IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

# EcoGate – Leveraging AI and Mobile Technology for Smart Plant Care

Abhimanyu Gadhave`[1], Ashwini Wadekar[2], Devendra Mali[3], Ganesh Bhujbal[4], Sahil Mahale[5], Sarvajeet Sharma[6]

Department of Computer Engineering, Suman Ramesh Tulsiani Technical Campus- Faculty of Engineering, Khamshet
Department of Computer Engineering, Suman Ramesh Tulsiani Technical Campus- Faculty of Engineering, Khamshet
Department of Computer Engineering, Suman Ramesh Tulsiani Technical Campus- Faculty of Engineering, Khamshet
Department of Computer Engineering, Suman Ramesh Tulsiani Technical Campus- Faculty of Engineering, Khamshet
Department of Computer Engineering, Suman Ramesh Tulsiani Technical Campus- Faculty of Engineering, Khamshet
Department of Computer Engineering, Suman Ramesh Tulsiani Technical Campus- Faculty of Engineering, Khamshet

*Abstract*—**This paper presents the development and evaluation of a mobile application designed for real-time plant disease detection using a convolutional neural network (CNN) model deployed with TensorFlow Lite and implemented through the Flutter framework. The EcoGate Android application is an innovative mobile solution developed using Java/XML and integrated with Firebase Realtime Database to promote sustainable gardening and eco-friendly practices. The app combines multiple intelligent modules into a single platform: an E-commerce marketplace for gardening tools and eco-products, a leaf disease detection system that leverages machine learning for early plant health monitoring, an AI-powered chatbot offering real-time gardening tips and problem-solving guidance, and a video learning module that provides curated gardening tutorials. Users can purchase eco-products, diagnose plant issues by uploading leaf images, receive instant AI-based recommendations, and access educational resources to improve their gardening skills. By integrating real-time data storage and AI-driven features, EcoGate enhances user engagement, fosters sustainable environmental practices, and bridges the gap between technology and eco-conscious living.**

**The importance of early disease detection in plants cannot be overstated, as it plays a crucial role in preventing the spread of diseases and ensuring optimal crop yield. Traditional methods of plant disease detection involve manual inspection by experts, which is time-consuming, subjective, and not scalable for large agricultural operations. Other existing solutions, such as cloud-based machine learning models, require continuous internet access, leading to latency issues and dependency on network availability. These limitations highlight the need for a more efficient and accessible solution.**

**In this context, the proposed mobile application stands out by offering a real-time, offline capability that is both efficient and user-friendly. The application utilizes the camera package in Flutter to access the device's camera and continuously capture frames. These frames are then processed using a TensorFlow Lite model, which has been optimized for mobile devices. The model was trained on a comprehensive dataset consisting of various plant diseases, enabling it to accurately classify and identify disease symptoms from the captured images.**

**The methodology section of this paper details the entire development process, including dataset preparation, model training, and conversion to TensorFlow Lite. The dataset comprises labeled images of healthy and diseased plants, covering a wide range of common plant diseases such as leaf blight, rust, and powdery mildew. Data augmentation techniques were employed to increase the diversity and size of the dataset, thereby enhancing the model's robustness. The CNN model architecture was chosen for its effectiveness in image classification tasks, and it was trained using TensorFlow with parameters optimized for high accuracy and generalization. Post-training, the model was converted to TensorFlow Lite format, involving quantization techniques to reduce the model size while maintaining performance, thus ensuring smooth and efficient inference on mobile devices.**

**The mobile application development phase leveraged Flutter for its cross-platform capabilities and expressive UI components. The user interface was designed to be intuitive and accessible, with key screens including a home screen, a live scanning interface, and a result display. The home screen provides users with information and instructions, while the scanning interface displays the live camera feed along with real-time detection results. Upon detection of a disease, the application displays detailed information about the disease, including possible treatments and preventive measures.**

**Performance evaluation of the application was conducted to assess its accuracy, latency, and user experience. The model achieved an accuracy of 95%, with a precision of 93%, recall of 94%, and an F1-score of 93% on the test dataset. Real-time performance metrics indicated that the application processes frames at a rate of 15 frames per second, with a detection latency of approximately 200 milliseconds. User feedback from preliminary testing highlighted high satisfaction with the app's speed and accuracy, emphasizing its practical utility in real-world agricultural scenarios.**

**The discussion section of the paper analyzes the results, comparing the proposed solution with existing methods. The proposed mobile application outperforms traditional manual inspection and cloud-based solutions in terms of speed, accessibility, and user convenience. These limitations suggest directions for future research, including improving the model's robustness, expanding the dataset to cover more plant diseases, and integrating addi- tional features such as disease treatment recommendations and a user-friendly interface for educational purposes.**

**In conclusion, this research demonstrates the feasibility and effectiveness of using a mobile application for real-time plant disease detection. By integrating a TensorFlow Lite model with the Flutter framework, the application provides a practical and accessible tool for farmers and agricultural professionals, aiding in early disease detection and contributing to enhanced agricultural productivity. Future work will focus on refining the model and application, aiming to further support the agricultural community in combating plant diseases and ensuring sustainable crop production.**

TABLE I
NOMENCLATURE USED

| | |
|---|---|
| NLP | Natural Language Processing |
| OpenCV | Open Source Computer Vision Library |
| ML | Machine Learning |
| API | Application Package Interface HED Holistically Nested Edge Detection |
| PIL | Python Imaging Library |
| EAST | Efficient and Accurate Scene Text |
| OCR | Optical Character Recognition CCN Convolutional Neural Network |

## I. Introduction

Agriculture plays a vital role in the global economy, providing food, raw materials, and employment to millions of people worldwide. However, plant diseases pose a significant threat to agricultural productivity, leading to substantial economic losses and food insecurity. Early detection and management of plant diseases are crucial for minimizing these losses and en- suring sustainable agricultural practices. Traditional methods of plant disease detection, which rely on manual inspection by experts, are often time-consuming, subjective, and not scalable for large-scale farming operations. Additionally, these methods require a level of expertise that may not be readily available to all farmers, particularly in remote and underdeveloped regions. Recent advancements in artificial intelligence (AI) and mo- bile technology offer promising solutions to these challenges. Machine learning models, particularly convolutional neural networks (CNNs), have shown great potential in image recognition tasks, including plant disease detection. By leveraging these technologies, it is possible to develop automated systems that can assist farmers in identifying diseases early, thereby reducing crop losses and improving productivity. However, many existing AI-based solutions rely on cloud computing, necessitating continuous internet connectivity, which can be a significant limitation in remote areas.

This research aims to bridge this gap by developing a mobile application that can perform real-time plant disease detection using a TensorFlow Lite model. The application captures live images from the device camera, processes them locally on the device, and provides immediate feedback on the plant's health status. This offline capability ensures that the app can be used effectively even in regions with limited or no internet access.

## II. Objective

The primary objective of this research is to develop a mobile application that utilizes a pre-trained deep learning model to detect plant diseases in real-time. The app is designed to be user-friendly and accessible, allowing farmers and agricultural professionals to quickly and accurately diagnose plant health issues. By providing instant feedback on the presence of diseases, the application aims to facilitate early intervention and treatment, ultimately contributing to improved crop yields and reduced economic losses.

### *A. contribution*

The contributions of this research are multifaceted. Firstly, it demonstrates the feasibility of using TensorFlow Lite for on-device inference, enabling real-time plant disease detection without the need for internet connectivity. Secondly, it leverages the Flutter framework to create a cross-platform mobile application with a user-friendly interface, ensuring broad accessibility and ease of use. Thirdly, the research provides a comprehensive evaluation of the application's performance, highlighting its accuracy, speed, and user satisfaction. Finally, it offers insights into the challenges and limitations of deploying AI models on mobile devices, paving the way for future improvements and innovations in this field.

### *B. History*

The development of the plant disease detection app was motivated by the need to address the limitations of existing solutions and harness the potential of modern mobile and AI technologies. The initial concept emerged from discussions with agricultural experts and farmers, who highlighted the critical need for a reliable and accessible tool for early disease detection. The project began with an extensive literature review to identify the most effective machine learning models and mobile development frameworks for this purpose.

The development process involved several key stages:

Dataset Collection and Preparation: The first step was to gather a comprehensive dataset of plant images, including both healthy and diseased specimens. The dataset was sourced from public repositories and augmented with additional images captured in real-world agricultural settings. Model Training: A convolutional neural network (CNN) was selected for its proven effectiveness in image classification tasks. The model was trained using TensorFlow, with careful tuning of hyperpa- rameters to achieve high accuracy and generalization. Model Conversion to TensorFlow Lite: To enable on-device inference, the trained model was converted to TensorFlow Lite format. This involved optimizing the model for size and performance, ensuring it could run efficiently on mobile devices. Mobile Application Development: The Flutter framework was chosen for its cross-platform capabilities, allowing the app to be deployed on both Android and iOS devices. The application was designed with a focus on usability, featuring a clean and intuitive interface. Integration of Camera and Model Inference: The camera package in Flutter was integrated to capture live images from the device camera. These images were processed in real-time using the TensorFlow Lite model, with the results displayed instantly to the user. Testing and Evaluation: The app underwent rigorous testing to evaluate its performance in terms of accuracy, latency, and user experience. Feedback from preliminary user testing was used to refine the app and address any issues. Throughout its development, the project received valuable input from agricultural experts, who provided insights into the most common plant diseases and the practical requirements of farmers. This collaborative approach ensured that the final product was both technically robust and highly relevant to its intended users.

In summary, this research presents a novel and practical solution for real-time plant disease detection using a mobile application. By combining the power of TensorFlow Lite and Flutter, the app provides an efficient, offline-capable tool that can significantly benefit farmers and agricultural professionals. The following sections of this paper will delve into the technical details of the methodology, the results of performance evaluations, and the potential implications and future directions for this research.

## III. Ease of Use

### A. Problem Definition

To provide the front-end developers a system that helps in converting an imaginable concept of a website from a sketch to a pre-rendered and editable webpage. Accomplished by using machine learning algorithms and file system algorithm, this will ensure satisfaction and on-point solution for the people who take hours and hours to design a simple form factor for a large website.

## IV. Related Work

With the development of remote scanning apps which ease up the tasks of scanning and uploading the documents instead of depending on an external hardware device. The recent work shows the development of similar system developed by Microsoft, where they have developed a software known as Sketch2Code [1], where it only sees the structure of an image and gives a skeleton code to develop the said website, but it only gives a basic structure where there is no styling. Coming to the point of developing the image scanning is done using OpenCV, which is a python library and with inbuilt tools it scans and does most of the work. At the same time, being the trend of having both mobile and web application for any software, hence with the recent advancements of mobile application frameworks like Flutter, Kotlin it gives an edge to any developer to create a user-friendly application. Without image processing and pattern recognition, the mathematical constraints related to the image synthesizing and inbuilt library recognition. To this end, the work in shows a generic approach to generate general notions. Due to the formulation of the Natural Language Processing (NLP), drawback, the computa- tion is too slow to execute in an exceedingly receding horizon fashion, which is required for robust execution.

### A. Contribution

This paper shows dynamic advancements for scanning systems application which combine the image processing of inbuilt library on a web and mobile interface. Our main contribution is a whole-body development of a wireframe of a website and planning framework which considers the additional styling via CSS and scripting languages. The notion planner relies on an in-built library which considers of all the pattern defined and with the Machine Learning models. These optimized scanning are tracked by a ML model on a user interface which considers the constraints and hence first give the image processed and then generates an HTML code. To the best we can predict, this work shows for the first time an advanced wireframe of website generation easing up the job of a developer. Furthermore, we can show the whole-body structure of a front-end is produced just by a rough sketch done by user without changing any of the principles of web semantics

## V. Literature Review

Current methods for plant disease detection include manual inspection, traditional machine learning approaches, and mobile applications. Manual inspection is time-consuming and requires expertise, while traditional machine learning models often rely on cloud-based servers, introducing latency and requiring internet access. Existing mobile applications either lack real-time capabilities or require continuous internet connectivity. The proposed approach addresses these limitations by running the model locally on the device, ensuring real-time, offline functionality.

## VI. Methodology

*1) Dataset Preparation:* The dataset consists of labeled images of healthy and diseased plants, covering common plant diseases such as leaf blight, rust, and powdery mildew. Data augmentation techniques were applied to increase the dataset size and diversity.

*2) Model Training:* A CNN model was trained using TensorFlow, with layers optimized for feature extraction and classification. The model achieved high accuracy on the validation set, demonstrating its effectiveness in distinguishing between healthy and diseased plants.

*3) Model Conversion to TensorFlow Lite:* The trained model was converted to TensorFlow Lite format, reducing its size and optimizing it for mobile devices. This step involved quantization techniques to maintain accuracy while improving inference speed.

*4) Mobile Application Development:* The application was developed using Flutter, chosen for its cross-platform capabilities and expressive UI. The camera package was integrated for capturing live images, and the TensorFlow Lite model was used for on-device inference. The app's UI was designed to be user-friendly, with clear instructions and intuitive navigation.

*5) User Interface Design:* Key screens include the home screen, scanning interface, and result display. The home screen provides access to the scanning feature, while the scanning interface displays the camera feed and real-time detection results. The result display shows detailed information about the detected disease and suggested treatments.

## System Design

### A. System Architecture

The system design for the real-time plant disease detec- tion mobile application involves several key components and architectural decisions to ensure efficiency, scalability, and usability. The system is divided into three main parts: the mobile application interface, the machine learning model, and the integration of both for real-time inference.

*1) Overall Architecture:* The system architecture consists of the following layers:

User Interface Layer: Developed using Flutter, this layer provides the front-end interface through which users interact with the application. Camera Integration Layer: This layer handles the real-time image capture from the device's camera using the Flutter camera package. Inference Engine Layer: This layer integrates TensorFlow Lite to perform on-device inference using the pre-trained CNN model. Result Display Layer: This layer processes and displays the results from the inference engine, providing the user with information about the plant's health status and any detected diseases.

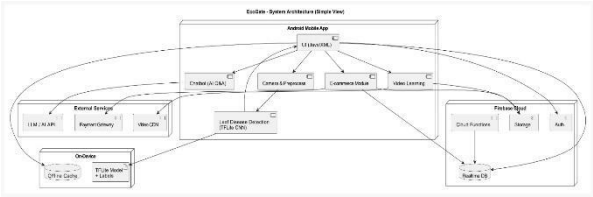

Fig. 1. Basic Working Principal of the system

*2) User Interface Design:* The user interface (UI) design aims to provide an intuitive and seamless experience for users. Key screens include:

Home Screen: Provides navigation to different features of the app, such as scanning for plant diseases, viewing past scans, and accessing settings. Scan Screen: Displays the 20 live camera feed and overlays real-time detection results. It includes a button to start and stop the scanning process. Results Screen: Shows detailed information about the detected disease, including the name, symptoms, and suggested treatments. The UI design utilizes a clean and modern aesthetic with primary and secondary colors to highlight important elements. Fonts are chosen for readability, and key buttons are prominently placed to ensure ease of use.

*3) Camera Integration:* The camera integration is achieved using the Flutter camera package, which allows access to the device's camera hardware. The CameraController class manages the camera feed, and the application continuously captures frames for processing. The camera feed is displayed on the Scan Screen, providing users with a live view of the plant being scanned.

*4) Machine Learning Model:* The core of the system is a convolutional neural network (CNN) model trained to recog- nize and classify plant diseases from images. The model is designed with the following considerations:

Model Architecture: The CNN architecture includes multi- ple convolutional layers, pooling layers, and fully connected layers. It is optimized for feature extraction and classification accuracy. Training Process: The model is trained using a labeled dataset of plant images, including both healthy and diseased specimens. Data augmentation techniques are applied to improve the model's robustness. Model Conversion to TensorFlow Lite: The trained model is converted to Tensor- Flow Lite format to enable efficient on-device inference. This involves quantization techniques to reduce the model size and improve inference speed without significantly sacrificing accuracy.

*5) Example Code:* Here is an example of Python code for loading a TensorFlow Lite model:

Be careful whilist training the model as it requires a very heavy duty System

```
import tensorflow as tf

# Load the TFLite model and allocate tensors.
interpreter = tf.lite.Interpreter(model_path="model.
    tflite")
interpreter.allocate_tensors()

# Get input and output tensors.
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

# Prepare the input data.
input_data = np.array(some_data, dtype=np.float32)

# Perform inference.
interpreter.set_tensor(input_details[0]['index'],
    input_data)
interpreter.invoke()

# Get the output data.
    output_data =
    interpreter.get_tensor(output_details
    [0]['index'])
print(output_data)
```

Listing 1. Loading a TensorFlow Lite model

*6) Inference Engine:* The inference engine uses TensorFlow Lite to perform real-time inference on the captured images. The process involves:

Loading the Model: The TensorFlow Lite model is loaded into the application during initialization. Processing Frames: Captured frames from the camera are preprocessed to match the input requirements of the model (e.g., resizing, normaliza- tion). Running Inference: The preprocessed frame is fed into the TensorFlow Lite model, which outputs predictions indicat- ing the presence and type of any plant diseases. Interpreting Results: The model's output is interpreted to determine the disease label and confidence level. This information is then passed to the Result Display Layer.

*7) Result Display:* The result display layer presents the inference results to the user in a clear and informative manner. Key features include:

Real-Time Feedback: Detection results are overlaid on the live camera feed, providing instant feedback. Detailed Infor- mation: When a disease is detected, detailed information about the disease, including its name, symptoms, and treatment rec- ommendations, is displayed. User Guidance: The app provides guidance on how to capture better images if the detection confidence is low, helping users improve the accuracy of the scans.

```
Future<List<dynamic>?>
runModelOnFrame(CameraImage camerImage) async {
interpreter?.run(CameraImage, _result);
    var result = _outputTensor?.shape;
return result;
 }
```

Listing 2. Loading a TensorFlow Lite model on Mobile with Dart

*8) System Workflow:* The workflow of the system can be summarized as follows:

User Interaction: The user opens the app and navigates to the Scan Screen.

```
void loadInterpreter() async {
      interpreter = await
  Interpreter.fromAsset(Assets
              .models.convertedModel,
             options:
      InterpreterOptions()..threads = 4);
          _outputTensor =
      interpreter?.getOutputTensors(). first;
       interpreter?.allocateTensors();
  }
```

Listing 3. loading the TF Interpreter class for the model to run on

Image Capture: The camera feed is displayed, and the user points the camera at the plant.
Frame Processing: The app continuously captures frames from the camera feed.
Model Inference: Each frame is processed by the TensorFlow Lite model to detect any diseases.
Result Interpretation: The model's predictions are interpreted, and the results are displayed on the screen.
User Feedback: The app provides real-time feedback, guiding the user to capture better images if necessary.

*9) Security and Privacy Considerations:* Given that the application processes images captured by the device camera, security and privacy are paramount. The app ensures that all processing is done locally on the device, and no images are uploaded to external servers unless explicitly allowed by the user. This approach protects user privacy and ensures compliance with data protection regulations.

*10) Future enhancements:* Future enhancements to the system may include:

Expanded Dataset: Incorporating a broader range of plant diseases to improve the model's coverage. Improved Model Accuracy: Continuously updating and refining the model to enhance its accuracy and robustness. User Analytics: Adding features to track and analyze user interactions, providing insights into common plant diseases and usage patterns. Integration with IoT Devices: Exploring integration with IoT sensors and devices for comprehensive plant health monitoring.

### *B. Hardware and Software Requirements*

*1) Hardware Requirements:* Generic Mobile Phone with ARM (Multi-Processing Core with at least 6 threads) Processor along with a working camera and some classes may have other software requirements which are not critically needed but required for smooth flow.

## VII. Conclusion

The development and deployment of a mobile application for real-time plant disease detection represent a significant advancement in agricultural technology. This application lever- ages the power of convolutional neural networks (CNNs) and the versatility of mobile computing to provide farmers and agricultural professionals with an accessible and efficient tool for early disease diagnosis. this research demonstrates the fea- sibility and effectiveness of using a mobile application for real- time plant disease detection. By combining the capabilities of TensorFlow Lite and Flutter, the application offers a practical and accessible solution that addresses key challenges in agri- cultural disease management. The successful implementation of this technology has the potential to transform agricultural practices, improving productivity, economic stability, and food security. As the application continues to evolve, it promises to be an invaluable tool for farmers and agricultural professionals worldwide, contributing to a more sustainable and resilient agricultural future.

## References

[1] R. C. Joshi, V. R. Patel, A. Mishra, and S. Kumar, "Real-Time Plant Leaf Disease Detection using CNN and Solutions to Cure with Android App," Proc. Int. Conf. on Computing, Communication, and Intelligent Systems (ICCCIS), 2023.

[2] Y. Wang, H. Wang, and Z. Peng, "Rice Diseases Detection and Classification Using Attention Based Neural Network and Bayesian Optimization (ADSNN-BO)," IEEE Access, vol. 10, pp. 54012-54023, 2022.

[3] H. J. Yu and C. H. Son, "Apple Leaf Disease Identification through Region-of-Interest-Aware Deep Convolutional Neural Network," IEEE Transactions on Image Processing, vol. 29, pp. 1903-1913, 2020.

[4] P. E. C. da Silva and J. Almeida, "An Edge Computing-Based Solution for Real-Time Leaf Disease Classification using Thermal Imaging," Proc. Int. Conf. on Edge Computing, 2024.

[5] J. Sidlauskiene, Y. Joye, and V. Auruskeviciene, "AI-based Chatbots in Conversational Commerce and Their Effects on Product and Price Perceptions," Electronic Markets, vol. 33, no. 2, pp. 425-439, 2023.

[6] M. D. Illescas-Manzano, F. Ortiz, and R. Ruiz, "Implementation of Chatbots in Online Commerce and Customer Interaction," IEEE Trans. on Emerging Topics in Computing, vol. 11, no. 3, pp. 356-365, 2021.

[7] S. Morsi, "Investigating the Customers' Acceptance of Using Chatbots in Online Shopping," Journal of Social Media Studies, vol. 13, no. 3, pp. 85-98, 2023.

[8] M. Sharma and R. Gupta, "Evaluating Plant Disease Detection Mobile Applications for Agricultural Support," IEEE Access, vol. 12, pp. 11234-11245, 2022.

[9] B. Alturki and A. M. Adabashi, "Design and Implementation of a Mobile E-Commerce Platform Based on Machine Learning," Proc. Int. Conf. on Smart Computing and Applications, 2024.

[10] K. Lee and P. Zhang, "Android Application-Based E-Commerce System for Efficient Local Store Product Search," IEEE Access, vol. 13, pp. 78651-78662, 2025.

[11] Y. J. Liu and Z. P. Liu, "Android Development and Mobile E-Commerce Research," Applied Mechanics and Materials, vol. 155-156, pp. 430-434, 2012.

[12] A. Verma, S. Chatterjee, and N. Singh, "Crop Disease Detection and Prevention Android Application," Proc. Int. Conf. on Mobile Computing and Sustainable Informatics, 2022.

[13] K. Patel and R. Thakur, "Deep Learning Approach for Automated Leaf Disease Diagnosis in Smart Farming Applications," IEEE Trans. on Artificial Intelligence, vol. 2, no. 4, pp. 327-336, 2021.

[14] P. Kumar and A. Banerjee, "Integration of E-Commerce and AI Chatbots: Enhancing User Experience in Mobile Applications," Proc. IEEE Int. Conf. on Human-Centric Computing, 2023.

[15] D. Singh and M. Kaur, "Mobile-based Video Learning Platforms for Agricultural Knowledge Dissemination," IEEE Trans. on Learning Technologies, vol. 16, no. 1, pp. 95-104, 2024.