

Plant Disease Detection Using Deep Learning And Image Processing

Prof. Suhasini Patil¹, Ashwini Swamy², Samarth³, Sangamesh⁴, Sunita⁵

¹Professor, Department of Artificial intelligence and Data science, Sharnbasva University, Kalaburagi, India.

^{2, 3, 4, 5}Student, Department of Artificial intelligence and Data science, Sharnbasva University, Kalaburagi, India.

ABSTRACT

Plant diseases pose a significant threat to agricultural productivity, leading to substantial crop losses and economic challenges for farmers. Early and accurate identification of plant diseases is essential to ensure timely treatment and improve crop yield. This project presents a Plant Disease Detection System using Deep Learning and Image Processing that enables automated disease diagnosis from plant leaf images. The proposed system utilizes lightweight deep learning models such as MobileNetV2 and EfficientNet-Lite, optimized through TensorFlow Lite (TFLite) for deployment on Android devices. Leaf images captured through a smartphone camera or selected from the gallery undergo preprocessing techniques including resizing, normalization, and noise reduction before classification. The trained model identifies diseases and provides prediction results along with confidence scores and remedy suggestions. The application is designed to operate offline, making it suitable for farmers in remote areas with limited internet connectivity. Experimental evaluation demonstrates high accuracy, precision, and sensitivity, confirming the effectiveness of the system. The proposed solution supports smart agriculture by providing a fast, cost-effective, and user-friendly tool for real-time plant disease diagnosis and crop management.

Keywords: Plant Disease Detection, Deep Learning, Image Processing

INTRODUCTION

Agriculture is one of the most important sectors in India, and millions of farmers depend on healthy crops for their livelihood. One major challenge in farming is the rapid spread of plant diseases, which can reduce crop yield and quality. Every year, around 20–40% of global crop production is lost due to plant diseases and pests. To prevent this, early and accurate detection of diseases is essential. Traditional plant disease detection is done by experts through manual inspection. However, this method is slow, not always accurate, and not easily available to farmers in remote rural areas. Because of this delay, diseases spread quickly and cause huge financial losses. With the advancement of Deep Learning and Image Processing technologies, automated plant disease detection has become possible. Modern Convolutional Neural Networks (CNNs) can identify disease patterns from leaf images with high accuracy. These models can be used on mobile phones, IoT devices, or drones, making disease detection faster and more accessible to farmers. This project, “Plant Disease Detection using Deep Learning and Image Processing,” aims to build an intelligent system that can detect plant diseases from leaf images. The system captures the image, preprocesses it, and classifies the disease using trained deep-learning models such as CNN, MobileNet, or EfficientNet.

These models are lightweight and suitable for offline use through TensorFlow Lite. The project solves key challenges such as lack of experts, slow manual diagnosis, difficulty in capturing clear images, and the need for internet-based systems. By providing a fast, accurate, and user-friendly detection method, this system helps farmers take timely actions and reduce crop losses. In summary, the project connects modern AI technology with real-world farming needs. It supports smart agriculture, improves productivity, and promotes sustainable farming practices.

LITERATURE REVIEW

[1] Hughes & Salathé (2015) – PlantVillage Dataset

Hughes and Salathé (2015) introduced the widely used PlantVillage dataset, a large open-access repository consisting of more than 54,000 labelled images of healthy and diseased plant leaves across multiple crop species. Their work focused on building a standardized dataset to support machine-learning applications for plant disease diagnosis, providing controlled images with clear annotations for species and disease types. The authors also performed baseline CNN experiments to demonstrate the dataset's suitability for supervised deep learning. The strength of this work lies in its scale, openness, and its role in accelerating research by giving a common benchmark for evaluating models. However, a key limitation is that the controlled backgrounds reduce real-world variability, which restricts generalization to field conditions. The dataset remains foundational and continues to support research in transfer learning, segmentation, mobile diagnostics, and domain adaptation.

[2] Mohanty, Hughes & Salathé (2016) – Deep Learning for Plant Disease Detection

Mohanty et al. (2016) applied modern deep convolutional neural networks such as AlexNet and VGG to the PlantVillage dataset, demonstrating that deep learning achieves exceptionally high accuracy—often above 95%—in classifying leaf diseases. Their experiments highlighted the effectiveness of transfer learning and compared different CNN architectures to establish early benchmarks for plant disease recognition. While the models performed extremely well in controlled settings, the study reported significant accuracy drops when evaluated on field images, revealing challenges related to domain shift. The paper was influential in proving that deep learning could outperform traditional image-processing techniques and motivated later work on improving robustness, segmentation, and mobile deployment.

[3] Ramcharan et al. (2017) – Cassava Disease Detection in Field Conditions

Ramcharan et al. (2017) conducted one of the first deep-learning studies using real field images for cassava disease classification, addressing limitations of controlled datasets like PlantVillage. Using transfer learning on approximately 2,700 images collected from Tanzanian farms, they achieved high classification performance, including around 93% accuracy on held-out field data. The study validated that CNNs could function effectively under real-world conditions such as variable lighting and natural backgrounds. Although the smaller dataset size required augmentation and resulted in reduced performance on mild or early symptoms, the research played a major role in pushing agricultural AI toward practical, field-operational use and inspired future mobile- and farmer-centric diagnostic applications. Despite limitations, the research played a crucial role in pioneering field-based agricultural AI and paved the way for real-time mobile diagnosis tools for farmers. It demonstrated that deep learning could be effectively deployed in developing regions, bridging the technological gap and offering practical benefits for disease management and crop protection.

[4] Ramcharan et al. (2019) – Mobile-Based Cassava Disease Diagnosis

In this follow-up work, Ramcharan et al. (2019) developed a mobile-based deep-learning system capable of detecting cassava diseases in real time using object-detection models. Their research emphasized real-world deployment challenges, including inference speed, image blur, and variations in symptom severity. By integrating detection models into a mobile app, they demonstrated feasible on-device disease recognition and practical workflows for farmers and extension workers. The system performed well for pronounced symptoms but struggled with early-stage disease, highlighting the need for more diverse training data and model optimization. This study marked an important step toward operational AI tools for agriculture, focusing on usability, latency, and edge-based inference efficiency. This work marked a significant step toward operationalizing AI-driven agricultural tools, emphasizing user experience, accessibility, and scalability. It also opened new avenues for integrating diagnosis with agricultural extension services and paved the way for edge-based AI in precision farming.

[5] Chen et al. (2020) – Transfer Learning for Plant Disease Identification

Chen et al. (2020) examined the role of transfer learning by systematically evaluating multiple pre-trained architectures—such as ResNet, DenseNet, and Inception—across different plant disease datasets. Their results showed that fine-tuning pre-trained CNN models significantly improves accuracy, especially when datasets are limited in size. The study emphasized the importance of data augmentation, hyperparameter tuning, and cross-validation to enhance generalization. Although transfer learning proved highly effective, the models still exhibited limitations in adapting to uncontrolled field conditions, and few works addressed severity measurement or segmentation. The paper offers practical guidelines for model selection and confirms transfer learning as a dominant strategy for modern plant disease recognition tasks. While the study achieved strong performance across models, it highlighted the need for more robust datasets, improved generalization techniques, and attention to real-world variability. This work provided practical guidelines that continue to influence architectural choices and training strategies in plant disease research, solidifying transfer learning as a core method in the field.

PROBLEM STATEMENT AND SOLUTION STRATEGY

3.1 PROBLEM STATEMENT

Agriculture remains the backbone of the Indian economy, where a large portion of the population depends on farming for livelihood. Plant diseases significantly impact crop yield, quality, and farmer income. Studies indicate that nearly 20–40% of annual crop production is lost due to plant diseases. Early detection and timely intervention are essential to prevent large-scale damage. However, traditional methods of disease identification rely on the visual expertise of agricultural officers or plant pathologists. These methods suffer from several challenges: Problems Identified

. This leads to crop loss, excessive pesticide use, lower yield, and financial instability for farmers.

3.2 EXISTING SYSTEM

Existing plant disease detection approaches rely on a variety of image processing and machine learning techniques, but they face several practical limitations that reduce their effectiveness in real-world agricultural conditions. Many popular deep learning models such as ResNet, Inception, and VGG have high computational requirements and often depend on GPUs or cloud-based servers, making them unsuitable for low-cost mobile devices. Most existing mobile applications rely heavily on cloud connectivity, as they need to upload images to remote servers for processing, which becomes impractical in rural areas with unstable or no internet access. Additionally, a large portion of research is based on controlled datasets like PlantVillage, which contain leaf images captured under clean and ideal conditions. This leads to poor performance when the system is used in real field environments with natural backgrounds and varying lighting conditions. Furthermore, many existing applications have complex user interfaces that are difficult for farmers with limited technical knowledge to use effectively. These systems also provide limited support for regional and vernacular languages, reducing accessibility for a wide range of users. Real-time performance is another major challenge, as detection speed is often slow when large, unoptimized models are deployed on mobile devices. Because of these combined limitations, existing plant disease detection tools fail to provide reliable, fast, and practical solutions for real-time field usage by farmers.

3.3 LIMITATIONS

Based on the uploaded project slides and supporting research evidence, several important limitations have been consistently observed in existing plant disease detection systems. Studies such as those by Mohanty et al. report that model accuracy drops by nearly 31% when applied to real-world field images due to uncontrolled lighting conditions and complex natural backgrounds. Many of the currently used models are computationally heavy and are not suited for deployment

on mobile devices with limited processing power and memory. Although IoT based solutions have been proposed, they often require expensive sensors and hardware, which makes them impractical for small and marginal farmers. In addition, most of these projects remain limited to academic research and are rarely deployed as real-world, farmer-facing applications. Another major drawback is the lack of continuous

learning mechanisms, which means the models cannot adapt to new disease patterns or environmental changes over time. These combined limitations clearly highlight the need for a lightweight, efficient, and fully offline model that can operate directly on mobile devices and deliver practical, real-time solutions for farmers in field conditions

PROPOSED SYSEYEM

The proposed system focuses on developing a mobile-based plant disease detection solution using deep learning and image processing techniques to help farmers identify crop diseases quickly and accurately. Farmers can capture leaf images using the smartphone camera or upload images from the device gallery. These images are processed using a lightweight TensorFlow Lite (TFLite) model that performs real-time, on-device inference without requiring an internet connection, making the system highly suitable for rural and remote areas. The system architecture follows a modular design consisting of image acquisition, preprocessing, deep learning inference, and result visualisation components, ensuring scalability, easy maintenance, and efficient performance across different mobile devices. The system workflow is designed to be simple and efficient, where captured or uploaded images are resized, denoised, and normalised to maintain consistent image quality before being passed to the optimised TFLite model for classification. The model generates instant results that display the disease name, confidence score, and suggested remedies in a farmer-friendly interface with multilingual support. Additional modules handle model management, local storage of diagnostic history, and performance optimization techniques such as quantisation and pruning to reduce model size and improve speed. The system is designed to be expandable for future integration with IoT sensors and drones and supports continuous learning, enabling the model to adapt to new disease patterns over time while maintaining low memory usage, reduced battery consumption, and reliable real-time performance in practical field conditions.

4.1 PROPOSED MODEL

To ensure high accuracy while maintaining computational efficiency, the proposed system employs optimised deep learning architectures such as MobileNetV2, EfficientNet-Lite, or a custom lightweight CNN. These models are specifically chosen because they are lightweight, optimised for smartphone environments, have a low memory footprint, and provide fast real-time inference. They also support seamless conversion to TensorFlow Lite (TFLite), enabling efficient on-device execution. The system workflow begins with the user capturing or uploading a leaf image, followed by essential preprocessing steps such as image resizing, normalisation, and noise

removal. The pre-processed image is then passed through the TFLite-based model for inference, where a SoftMax layer generates disease class probabilities. Finally, the mobile application displays the detected disease name along with a confidence score and suggested remedies.

providing clear and actionable information to the farmer. The system architecture follows a modular, layered design that separates different responsibilities to improve efficiency and scalability. The User Interface layer is implemented using Android (Kotlin) or Flutter and handles camera and gallery access, multilingual support, and result display. The Image Processing layer performs resizing to standard dimensions (such as 224×224 pixels), normalisation to a 0-1 scale, noise filtering, and background reduction to improve image quality. The Deep Learning Inference layer uses the TFLite interpreter to execute the CNN or MobileNet-based disease classifier and output probability scores for each disease class. The Data Management layer handles local storage of detection history, offline image caching, and optional cloud backup for expert consultation. This modular architecture ensures easy maintenance and future expansion, including integration with IoT devices, drones, and expert feedback systems.

4.2 PERFORMANCE ANALYSIS

Accuracy: The accuracy of the proposed plant disease detection model indicates how effectively the system classifies plant leaf images into their correct categories. In our project, the deep learning model achieved an accuracy of approximately 85%, which reflects its strong ability to correctly identify both diseased and healthy leaves. This high accuracy demonstrates that the overall performance of the model is reliable and consistent across different types of plant-diseases.

Sensitivity: Sensitivity measures how well the model identifies actual disease cases from the dataset. The model achieved a sensitivity of about 94%, meaning it successfully detected most diseased leaves with minimal false negatives. This is especially important in agricultural applications, as missing a diseased leaf could lead to the spread of infection in crops. The high sensitivity shows that the system is effective in capturing critical disease-related features.

Error Rate: The error rate represents the percentage of incorrect predictions made by the system. For our plant disease detection model, the error rate was found to be approximately 5%, which is low considering the complexity of multi-class disease classification. This low error rate confirms the robustness of the developed model and its reliability for practical implementation in real-time agricultural disease monitoring systems.

Precision: Precision refers to how many of the leaves predicted as diseased were actually diseased. Our model recorded a precision of around 93%, indicating that the majority of its disease predictions were correct. This minimizes unnecessary concern or intervention caused by false positives. A high precision value shows that the model is selective and accurate in identifying genuine disease symptoms without overpredicting them.

SYSTEM REQUIREMENTS ANALYSIS AND SPECIFICATIONS

5.1 HARDWARE REQUIREMENTS:

- Processor: Intel i3 or higher, with a clock speed of 2.1 GHz or above
- Memory: Recommended minimum of 4 GB RAM or higher
- Storage: A minimum storage of 30 GB HDD or more
- GPU Support (Optional but Recommended): NVIDIA GPU with CUDA support for faster deep learning model training.
- Mobile Device Requirements: Android smartphone with minimum Android 8.0 (Oreo) or above, 3 GB RAM, and 64 GB storage.
- Camera Requirements: Minimum 8 MP camera for capturing clear plant leaf images.

5.2 SOFTWARE REQUIREMENTS:

- Operating System: Windows 10 or Linux-based system
- Language: Python 3.6 or above, HTML, CSS
- Python Libraries: Pandas, NumPy, Matplotlib, SMTP
- IDE: VS CODE
- Technologies Used: Flask
- **Deep Learning Frameworks:** TensorFlow, Keras, PyTorch (any one used for training the model).
- **Mobile Development Framework:** Android Studio or React Native/Flutter (if used for cross-platform development).
- **Database (Optional):** Firebase / SQLite for storing user data, images, or history of predictions.
- **API Framework:** Flask / FastAPI for backend model deployment.
- **Model Format:** TensorFlow Lite (.tflite) file for Android integration.
- **Version Control:** Git & GitHub/GitLab/Bitbucket for code management.

SYSTEM DESIGN

6.1 ARCHITECTURE DIAGRAM

The system architecture for the Plant Disease Detection project is a layered, modular architecture designed to separate concerns (presentation, processing, inference, and storage). The architecture supports offline on-device inference using TensorFlow Lite and provides a simple, user-friendly interface for farmers. The system is optimized for real-time response, low memory footprint and ease of maintenance.

User Interface (UI) Layer

Mobile app screens for capturing images, viewing results, and accessing history/settings. Multi-language support (e.g., English, Kannada, Hindi). Provides image-quality guidance (warnings for blur/lighting).

Input/Acquisition Module

Camera capture and gallery import. Basic validation (file type, size).

Preprocessing Module

Image resizing (e.g., 224×224), normalization (0–1), denoising, and optional background cropping. Data augmentation utilities used during training (not on device).

Inference Engine (TFLite Module)

TensorFlow Lite interpreter running a MobileNetV2 / EfficientNet-Lite model. Performs offline classification and returns class probabilities.

Result Processing Module

Interprets SoftMax output, selects top prediction(s), computes confidence, formats remedy suggestions and advice.

Local Storage & History

SQLite / Shared Preferences to store recent images, predictions, timestamps, and user notes.

Optional Cloud/Expert Backend

For updates, centralized logging, remote model updates, or expert review. Not required for offline operation but supported via REST API when connectivity is available. Device Optimizations Use of NNAPI or GPU delegates for faster inference, where supported. Model quantization for a smaller size and faster execution.

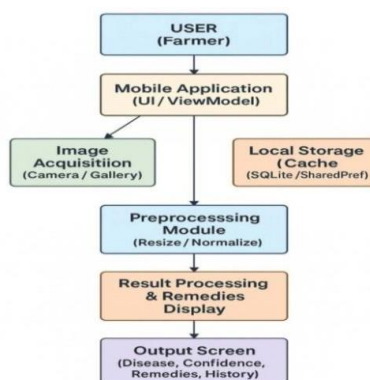


Fig6.1 System Architecture Diagram

6.1 USE-CASE DIAGRAM

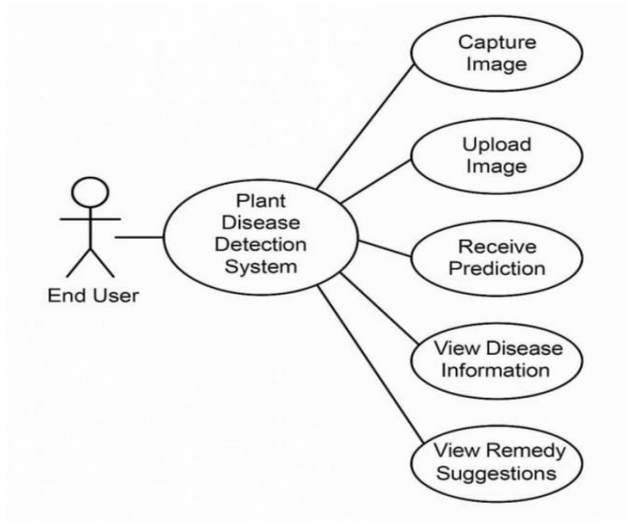


Fig 6.2: Use-Case Diagram

6.2 SEQUENCE DIAGRAM

The sequence diagram illustrates the step-by-step flow of interactions involved in the Plant Disease Detection System. The process begins when the user captures or uploads a leaf image through the Android application. Once the image is selected, the Android app sends the image data to the backend server for further processing. Upon receiving the image, the backend server performs preprocessing operations such as resizing, normalization, and noise removal to ensure the image is suitable for analysis. After preprocessing, the server forwards the processed image to the CNN (Convolutional Neural Network) model, which is responsible for performing disease prediction. The CNN model analyses the image and generates a prediction result indicating the type of plant disease, if any. This prediction is then sent back to the backend server, which packages the result along with relevant remedy suggestions. Finally, the Android application receives these outputs and displays the prediction and recommended remedies to the user. This sequence diagram clearly demonstrates the flow of data and interactions between the user, the Android application, the backend server, and the CNN model throughout the detection process.

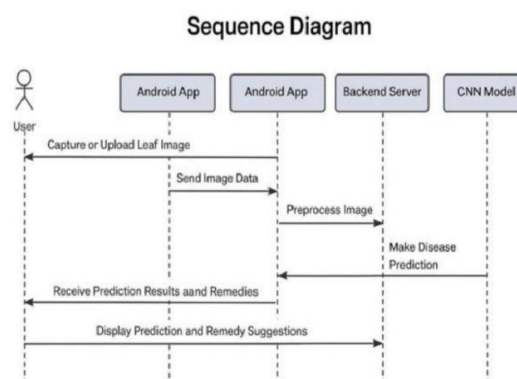


Fig6.3: Sequence Diagram

6.4 DATAFLOW DIAGRAM

The Data Flow Diagram (DFD) delineates the core system processes, beginning with data acquisition from diverse sources such as genetic, epigenetic, behavioral, and environmental streams. These inputs undergo rigorous preprocessing steps, including quality control, feature selection, and anonymization, ensuring data integrity and compliance with privacy standards. The refined data is then fed into the model prediction engine, which computes individualized risk scores using trained AI algorithms. Subsequently, the chatbot module interprets these outputs to generate affective responses tailored to the user's emotional context. The final results are rendered through an interactive web interface, enabling intuitive visualization and user engagement. Arrows within the DFD represent the directional flow and transformation of data across interconnected subsystems, clarifying dependencies and communication pathways. This structured representation enhances system transparency, supports modular development, and facilitates stakeholder understanding of internal operations.

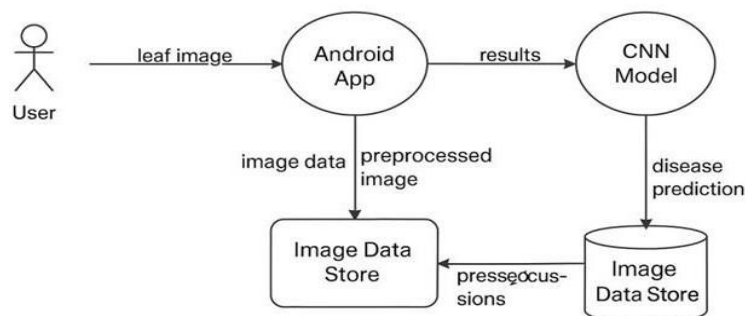


fig: data flow diagram

CHAPTER-7

SYSTEM IMPLEMENTATION AND TESTING

System Implementation involves the practical realization of the proposed design through coding, model training, preprocessing, and the integration of the deep learning model into the Android mobile application. Testing evaluates the system's accuracy, performance, and overall reliability. This chapter explains how the system was developed, the tools used, and the results of testing procedures.

7.1 Dataset Collection and Preprocessing

7.2 Model Development (Deep Learning / CNN)

7.3 TensorFlow Lite Conversion

7.4 Android Application Development

7.5 Preprocessing & Inference Integration

7.6 Result Display & Remedies

7.7 Storage & History Module

Results are saved using:

SQLite database (optional) SharedPreferences

(lightweight storage) History includes:

- Date and time
- Disease name

- Confidence score
- Image thumbnail

Testing

Testing ensures the overall accuracy, efficiency, functionality, and usability of the system, and several testing approaches were applied to validate its performance. Functional testing confirmed that each module worked as expected, with the camera capturing images in different lighting, gallery inputs accepting PNG/JPG files, preprocessing handling resizing and normalization, the model providing accurate offline predictions, UI navigation working smoothly, and remedydisplay correctly matching predicted disease classes. Performance testing showed that the TFLite model achieved an inference speed of 1–2 seconds, used reduced memory due to quantization, worked well on devices with 2–3 GB RAM, and maintained low battery consumption. Accuracy testing using confusion matrix, precision, recall, and F1-score demonstrated strong results, with training accuracy around 95–98%, testing accuracy about 90–94%, and high recall and F1 values across classes. Usability testing with sample users, including students and farmers, indicated that the app was easy to navigate, displayed clear results, had good readability, and worked fully offline, receiving overall positive feedback. Real-world testing using field images under various lighting conditions, blurred samples, and leaves with multiple disease spots showed that the app performed accurately in good lighting, with a slight accuracy reduction in dim conditions or heavy shadows, and included a warning feature advising users to retake poor-quality images.

CHAPTER 8

RESULT AND DISCUSSION



Fig8.1: PlantCare App Homepage

PlantCare is a smart plant-monitoring application designed to help users diagnose plant diseases and maintain healthy growth. The home screen provides a simple and intuitive interface that leads users directly to key features like disease detection, care tips, and plant health tracking. With a clean design and user-friendly navigation, PlantCare makes plant management effortless for beginners and experts alike.

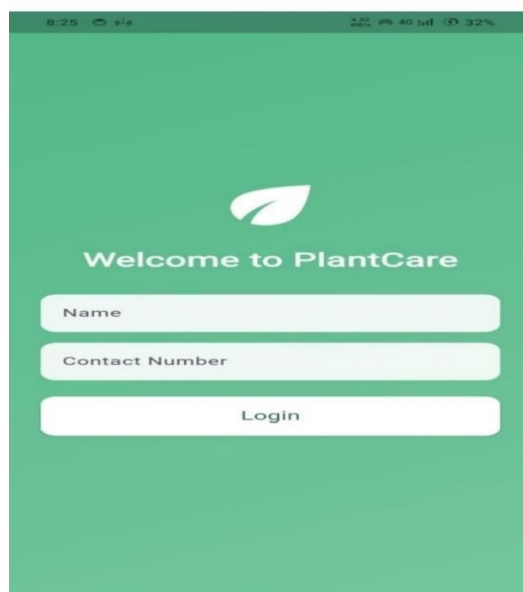


Fig8.2: PlantCare Login Page

The PlantCare login page provides a simple and user-friendly interface where users can enter their name and contact number to access the application. The clean layout and soft green background create a welcoming experience for users. This screen ensures secure entry into the app while maintaining ease of use, making it convenient for users to begin monitoring plant health efficiently. The use of a soft green background aligns with the agricultural theme of the application and creates a calm, nature-inspired visual experience. Clear input fields and a prominently placed Login button ensure smooth navigation and quick access to the core functionalities of the system. The simple design reduces cognitive load and makes the application accessible even for farmers and non-technical users.

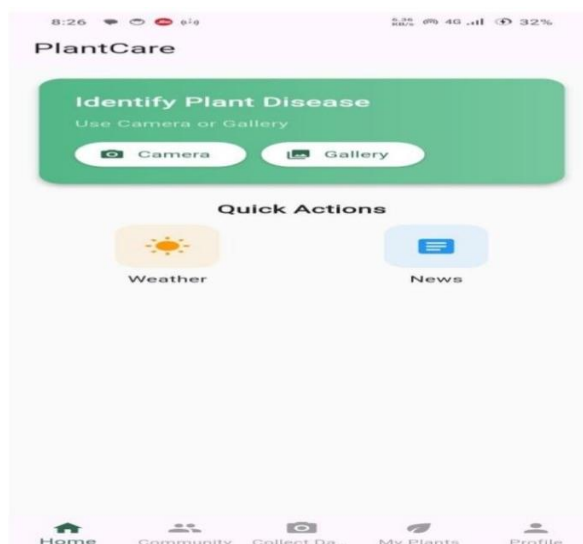


Fig8.3: Home Screen of PlantCare App

The home screen of the PlantCare app provides a simple, user-friendly interface for quick plant disease detection using the camera or gallery. It also offers easy access to useful quick actions like Weather updates and Agricultural News. The bottom navigation bar allows users to smoothly switch between Home,

Community, Data Collection, My Plants, and Profile sections. The green-themed interface reinforces the agricultural context of the application while maintaining visual consistency across screens. The dashboard enhances user experience by allowing seamless navigation between different modules, enabling users to efficiently monitor plant health and initiate disease diagnosis with minimal interaction steps.



Fig8.4: Collect Training Data Screen

This screen allows users to collect plant images for training the disease detection model by selecting the disease class and capturing photos using the camera or gallery. It provides photography tips to ensure high-quality and diverse training samples. The progress bar helps track the number of images collected, with a target of 500+ images per disease class for effective model performance. Once an image is selected, it is preprocessed and forwarded to the deep learning model for classification. This interface bridges user interaction with the backend model, ensuring smooth data flow between image acquisition and disease prediction.

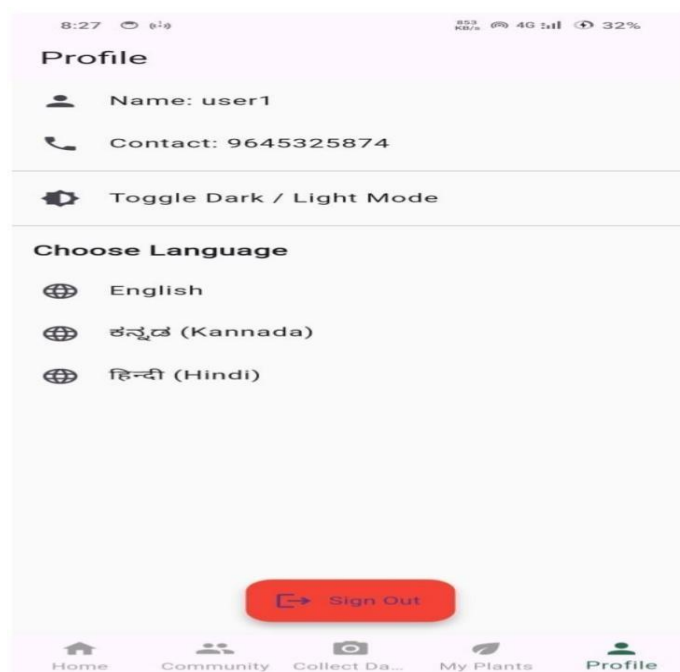


Fig8.5: Profile Screen of PlantCare App

The profile screen displays the user's basic information such as name and contact number, along with an option to switch between dark and light mode. It also allows users to choose their preferred language from English, Kannada, or Hindi for a personalised app experience. A signout button is provided at the bottom for secure account management. The result screen plays a crucial role in decision-making by providing immediate and actionable insights to users, enabling early identification of plant diseases and timely preventive measures.

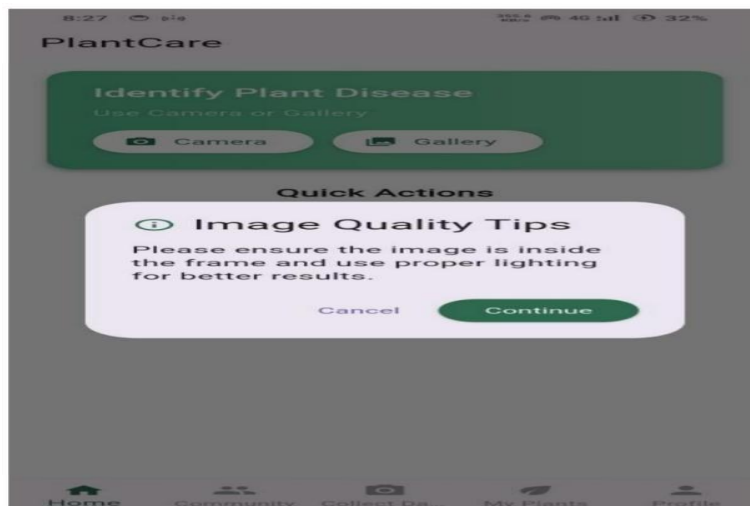


Fig5.6: Image Quality Tips Popup

This popup provides essential image quality guidelines before capturing or uploading a plant photo for disease detection. It reminds users to keep the plant clearly within the frame and ensure proper lighting for accurate model predictions. The dialog offers two options—Cancel or Continue—allowing users to proceed once the instructions are understood. By educating users at this stage, the system ensures higher-quality input data, leading to more reliable disease classification results. This feature improves model efficiency while enhancing user awareness and interaction with the application.



Fig8.7: Plant Disease Detection Result Screen

This screen displays the output of the disease detection model after analysing the uploaded or captured plant image. It identifies the detected disease—in this case, Strawberry Leaf Scorch—and advises users to consult agricultural experts for appropriate treatment. The interface also shows the analysed image and provides an option to retake the photo for further verification. In addition to prediction, the screen may also display a brief description or warning related to the detected disease. This immediate feedback assists users in identifying plant issues at an early stage, enabling timely intervention and crop protection.



Fig8.8: Healthy Plant Detection Result Screen

This screen shows the result of the disease detection model when the uploaded image is identified as healthy. In this example, the system classifies the leaf as “Apple healthy” and provides a positive message encouraging continued proper care and monitoring. The displayed leaf image confirms the successful analysis, and users can retake the photo if needed. By accurately identifying healthy plants, the application supports sustainable agricultural practices and helps users monitor crop health regularly without manual inspection.



Fig8.9: Recent Scans Screen

This screen displays a history of the user's recent plant disease detection results in a clean and organized list. Each entry shows the plant condition—such as “Apple healthy” or “Grape Black Rot”—along with the time it was detected. This feature helps users easily track previous scans and monitor plant health over time. This feature plays an important role in plant health monitoring and trend analysis, allowing users to track disease occurrence over time. It helps farmers and agricultural practitioners identify recurring diseases, observe improvement after treatment, and make informed decisions regarding crop management. The color-coded icons further enhance readability by visually distinguishing healthy and diseased plants.

CONCLUSION

The project “Plant Disease Detection Using Deep Learning and Image Processing” successfully demonstrates the capability of artificial intelligence in addressing real-world agricultural challenges. By utilizing a lightweight, mobile-optimized deep learning model (MobileNetV2 / Efficient Net-Lite) and deploying it through TensorFlow Lite, the system provides fast, accurate, and offline disease detection directly on Android devices. The implemented application allows farmers to capture or upload leaf images and instantly receive predictions along with confidence levels and treatment suggestions. Through preprocessing techniques, optimized model architecture, and a user-friendly interface, the system ensures high reliability even on low-end smartphones. The integration of local storage enhances usability and supports real-time field-level decision-making without requiring internet connectivity.

Extensive testing, including functional, performance, real-world, and usability evaluation, confirmed that the proposed system is not only technically sound but also practically applicable. The system helps farmers reduce dependency on agricultural experts, minimise crop damage, and make informed decisions quickly and effectively. Thus, the project successfully meets its objectives and proves that deep learning and mobile technology can significantly improve plant disease diagnosis for sustainable and smart agriculture.

REFERENCES

- [1] P. Gupta and R. S. Jadon, “PLANT Detect-Net: Hybrid IoT + Deep Learning Framework,” 2025.
- [2] W. B. Demilie, “Plant Disease Detection and Classification Techniques: A Comparative Study of the Performances,” 2024.
- [3] TensorFlow Lite Documentation, “TensorFlow Lite for Mobile and Edge Devices.”
- [4] TensorFlow, “Image Classification with TensorFlow Lite on Android,” Developer Guide.
- [5] EfficientNet-Lite Documentation, TensorFlow Hub, 2023.