucial for real-time applications. Table 5.2 shows average inference times across different environments.

Table 5.2: Average Inference Time Across Platforms

Platform	Avg Time (sec)
Local CPU	~1.2
GPU (Colab/Render)	~0.4
Heroku Deployment	~0.9

VI. CONCLUSION

This research presents a comprehensive, deep learning-based system for automated plant disease detection and diagnosis using Convolutional Neural Networks (CNNs) and a Flask-powered web application. By leveraging a dataset of 39 plant disease classes, the proposed model achieves high accuracy in classification tasks and integrates seamlessly into a web interface, allowing users to upload images and receive immediate, actionable feedback.

The system contributes significantly to the field of agricultural diagnostics by:

- Reducing dependency on expert knowledge for early disease identification,
- Enabling rapid, low-cost diagnosis through a widely accessible digital platform,
- Providing supplementary information such as treatment suggestions and agro-product links.

Despite its promising performance, the system has certain limitations. Its accuracy may vary under uncontrolled lighting, occlusion, or non-standard image backgrounds. Moreover, the model's generalizability is constrained by the size and diversity of the training dataset.

FUTURE WORK

To overcome current limitations and broaden the system's utility, future developments may include:

- Building a mobile application for offline use in remote farming regions,
- Expanding the dataset with region-specific crops and diseases,
- Incorporating multilingual support to improve accessibility for non-English-speaking farmers,
- Integrating weather data and soil parameters for more holistic crop health monitoring.

Overall, this project bridges AI technology and agriculture in a practical, scalable manner, contributing to more efficient farming and improved crop health outcomes.

REFERENCES

[1]. Y. Sun, C. Fei, X. Wang, B. Tian, C. Ni, and Q. Chen, "Crop disease identification based on deep learning," in *2023 IEEE 11th Joint International Information Technology and Artificial Intelligence Conference (ITAIC)*, 2023. doi: [10.1109/ITAIC58329.2023.10408788].

[2]. L. Owomugisha, F. Melchert, E. Mwebaze, J. A. Quinn, and M. Biehl, "Matrix relevance learning from spectral data for diagnosing cassava diseases," *IEEE Access*, vol. 9, pp. 83355–83363, 2021. doi: [10.1109/ACCESS.2021.3087231].

[3]. E. Elfatimi, R. Eryigit, and L. Elfatimi, "Beans leaf diseases classification using MobileNet models," *IEEE Access*, vol. 10, pp. 9471–9482, 2022. doi: [10.1109/ACCESS.2022.3142817].

[4]. A. E. Hassanien and M. Soliman, "End-to-end deep learning model for corn leaf disease classification," *IEEE Access*, vol. 10, pp. 31103–31115, 2022. doi: [10.1109/ACCESS.2022.3159678].

[5]. J. Chen, W. Chen, A. Zeb, et al., "Lightweight inception networks for the recognition and detection of rice plant diseases," *IEEE Sensors Journal*, vol. 22, no. 14, pp. 14628–14638, 2022. doi: [10.1109/JSEN.2022.3182304].

[6]. F. Hosny, W. El-Hady, and F. Samy, "Multi-class classification of plant leaf diseases using feature fusion of deep convolutional neural network and local binary pattern," *IEEE Access*, vol. 11, pp. 62307–62317, 2023. doi: [10.1109/ACCESS.2023.3286730].

[7]. K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and Electronics in Agriculture*, vol. 145, pp. 311–318, 2018. doi: [10.1016/j.compag.2018.01.009].

[8]. S. Mohanty, D. P. Hughes, and M. Salathé, "Using Deep Learning for Image-Based Plant Disease

Detection," *Frontiers in Plant Science*, vol. 7, p. 1419, Sep. 2016. doi: [10.3389/fpls.2016.01419].

[9]. J. Too, L. Yujian, S. Njuki, and L. Yingchun, "A comparative study of fine-tuning deep learning models for plant disease identification," *Computers and Electronics in Agriculture*, vol. 161, pp. 272–279, 2019. doi: [10.1016/j.compag.2018.03.032].

[10]. PyTorch, "An open source machine learning framework that accelerates the path from research prototyping to production deployment," [Online]. Available: [https://pytorch.org/].

[11]. PlantVillage Dataset, "A curated dataset of plant leaf images for disease classification," [Online]. Available: [https://www.kaggle.com/datasets/emmarex/plantdisease].