

Planterbox: Smart Plant Identification and AI-Powered Care Guidance

Carlos Rodriguez, MD Omit, Kenny Zhu
Professor Yunhua Zhao
Department of Computer Science
City College Grove School of Engineering
December 23, 2025

Personal Contribution

Carlos Rodriguez: Project Lead and primary integrator for Planterbox. Responsible for application development through Android Studio and guiding project workflow. (40%)

Kenny Zhu: System Savvy, responsible for the design of the care recommendations and documentation of the project. (35%)

MD Omit: Techsmith, responsible for the frontend and backend design of the application. (25%)

Abstract

Plant identification and care remain common pain points for hobbyists and new plant owners, especially when symptoms are subtle or when users encounter unfamiliar species. Planterbox addresses this gap with a mobile-first workflow that combines on-device image classification with cloud-based large language model (LLM) guidance. The application allows users to capture or upload a plant photo, classify the plant using a custom TensorFlow Lite model, and retrieve AI-generated care and health recommendations from a hosted backend.

Our approach separates responsibilities to balance latency, privacy, and reliability. The on-device classifier provides immediate species predictions without requiring internet access. At the same time, the backend uses a curated knowledge base (species care reference) and an LLM to generate readable, structured guidance. This design supports a consistent user experience across devices while enabling rapid iteration on the AI guidance layer without shipping new mobile builds.

Key outcomes include a functional Android prototype developed in Android Studio with a modern Jetpack Compose interface, a working inference pipeline utilizing a custom MobileNetV3-based model exported to TensorFlow Lite (TFLite), and a deployed Flask backend that exposes health and image-processing endpoints. In demonstration tests, the app successfully performs on-device classification and returns LLM-generated care tips when internet connectivity is available. The project demonstrates an end-to-end AI product pipeline (data → model training → mobile deployment → backend/LLM integration) and highlights practical constraints such as hosting cold-start latency, API quotas, and mobile network variability. Planterbox contributes a reference architecture and implementation blueprint for lightweight mobile AI applications that require both offline inference and contextual, human-readable recommendations. Future work includes expanding the labeled dataset to additional species and diseases, improving robustness under varied lighting conditions, and adding on-device plant health detection models to reduce dependency on cloud services.

Introduction

Plants play a critical role in agriculture, home gardening, and environmental sustainability. Correctly identifying a plant and maintaining its health are essential to its longevity. Early detection of unhealthy conditions helps prevent crop loss and improves outcomes for home growers. However, recognizing early signs of plant stress, disease, or poor growing conditions often requires specialized knowledge and experience. In many cases, visible symptoms such as discoloration, wilting, or abnormal leaf patterns can be subtle and difficult for the average person to interpret correctly.

The motivation for this project comes from the desire to build a free, user-friendly plant identification and health assessment system that is practical and accessible. With the rapid growth of mobile devices and advancements in computer vision, image-based plant analysis has become a promising way to bridge this knowledge gap. By leveraging machine learning models that learn visual patterns from plant images, users can access monitoring tools that support better care decisions. Such tools are valuable for hobby gardeners and for users who receive plants as gifts but lack access to experts or professional diagnostic resources.

The primary objective of this project is to design and implement a system capable of identifying plant species, assessing a plant's health, and providing care recommendations from a single image. The project aims to develop a mobile application that uses a lightweight identification model to determine the species of a plant in a given image. It also integrates a language model to provide tailored care recommendations based on the same image. Environmental sensor data (such as soil moisture or temperature) and detailed disease identification are outside the scope of this project.

This project resulted in a mobile application that delivers a complete pipeline connecting machine learning with a real-world user workflow. We implemented the GreenThumbV1 model, based on MobileNetV3, and fine-tuned it from a baseline MobileNetV3 model to improve plant identification performance. In addition, we integrated Google's Gemini 2.5 Flash language model to generate care recommendations. A plant image and a tailored prompt are sent to the language model, and the generated output is displayed in the application. Overall, this project demonstrates that efficient deep learning models can provide meaningful plant-care insights while remaining suitable for practical mobile use.

Literature Review/ Background

Several CNN architectures have been widely explored for plant identification and disease classification. Models such as AlexNet, VGG16, ResNet101, EfficientNet, InceptionV3, and MobileNet have been applied to classification tasks using leaf images. Prior work has also investigated multi-prediction approaches, as seen in the study "Deep Learning for Plant Identification and Disease Classification from Leaf Images: Multi-prediction Approaches." That work reported InceptionV3 as a strong backbone among the evaluated CNNs and suggested that using a single model for both identification and disease-related tasks can be more effective than using separate models. A unified model can simplify model selection, reduce memory usage, and improve efficiency while maintaining strong performance.

Recent research has also explored hybrid architectures that combine Vision Transformers (ViTs) with convolutional feature extraction. Studies such as Plant Disease Detection Using a Hybrid Approach Based on Vision Neural Network Transformers and Enhancing Plant Leaf Disease Identification with a CNN and DenseNet Hybrid Model report that hybrid designs can achieve high accuracy with fewer trainable parameters, improving efficiency. These results suggest that alternative deep learning approaches can be robust in distinguishing between disease categories while supporting accurate plant identification.

In addition to research efforts, several commercial applications provide functionality similar to Planterbox. PictureThis is a plant identification app that uses AI-based image recognition to identify plants, flowers, and trees. It also offers features such as disease diagnosis, care tips, and toxic plant identification. Planta is a plant care assistant that provides tailored schedules and reminders (e.g., watering and fertilizing) and includes plant identification and basic treatment guidance. PlantIn is another plant care application that combines identification, care guides, and community support, with features such as articles and quizzes. While these applications have strong feature sets, a shared limitation is pricing: all three require ongoing subscription payments.

Most existing plant disease classification systems focus on multi-class disease identification across numerous plant species. Although this approach can be informative, it may be more detailed than what many everyday users need, especially users who primarily want to know what plant they have and whether it appears healthy. In addition, many systems are evaluated primarily on curated datasets and are not integrated into user-facing applications. In

contrast, this project prioritizes deployability and usability on mobile devices rather than highly detailed disease classification. By framing health assessment as a simpler classification task, the system reduces complexity and improves practical robustness, aligning with the project’s goal of supporting real-world usage scenarios.

Overall, existing work highlights a gap between high-performing research models and deployable systems suitable for real-world use. Many studies emphasize accuracy improvements without fully addressing integration constraints, efficiency requirements, or user interaction design. Additionally, much prior research assumes access to large, balanced datasets and powerful hardware. Planterbox addresses these gaps by emphasizing lightweight model design using MobileNetV3, practical health assessment framing, and integration into a functional mobile application. By focusing on deployability and usability, this project complements existing research and demonstrates how deep learning techniques can be translated into a real-world plant care tool.

Methods and System Design

The Planterbox project implements a full end-to-end pipeline for image-based plant health classification and care tip generation. The methodology follows these major steps: (1) image acquisition and upload through a mobile interface, (2) on-device model inference using TensorFlow Lite for image classification, (3) an optional backend call to a large language model to generate care tips, and (4) user feedback display that includes both the classification result and the generated care text. This pipeline balances efficient on-device inference with the option to use more powerful backend processing when internet connectivity is available.

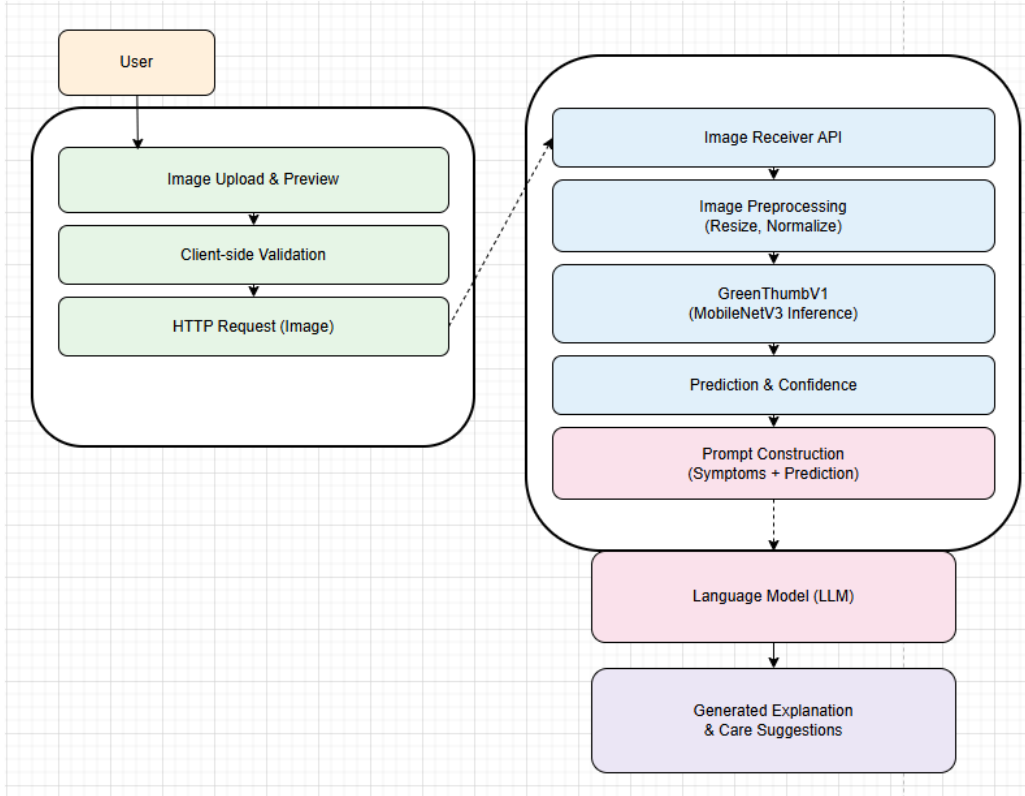


Figure 1. Full end-to-end Planterbox pipeline.

The system is organized into two major components: the mobile application (frontend) and the backend server responsible for inference and care guidance, as shown in Figure 1. The mobile application is a native Android app written primarily in Kotlin. Its responsibilities include providing a user interface for capturing and uploading plant images, preprocessing images for local inference, triggering on-device classification, and displaying results. When internet connectivity is available, the app also coordinates REST API calls to backend services to retrieve care tips. This design supports both offline classification (pure on-device inference) and online features (backend-generated care guidance), ensuring the core identification functionality remains available without network access.

The backend server was implemented in Python and is responsible for receiving image data from the mobile app when the network is available. It calls the large language model (LLM) through an API request to generate care recommendations based on the image, then returns the generated text and relevant metadata to the app for display. This separation keeps the on-device

model lightweight while offloading heavier processes, such as natural language generation, to the backend, improving responsiveness while still providing useful guidance.

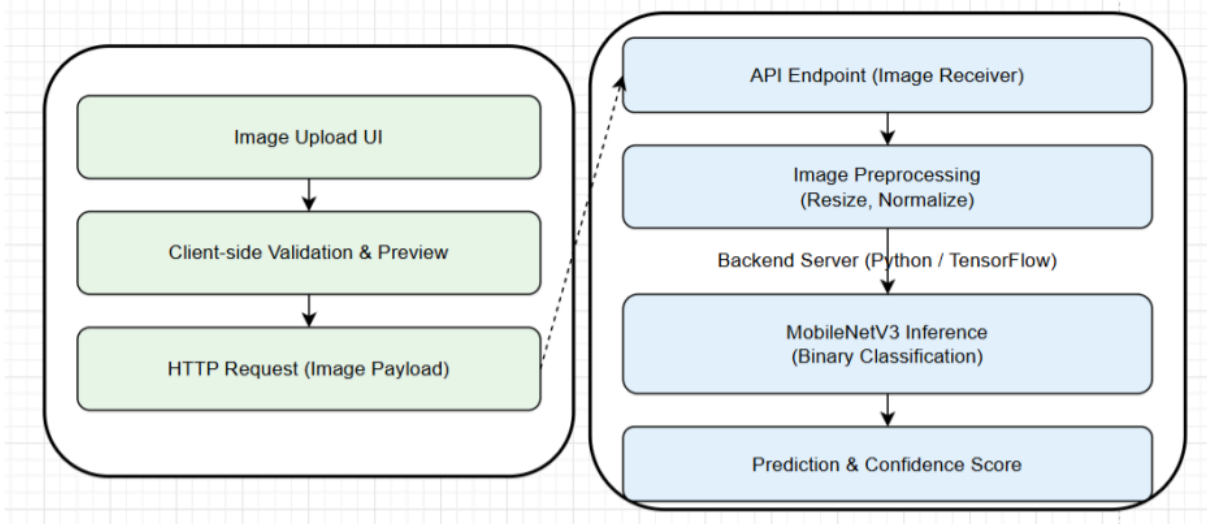


Figure 2. End-to-end GreenThumbV1 system pipeline for plant health classification.

As illustrated in Figure 2, the GreenThumbV1 system is divided into multiple layers that visualize the data flow from image acquisition to classification output. The structure highlights a separation between interface logic and machine learning computation, which improves maintainability and scalability. User interaction is the first stage: the user uploads a plant image through the application interface. The system is designed to accept common image formats, whether captured by a phone camera or selected from an existing source.

Next, the frontend provides an image upload interface and performs basic client-side validation, including checking file type and file size and displaying a preview of the uploaded image. These checks improve usability and reduce unnecessary backend requests. The image is then packaged and transmitted to the backend inference service. Upon receipt, the backend preprocesses the image to match the model's input requirements, including resizing to the expected resolution and normalizing pixel values. The preprocessed image is then passed to the MobileNetV3-based classifier, which performs inference and outputs a probability score for the predicted plant class. Finally, the raw model output is converted into a human-readable prediction with an associated confidence score.

How Our AI Identifies a Plant

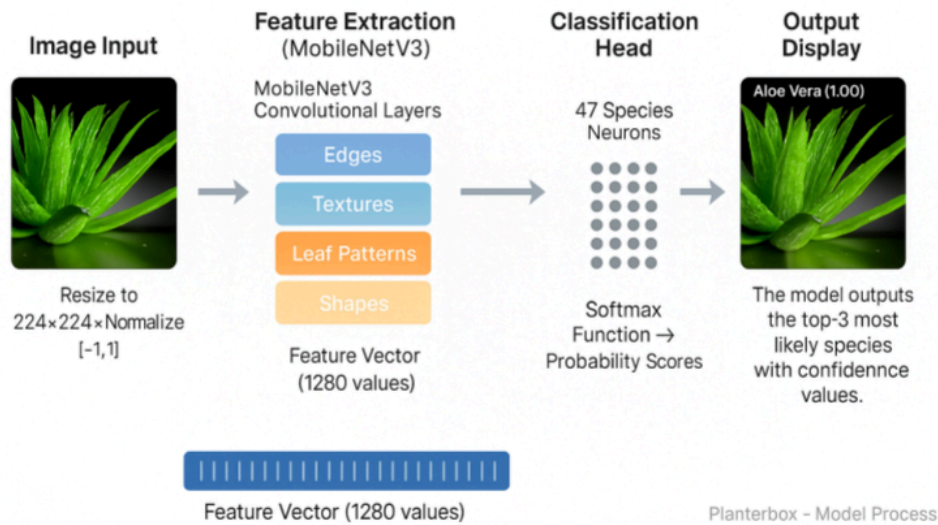


Figure 3. A deeper look at how GreenThumbV1 internal breakdown.

While Figure 2 illustrates the end-to-end system pipeline, Figure 3 provides a conceptual view of the internal feature extraction and classification process performed by the MobileNetV3 model. This figure highlights how low-level visual cues, such as edges and textures, are progressively transformed into higher-level representations used for plant species prediction. At the core of the system is a MobileNetV3 convolutional neural network, selected for its balance between accuracy and computational efficiency. MobileNetV3 uses depthwise separable convolutions, inverted residual blocks, and lightweight attention mechanisms, making it well-suited for deployment in resource-constrained environments such as mobile phones. The model is initialized with pretrained ImageNet weights and fine-tuned on a plant image dataset. This choice aligns with the project's design goals of real-time plant identification and deployability while maintaining strong performance for practical use.

```

You are a plant diagnosis and care assistant.

The user provides:
1. An image of a plant
2. The species name: "{species_name}"

Your job is to produce ONE FINAL JSON OUTPUT with:

{{
  "validated_species": "",
  "visible_symptoms": [],
  "likely_causes": [],
  "care_recommendations": "",
  "urgency_level": "low | medium | high"
}}

CARE RECOMMENDATIONS MUST INCLUDE:

1. Growth rate (slow, moderate, fast)
2. Soil type (well-drained, sandy, loamy, acidic)
3. Sunlight requirement (full sunlight, indirect sunlight, partial sunlight)
4. Watering frequency (water weekly, keep soil evenly moist, let soil dry between watering, keep soil slightly moist, keep soil consistently moist, water when soil is dry, water when topsoil is dry, water when soil feels dry)
5. Fertilization type (balanced, acidic, low-nitrogen, organic, no)

Additional instructions:
- Analyze the plant image to identify symptoms.
- Tailor recommendations to both species and symptoms.
- Make all recommendations accurate, and concise.
- Keep the information short (a couple of sentences).
- Output ONLY the JSON. No markdown or commentary.

```

Figure 4. Prompt that the language model is given.

In addition to plant identification, the image is also sent to a large language model (LLM) through an API request. The LLM used in this project is Google's Gemini 2.5 Flash, and the prompt provided to the model is shown in Figure 4. The prompt instructs the model to analyze the plant image, identify visible symptoms, and provide tailored recommendations based on the predicted species and observed condition. The generated guidance includes items such as growth rate, soil type, sunlight requirements, watering frequency, and fertilization recommendations. The model returns this information in a JSON format, which is then parsed and displayed in the mobile application interface for the user.

Mobile Application Design and Implementation

The Planterbox system is implemented as a mobile application designed to allow users to capture or upload plant images directly from their devices. A mobile approach was selected to reflect real-world usage, since plant images are most commonly taken with smartphone cameras in natural settings. Rather than developing a web-based interface, the project focuses exclusively on mobile deployment to emphasize portability, accessibility, and practical applicability. The

mobile application serves as the primary user interface, while computationally intensive tasks such as backend processing and language generation are handled on the server.

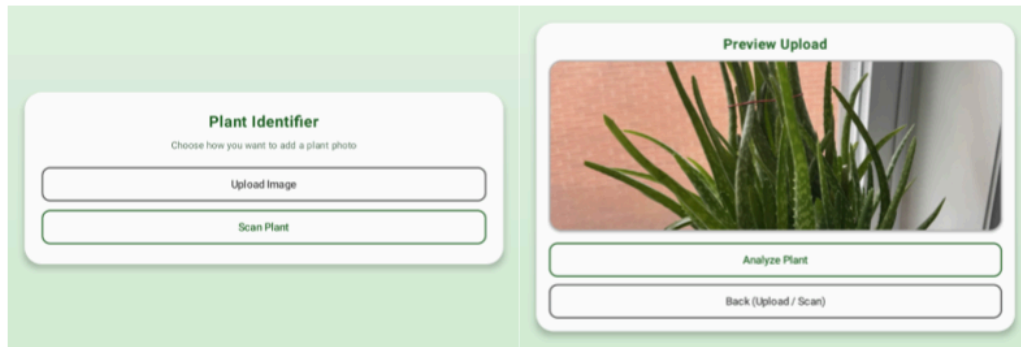


Figure 5. Planterbox mobile application interface.

The mobile application is structured around functional modules that correspond to the stages of the system pipeline illustrated in Figure 1. It provides a streamlined and intuitive interface that allows users to interact with the system in a small number of steps. As shown in Figure 5, the main screen enables users to capture an image using the device camera or select an image from the gallery, and then preview the selected image for analysis. The interface is intentionally minimal to ensure usability for non-technical users, including hobbyist gardeners and everyday plant owners.

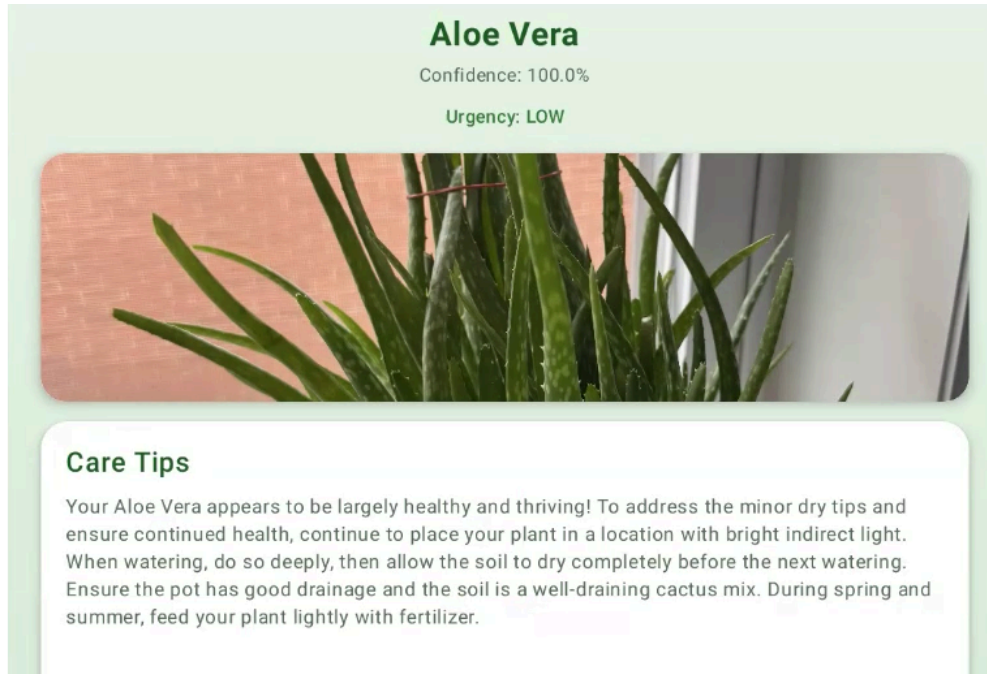


Figure 6. Planterbox mobile application result interface.

On the backend, the system is responsible for executing inference. After receiving an image, the server performs image decoding and preprocessing, then applies the model to predict the plant species. To improve interpretability and overall user experience, the system also integrates a language model that generates human-readable explanations based on the model's prediction and the input image. The language model processes the prompt shown in Figure 4 and returns an explanation describing the plant's condition along with care recommendations. The final response returned to the mobile application includes both the classification result and the generated explanation. The application presents this information in a clear and readable format, prioritizing interpretability over technical detail. This helps reduce the likelihood of user misinterpretation and encourages practical use of the system.

The following technologies were used in the implementation: Android Studio for mobile development, Python for the backend, MobileNetV3 as the model architecture, Gemini 2.5 Flash as the language model, and GitHub for version control. Several implementation techniques were employed to improve reliability and usability, including client-server separation, consistent image preprocessing, asynchronous backend requests, and modular backend design. Client-server separation keeps the mobile application lightweight while offloading heavier computation. Consistent image preprocessing ensures alignment between training and inference

behavior. Asynchronous backend requests help maintain UI responsiveness during model inference and LLM processing. Modular backend design enables independent updates to the classifier or language model components.

The main challenges encountered during this project involved coordinating communication between the mobile application, backend inference service, and the language model API. Common runtime errors included “AI analysis failed: Failed to connect ...” as well as HTTP error codes such as 429 (quota/rate limit) and 503 (service unavailable). Typical causes included lack of internet connectivity, network restrictions, or the backend service being asleep or temporarily unavailable. These issues were compounded by the use of a free-tier plan for Gemini 2.5 Flash, which enforces strict request limits (e.g., at most 10 requests per minute and 250 requests per day). In practice, most failures were mitigated by retrying after 30–60 seconds, testing with a different image, and restarting or waking the backend service to re-establish a valid connection to the language model.

Results and Evaluation

The primary goal of the evaluation phase was to assess the effectiveness of the Planterbox system in accurately identifying plant species from real-world images and delivering interpretable results to end users. This evaluation focuses on the plant identification performance of the GreenThumbV1 model implemented in this project. GreenThumbV1 was evaluated using accuracy and macro F1-score. Accuracy measures how often the model’s species predictions are correct, while macro F1-score provides a balanced measure of classification quality across classes.

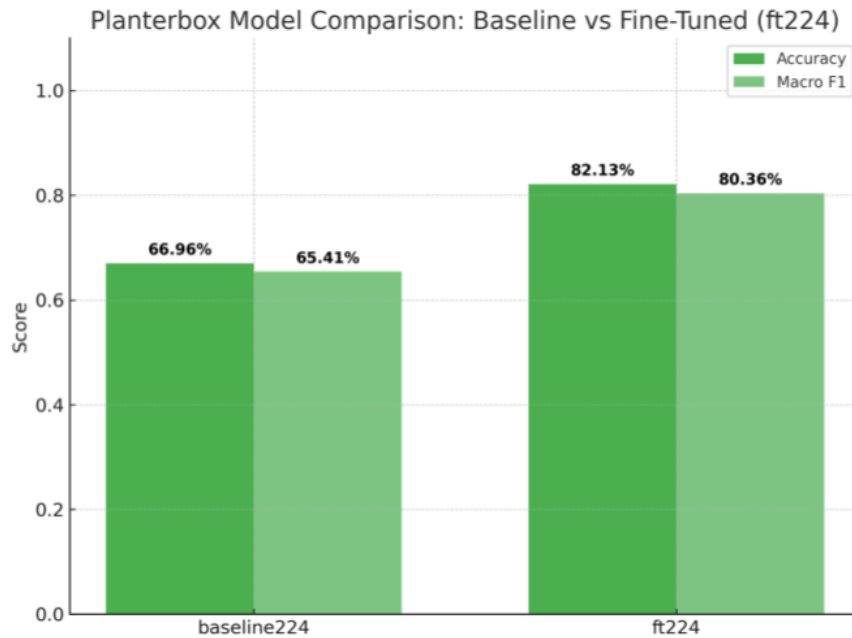


Figure 7. GreenThumbV1 model comparison with baseline MobileNetV3.

Figure 7 compares the accuracy and macro F1-score between the baseline MobileNetV3 model and the fine-tuned version, referred to as GreenThumbV1. The results show that fine-tuning MobileNetV3 improves performance across both evaluation metrics. The baseline model achieves an accuracy of 66.96% and a macro F1-score of 65.41 (averaged across 12 epochs). After fine-tuning, GreenThumbV1 achieves an accuracy of 82.13% and a macro F1-score of 80.36. The improvement in macro F1-score is substantial, indicating that fine-tuning improves performance across different plant species classes rather than disproportionately favoring a small subset. These results suggest that fine-tuning helps the model learn plant-specific visual features—such as leaf texture, discoloration, and structural patterns—that are less emphasized in generic pretraining.

```

===== Evaluation Results =====
BLEU Score: 0.0000

ROUGE Scores:
  rouge1: 0.6383
  rouge2: 0.2667
  rougeL: 0.5106

BERTScore:
  Precision: 0.5844
  Recall:    0.2809
  F1 Score:  0.4293
=====

```

Figure 8. Language model evaluation results.

In addition to evaluating the performance of the GreenThumbV1 model, we also evaluated the Gemini 2.5 Flash language model. This was done by comparing the model’s generated text against a dataset from Kaggle titled “Plant Growth and Care Recommendations.” The dataset provides basic care information for selected plant species, including growth rate, soil type, sunlight requirements, watering guidance, and fertilization type.

Common evaluation methods for generative AI include BLEU, ROUGE, and BERTScore. BLEU (Bilingual Evaluation Understudy) measures similarity between generated and reference text using n-gram overlap. ROUGE measures how much reference content is captured in the generated text. BERTScore measures semantic similarity by comparing contextual embeddings. However, BLEU and ROUGE are best suited for outputs that closely match reference wording, while this model produces free-form text. As a result, BLEU may be low even when the content is correct, and ROUGE may undercount paraphrases that preserve meaning.

As shown in Figure 8, the BLEU score was 0, which is consistent with BLEU heavily penalizing mismatched higher-order n-grams (e.g., 4-grams). The ROUGE results indicate that 64% of important unigrams and 27% of important bigrams were captured, reflecting moderate structural similarity. BERTScore produced a precision of 58%, a recall of 28%, and an F1-score of 43%, suggesting that the generated recommendations matched the reference meaning moderately well.

While the results demonstrate promising performance for GreenThumbV1, the same cannot be concluded as strongly for the language model evaluation. For GreenThumbV1, no large-scale testing was performed beyond the experiments reported, largely due to time

constraints and limited computational resources. Training and testing with large image datasets is time-consuming, and using hundreds of images across multiple experiments can require extensive training time without access to higher-end hardware.

For the language model, evaluation was also challenging because there is no single “correct” reference response for free-form recommendations. Instead, we relied on commonly used text-similarity metrics that provide partial signals of quality. These limitations highlight areas for future improvement in both evaluation and system robustness. Overall, the evaluation confirms that Planterbox can accurately classify plant species using a lightweight model and can operate reliably with a mobile-to-backend architecture. These results validate the system’s design choices and support its value as a real-world mobile plant identification and health assessment tool.

Objective and Solution

The primary objective of this project was to design and implement a practical, mobile-based system capable of identifying plant species from user-provided images while providing care recommendations. Based on the experimental results and evaluations, this objective was largely achieved. The GreenThumbV1 model demonstrated strong classification performance after fine-tuning, significantly outperforming the baseline MobileNetV3 model. Additionally, the complete pipeline—from image capture on a mobile device to backend inference and language model guidance—operated reliably within the constraints of the project. Together, these results indicate that the system meets the core functional and performance goals established for Planterbox.

One of the key strengths of the proposed solution is its balanced system design, which prioritizes both computational efficiency and interpretability. By selecting MobileNetV3 as the backbone architecture, the project achieves a favorable trade-off between accuracy and prediction cost, making the system suitable for mobile-oriented deployment. Fine-tuning the model on plant-specific images proved critical, as shown by the substantial improvements in accuracy and macro F1-score. This supports the conclusion that pretrained convolutional networks benefit from task-specific adaptation, particularly for visual tasks such as plant identification. Another notable strength is the integration of a language model to generate

human-readable explanations. This transforms predictions into natural language guidance that aligns more closely with real-world usage scenarios.

Despite these strengths, the system has limitations. The classification task does not identify specific plant diseases, and model performance is constrained by the size and diversity of the available dataset. In addition, while the mobile application provides a complete functional interface, extensive user testing was not performed. From a practical standpoint, this project should be viewed as a decision-support tool rather than a definitive diagnostic system. Incorrect classifications could lead to inappropriate plant care actions if users rely solely on the output. Future work could expand health assessment to include disease-specific classes or nutrient deficiency indicators, and increase the number of plant species supported by the model.

Conclusion

This project presented Planterbox with GreenThumbV1, a mobile-based plant identification and care recommendation system that integrates lightweight deep learning with a practical user application. The primary objective was to develop a deployable and efficient solution capable of identifying plants and providing care recommendations from a single image, and this goal was successfully achieved. Fine-tuning MobileNetV3 proved effective for plant species identification. Experimental results showed clear improvements over the baseline model in both accuracy and macro F1-score, demonstrating that lightweight architectures can deliver reliable performance when adapted to project-specific data. These results validate the design choices and support the system's suitability for mobile-oriented deployment.

The project implemented an end-to-end pipeline that includes image capture, preprocessing, model inference, and result presentation within a mobile application. Integrating a language model to generate natural language explanations improved interpretability and enhanced the overall user experience by translating predictions into actionable feedback. While the system is limited to a small set of species and relies solely on visual input, this aligns with real-world use cases where fast and understandable feedback is most valuable. Overall, this project demonstrates the practicality of efficient AI models in real-world applications and provides a strong foundation for future improvements such as multi-class disease detection, expanded datasets, and broader user testing.

References

- Dari, S. S., Tiwaskar, S., Kumar, J. R. R., Langote, V., Mohadikar, G. M., & Ashtagi, R. (2024). *Enhancing plant leaf disease identification with a CNN and DenseNet hybrid model*. Proceedings of the 5th International Conference on Information Management & Machine Intelligence (ICIMMI 2023), Article 102, 1–6. <https://doi.org/10.1145/3647444.3647929>
- Hajoub, M. W., Touil, H., & Begdouri, M. A. (2024). *Plant disease detection using a hybrid approach based on vision neural network transformers*. Proceedings of the 7th International Conference on Networking, Intelligent Systems and Security (NISS 2024), Article 5, 1–6. <https://doi.org/10.1145/3659677.3659685>
- Yao, J., Tran, S., Garg, S., & Sawyer, S. (2023). *Deep learning for plant identification and disease classification from leaf images: Multi-prediction approaches*. arXiv. <https://arxiv.org/abs/2310.16273>