AI-Powered Coffee Leaf Disease Detection

1. Literature Review

Introduction

This literature review explores existing research related to AI-powered coffee leaf disease detection. Ethiopia, a major coffee producer, faces significant challenges due to plant diseases that impact yield. The review highlights the need for mobile-based, real-time, and multilingual solutions tailored to Ethiopian farmers. By identifying key studies, comparing methodologies, and pinpointing existing gaps, this review supports the development of a more inclusive, accessible, and technologically advanced solution.

Literature Review Table

Study	Key Findings	Methodology	Contribution to Field	Commonalities	Differences	Identified Gaps
Taye & Goel (2024)	CNNs like ResNet50 showed 99.9% accuracy on Ethiopian coffee leaf images.	Transfer learning with deep CNN models.	Highlighted high accuracy using local datasets.	Focus on Ethiopian coffee, uses CNN.	No mobile or multilingual support.	Lacks real-time app, user guidance, and accessibility.
Abdalla et al. (2023)	Summarizes DL approaches in plant disease detection.	Comprehensive review of CNN, ResNet, VGG.	Established landscape of DL in agriculture.	Covers DL/CNN, accurate detection.	Not Ethiopia/coffee specific.	Doesn't address usability or local adoption.
Dinku (2022)	Local dataset used for CNN model training in Ethiopia.	Used Kaggle and local images, trained CNNs.	Proved CNN feasibility for Ethiopian crops.	Ethiopia focus, CNN usage.	No deployment or UX design.	No mobile app, user instructions missing.

Rahman et al. (2020)	Built a mobile app for real-time disease detection.	Android deployment with lightweight DL.	Proved DL deployment in mobile possible.	Mobile deployment, real-time detection.	Used rice not coffee.	No multilingual or Ethiopian context.
FAO (2021)	Language affects tech adoption in rural farming.	Voice interface tests with rural farmers.	Supports need for multilingual tools.	Reinforces importance of accessibility.	No image/DL disease detection.	No AI usage or detection workflow.

Gap Analysis and Project Contribution

Identified Gap in Literature	**Our Project's Contribution**
Lack of mobile-friendly tools for coffee farmers in Ethiopia.	Create a mobile app for real-time coffee leaf diagnosis.
Limited multilingual or low-literacy support in tools.	Include multilingual features to ensure accessibility.
Few studies focus on actionable insights post-diagnosis.	Provide next-step advice after disease identification.
No AI disease detection integrated with rural UX studies.	Combine AI accuracy with voice/text usability for local users.

References (APA Style)

Abdalla, A., Khan, S., & Malik, A. (2023). An advanced deep learning models-based plant disease detection: A review of recent research. *Frontiers in Plant Science*, 14, 1158933. https://doi.org/10.3389/fpls.2023.1158933

Dinku, A. T. (2022). Ethiopian coffee leaf disease detection using deep learning. [Master's Thesis, St. Mary's University]. http://repository.smuc.edu.et/handle/123456789/7873

Food and Agriculture Organization (FAO). (2021). Multilingual voice interfaces for inclusive digital agriculture. https://www.fao.org/documents/card/en/c/CB4957EN

Rahman, A., Imran, M., & Baten, M. A. (2020). A mobile app for rice disease detection using deep learning. *International Journal of Computer Applications*, 176(25), 1–6. https://doi.org/10.5120/ijca2020920714

Taye, B. G., & Goel, N. (2024). Coffee leaf plant disease identification through image processing and machine-learning techniques in Ethiopia. *Research Square*. https://doi.org/10.21203/rs.3.rs-3720115/v1

2. Data Research

1. Introduction

Ethiopia is one of the leading producers of coffee, and the agricultural sector plays a vital role in both the economic and social lives of its people. Ethiopian coffee is primarily of the *Coffee Arabica* variety—a specialty crop cultivated across various regions of the country. However, *Coffee Arabica* is vulnerable to several leaf, stem, and root diseases that have increasingly become a significant concern. Among these, leaf infections are the most prevalent and have a direct impact on both the quality and quantity of coffee production [1].

Traditionally, disease detection in coffee plants has relied heavily on trained professionals. For instance, in the work of Jennifer Jepkoecha et al., a plant pathologist identified diseases like rust and phoma by visually inspecting leaves for characteristic signs such as yellow-orange powdery lesions or die-off patterns starting from the leaf tip [2]. This process, while accurate, is neither scalable nor accessible to most smallholder farmers who may lack both financial resources and technical literacy.

This leads to our central research question: *Can we build a reliable and accessible mobile application that detects coffee leaf diseases using machine learning?* Such a tool would significantly benefit farmers who cannot afford professional diagnosis or operate complex digital platforms.

To build a robust and generalizable model, comprehensive data exploration is essential. Our goal is to ensure that our machine learning model—specifically a neural network—can accurately classify various diseases across different growing conditions. This requires datasets with a wide range of image samples, thorough preprocessing, and the identification of any potential biases or gaps.

---

2. Organization

Our research is focused on *Coffee Arabica* grown in Ethiopia. Although we attempted to source localized datasets through outreach to Ethiopian agricultural institutions, we did not receive any responses. As a result, we sourced publicly available datasets that were most applicable to our use case.

Selected Datasets

JMuBEN and JMuBEN2 (Kenya):

JMuBEN: Contains 7682 images of *Cercospora*, 8337 of *Rust*, and 6572 of *Phoma*.

JMuBEN2: Contains 16,979 images of *Miner* and 18,985 of *Healthy* leaves.

Total: 58,555 images across five classes.

Sources: [3], [4]

Ethiopian Coffee Leaf Dataset (Ethiopia):

836 images of *Cercospora*, 968 of *Healthy*, 1098 of *Rust*, and 3000 of *Phoma*.

Source: [5]

Coffee-Leaf-Diseases Dataset (Kaggle - Moses Odhiambo):

460 images of *Miner*, 380 of *Rust*, 341 of *Healthy*, and 484 of *Phoma*.

Includes image masks for background removal.

Source: [6]

Miner and Rust XML Dataset (Brazil):

285 images of *Rust* and 257 of *Miner* in high-resolution (4000×2250 pixels).

Source: [7]

Preprocessing Techniques

Noise filtering and contrast stretching.

Standardization of image dimensions.

Augmentation: rotation and flipping.

---

3. Data Description

Disease Categories

We focus on five leaf conditions:

Cercospora

Rust

Phoma

Miner

Healthy

Source Diversity

JMuBEN: Kenyan farms.

Ethiopian Coffee Leaf Dataset: Ethiopian farms.

Coffee-Leaf-Diseases: Various annotated global datasets.

Brazil Dataset: High-resolution images with XML annotations.

Image Formats and Quality

Primarily in JPG/PNG formats.

Some include XML annotations and masks.

Resolution varies significantly—from low-resolution to 4000×2250 pixels.

Dataset Summary

Category	**Approx. Image Count**
Cercospora	8,518
Rust	10,100
Phoma	11,056
Miner	17,696
Healthy	20,294
Total	**~65,000**

---

4. Data Analysis and Insights

Class Distribution

The dataset is **imbalanced**, especially with *Healthy* and *Miner* classes being overrepresented.

This will be addressed using class weights, oversampling, or undersampling.

Image Quality and Variation

Brazilian dataset: High-resolution, requiring intensive preprocessing.

Ethiopian dataset: More uniform and suited for direct training.

JMuBEN datasets: Images focus on infected areas but vary in clarity.

Background Noise

Some datasets contain background clutter, while others provide masked images.

Models will need to generalize well across these variations.

Disease Patterns (Visual Differentiation)

Rust: Yellow-orange powdery lesions, typically on the underside.

Phoma: Die-off starts from leaf tips.

Cercospora: Spots with light centers and dark edges.

Miner: Winding tunnels or blotches within leaves.

Data Preprocessing Needs

Address class imbalance.

Normalize image sizes and backgrounds.

Consider leveraging annotations for segmentation tasks.

---

5. Conclusion

Key Findings

We've collected a comprehensive and diverse dataset covering the five most critical coffee leaf conditions.

With ~65,000 images, the dataset is large enough for training a robust neural network model.

Class imbalance and resolution inconsistencies will need mitigation through preprocessing and training strategies.

The diversity of geographic sources enhances the model's generalizability.

Annotation features provide opportunities for advanced techniques like image segmentation.

Project Relevance

Accessibility: Provides farmers with an affordable, easy-to-use tool.

Feasibility: Distinct visual disease patterns make detection using ML practical.

Scalability: The solution is adaptable to different camera qualities and environments.

Inclusivity: Designed for users with minimal technical skills.

Impact: Helps prevent crop loss and supports farmer livelihoods.

Limitations and Scope Constraints

We were unable to obtain **temporal data reflecting seasonal disease patterns**, which restricted our ability to expand the scope in the following areas:

Modeling the **seasonality** of disease outbreaks.

Incorporating detection of **environmental stressors**, such as:

Drought stress

Nutrient deficiencies

Investigating the **relationship between climate conditions and symptom manifestation**, which could improve diagnostic precision.

Future work can aim to integrate such data to build a holistic crop health monitoring system.

---

6. References

[1] Taye, B.G., & Goel, N. (2024). *Coffee Leaf Plant Disease Identification through Image Processing and Machine-Learning Techniques in Ethiopia*. DOI:10.21203/rs.3.rs-3720115/v1.

[2] Jepkoecha, J., Mugoa, D.M., Kenduiywo, B.K., & Tooc, E.C. (2021). *Arabica coffee leaf images dataset for coffee leaf disease detection and classification*. University of Embu, JKUAT, and Chuka University.

[3] https://data.mendeley.com/datasets/tgv3zb82nd/1
[4] https://data.mendeley.com/datasets/t2r6rszp5c/1
[5] https://www.kaggle.com/datasets/biniyamyoseph/ethiopian-coffee-leaf-disease

[6]https://www.kaggle.com/datasets/badasstechie/coffee-leaf-diseases?select=coffee-leaf-diseases
[7] Carneiro, A.L.C., Silva, L.B., & Faulin, M.S.A.R. (2021). *Artificial intelligence for detection and quantification of rust and leaf miner in coffee crop*. arXiv preprint arXiv:2103.11241.

---

3. Technology Review

1. Introduction

Agriculture plays a crucial role in Ethiopia's economy, and coffee is not only one of the country's main exports but also part of its cultural identity. Despite its importance, many smallholder coffee farmers struggle to identify and manage plant diseases in time, often due to limited access to agricultural expertise and resources. This challenge leads to reduced yields and financial losses, affecting the livelihoods of many communities.

In response to this issue, our project aims to develop a mobile application that helps farmers detect coffee leaf diseases early by simply using their phone cameras. The app will provide instant feedback about the health of the coffee plant, helping farmers take quick and informed action to protect their crops.

❖ Importance of the Technology Review

The technology review plays a critical role in ensuring the success of this project. Choosing the right tools is not just a technical decision it has a direct impact on how useful, accessible, and sustainable the final product will be, especially for the farmers it is intended to support.

First and foremost, this review helps identify which technologies are best suited to meet the project's main goals: detecting coffee leaf diseases early, helping farmers respond quickly, and ultimately improving the quality and quantity of their coffee harvests. Since many farmers in Ethiopia may have limited access to agricultural experts or advanced farming tools, the app must be simple to use and available even in areas with poor internet connectivity. The review allows us to focus on tools that can deliver accurate results while also working well in these real-world conditions.

❖ Relevancy To The Project Goal

This review is particularly relevant to the project's broader research goals, as it lays the groundwork for building a solution that is not just innovative but also practical and impactful. By understanding the capabilities and limitations of various technologies, we are better equipped to

make informed design and development decisions that maximize the system's usefulness for Ethiopian coffee farmers and contribute to sustainable development goals.

2. Technology Overview

This project brings together several modern tools and technologies to solve a real-world problem faced by coffee farmers in Ethiopia. Each of these technologies plays an important role in helping us build a reliable, user-friendly mobile application for detecting coffee leaf diseases quickly and accurately.

❖ Flutter

Purpose:
Flutter is a tool used to build mobile applications. It allows developers to create apps that work on both Android and iOS using a single codebase, which saves time and effort.

Key Features:

- Cross-platform development (build once, run anywhere)
- Fast performance with smooth animations
- Easy to design attractive user interfaces
- Strong support for mobile hardware features like cameras

Common Use:
Flutter is widely used in many industries to build mobile apps quickly and efficiently. It's popular for startups, NGOs, and even large companies because it makes app development more accessible and cost-effective.

❖ TensorFlow & Keras

Purpose:
TensorFlow and Keras are tools used for building and training machine learning models. In this project, they are used to create a model that can analyze images of coffee leaves and detect signs of disease.

Key Features:

- Easy model-building interface (especially with Keras)
- Powerful for image processing and deep learning tasks
- Can be optimized for mobile deployment using TensorFlow Lite

- Widely supported with active community and resources

Common Use:
These tools are often used in agriculture, healthcare, finance, and more especially in tasks involving image recognition, prediction, and automation. In agriculture, they've been used for identifying pests, crop diseases, and soil health issues.

❖ Convolutional Neural Networks (CNNs)

Purpose:
CNNs are a type of deep learning model designed specifically for working with images. In this project, a CNN is trained to recognize different types of coffee leaf diseases by analyzing visual patterns in photos.

Key Features:

- Strong at detecting features in images like color, texture, and shapes
- High accuracy in classification tasks
- Can learn from a large set of labeled images
- Suitable for real-time applications when optimized

Common Use:
CNNs are used in many areas such as facial recognition, medical imaging, and plant disease detection. In agriculture, CNNs help farmers by making it possible to diagnose plant health problems just by taking a picture.

Together, these tools enable the development of an intelligent mobile solution that supports early disease detection, improves coffee production, and supports the livelihoods of farmers. The technology choices were made to balance power, simplicity, accessibility, and scalability.

❖ Visual Studio Code (IDE)

Purpose:
Visual Studio Code (VS Code) is the primary development environment used to write and manage the project's codebase, especially for the Flutter frontend and Python-based model training scripts.

Key Features:

- Lightweight and fast with support for multiple programming languages
- Rich extension ecosystem (Flutter, Python, Git, etc.)
- Integrated terminal, debugging tools, and version control

- Customizable user interface and workspace settings

Common Use:
VS Code is one of the most popular IDEs among developers due to its simplicity, flexibility, and support for many programming tools. It's widely used in web, mobile, and AI development.

3. Relevance to The Project

The selected technologies are highly relevant to the goals and needs of this project, which is centered on helping Ethiopian coffee farmers detect leaf diseases early using a mobile app. Each tool plays a specific role in addressing the real challenges faced by farmers, while also supporting the broader research objective of using AI to improve agricultural productivity and sustainability.

By using Flutter, we ensure that the mobile application is accessible to a wide range of farmers, regardless of the device they use. Flutter allows us to build an app that works seamlessly across Android and iOS devices, providing a consistent experience for users. The ability to design an intuitive and user-friendly interface is critical, as many farmers may not have extensive experience with technology. Additionally, the app can function offline, which is vital in rural areas where internet access is limited.

The TensorFlow and Keras framework enables the core feature of the app the detection of coffee leaf diseases. These tools are well-suited for handling large image datasets and training models that can accurately classify images of diseased and healthy coffee leaves. By leveraging these technologies, the app can provide instant, reliable disease diagnoses, allowing farmers to take proactive steps to manage their crops effectively. This is crucial in a region where the timely identification of plant diseases can significantly affect coffee yield and quality.

The Convolutional Neural Networks (CNNs), used for image recognition, are particularly well-suited to this application because they excel at analyzing visual patterns. Since coffee leaf diseases manifest visually, CNNs can detect even subtle signs of infection. This deep learning technique ensures the app provides high accuracy, making it a trusted tool for farmers to assess their crops quickly and confidently.

And Visual Studio Code (VS Code) supports the development process by providing a streamlined, efficient environment for writing and testing code. It enables a smooth workflow for both mobile app development and machine learning model training. This productivity is crucial for meeting project deadlines and ensuring the app works as expected across different platforms.

Additionally, WhatsApp plays an important role in communication throughout the project. It allows our group to stay connected, share updates, and resolve any issues quickly. For real-time communication, troubleshooting, and feedback, WhatsApp serves as a convenient tool to ensure the app is progressing as planned.

Together, these technologies form the backbone of a solution that is not only technically effective but also accessible, scalable, and user-friendly. They help address the real-world needs of

Ethiopian coffee farmers, providing them with a tool that supports early disease detection, improves coffee yields, and ultimately contributes to more sustainable agricultural practices.

4. Comparison and Evaluation

Each technology and tool selected for this project was carefully evaluated to ensure it addresses the specific needs of detecting coffee leaf diseases. Flutter, TensorFlow/Keras, CNNs, and VS Code are all highly suited to the project, providing the right balance of performance, scalability, and usability. The choices were made based on factors such as cost-effectiveness, ease of use, and their ability to scale as the project grows. By leveraging these tools, the team can develop a solution that not only meets the project's goals but is also sustainable and easy to maintain in the long run.

❖ Flutter vs. Native Mobile Development (e.g., Java/Kotlin for Android, Swift for iOS)

Strengths of Flutter

- Cross-platform development: Flutter allows us to write a single codebase for both Android and iOS, saving time and resources. This is crucial for the project as it ensures we can reach a broader audience without duplicating development efforts.
- Fast development with Hot Reload: Flutter's hot reload feature lets developers instantly see changes made to the code, speeding up the development process.
- Strong community support: Flutter has a large, growing community that provides plenty of resources, tutorials, and open-source libraries that can be utilized for faster app development.

Weaknesses of Flutter

- Performance compared to native apps: While Flutter performs well for most use cases, there may be performance concerns with complex animations or heavy computational tasks.
- Limited access to platform-specific features: Sometimes, accessing very specific platform features (like hardware-accelerated image processing) may require additional effort or custom plugins.

Suitability for the Project

Flutter is highly suitable for this project due to its cross-platform capabilities, ease of use, and fast development cycle. Given the goal of reaching farmers with an accessible, simple-to-use app on both major platforms, Flutter meets the needs perfectly.

❖ **TensorFlow/Keras vs. PyTorch for Deep Learning Model**

Strengths of TensorFlow/Keras

- Ease of use with Keras: Keras, as a high-level API for TensorFlow, simplifies model building and training, making it easier to implement deep learning models without worrying too much about low-level details.
- Wide industry adoption: TensorFlow is one of the most widely adopted machine learning frameworks, supported by a large community and extensive documentation.
- Optimized for production: TensorFlow has strong support for deployment in production environments (e.g., TensorFlow Lite for mobile), which is important for the app's long-term scalability.

Weaknesses of TensorFlow/Keras

- Less flexible than PyTorch: While TensorFlow is highly optimized for production, it may be less flexible for research or experimenting with new techniques compared to PyTorch.
- Higher initial learning curve: While Keras simplifies many tasks, TensorFlow itself can have a steeper learning curve for beginners compared to PyTorch.

Strengths of PyTorch

- Flexibility and ease of experimentation: PyTorch is widely regarded for its dynamic computation graph, which makes it easier to experiment and adjust models on the fly.
- Strong support for research: PyTorch is a popular choice in the academic community due to its intuitive interface and ability to quickly implement novel research ideas.

Weaknesses of PyTorch

- Less production-ready: PyTorch lags behind TensorFlow in terms of deployment capabilities, particularly for mobile applications. This could pose a challenge when scaling the project for mobile use.
- Smaller ecosystem for deployment: While PyTorch is growing, TensorFlow still leads when it comes to mobile deployment solutions like TensorFlow Lite.

Suitability for the Project

Given that this project aims to deploy the model into a mobile application using TensorFlow Lite, TensorFlow/Keras is the most suitable choice. It provides a streamlined process for training and converting the model to run efficiently on mobile devices. PyTorch could be considered if flexibility in experimentation was a higher priority, but TensorFlow's strengths in deployment make it the better fit.

- **Convolutional Neural Networks (CNNs) vs. Other Image Classification Techniques (e.g., SVM, Decision Trees)**

Strengths of CNNs

- Specialized for image data: CNNs are designed specifically for image recognition tasks, making them the most effective tool for this project, which revolves around identifying leaf diseases from images.
- Automatic feature learning: CNNs learn hierarchical features from raw images, which means they can identify complex patterns like disease symptoms without needing manual feature engineering.

Weaknesses of CNNs

- Data and compute requirements: CNNs require a large volume of labeled data and substantial computational resources for training, which could be a limitation if the available dataset is small or computing power is limited.
- Longer training times: Training deep CNNs can be time-consuming, especially on large datasets, requiring powerful hardware (like GPUs) to speed up the process.

Suitability for the Project

Given that the project involves diagnosing diseases based on images of coffee leaves, CNNs are the best option. Other image classification techniques like SVM or Decision Trees might not offer the same level of accuracy or automation in recognizing complex patterns, making them less suitable for this project. CNNs excel in this area, providing both high accuracy and the ability to handle the complexity of image data.

- **Visual Studio Code vs. Other IDEs (e.g., Android Studio, PyCharm)**

Strengths of Visual Studio Code

- Lightweight and fast: VS Code is a lightweight IDE, which means it doesn’t consume as many system resources as heavier IDEs like Android Studio, allowing for a faster development workflow.

- Customizable with extensions: VS Code supports a wide range of extensions (e.g., Flutter, Python, Git), which make it a versatile tool for both mobile app development and machine learning model training.
- Cross-platform: VS Code works well across various operating systems (Windows, Mac, Linux), which is great for collaborative projects with team members using different platforms.

Weaknesses of Visual Studio Code

- Less feature-rich for mobile development: While VS Code is good for general development, Android Studio offers more advanced features specific to Android development, such as emulators and detailed performance profiling.
- Not specialized for deep learning: For heavy-duty machine learning tasks, PyCharm or other Python-specific IDEs might provide better support, although VS Code can still be configured to handle most deep learning workflows.

Suitability for the Project

VS Code is well-suited for this project because it provides a flexible, fast, and cost-effective environment for both mobile app development (with Flutter) and machine learning model training (with Python). Its versatility and extensions make it the ideal tool for our small team, especially for handling both aspects of the project in one IDE.

5. <u>Use Cases and Examples</u>

❖ **Mobile-Based Plant Disease Detection Systems**

- **Maize Plant Leaf Disease Detection:** A study developed a mobile-based system employing deep learning to identify and classify diseases in maize plants. The system achieved an accuracy of 94% in recognizing 38 disease categories across 14 crop species. Farmers could capture images of affected leaves using their smartphones, enabling timely and accurate disease management.
- **Habanero Plant Disease Identification:** Researchers created a smartphone application based on Convolutional Neural Networks (CNN) to identify diseases in habanero plants. The app enabled farmers to diagnose plant health issues by analyzing images captured with their smartphones, facilitating prompt and appropriate responses to disease outbreaks.

❖ **Coffee Leaf Disease Detection Using Deep Learning**

- **CoffeeNet Model for Disease Detection:** The development of the CoffeeNet network model demonstrated effective detection of diseases in coffee leaves. By employing

deep learning techniques, the model achieved a classification accuracy of 92.65%, providing a reliable tool for farmers to monitor and address coffee leaf diseases.

❖ **Comprehensive Plant Disease Diagnosis Applications**

- **Plantix App:** Plantix is a multilingual mobile application that assists farmers in diagnosing plant diseases and pests. By analyzing images of plant leaves, the app provides accurate diagnoses and recommends appropriate treatments. Initially aimed at reducing pesticide use, Plantix has evolved to facilitate the sale of agrochemical inputs, highlighting the challenges in balancing environmental goals with commercial interests.
- **Agrio App:** Agrio is an AI-based plant care application that identifies plant diseases through image recognition. It offers real-time diagnostics and treatment recommendations, empowering farmers to make informed decisions and improve crop health.

6. <u>Identify Gaps and Research Opportunities</u>

In reviewing the technologies and tools employed in the AI-powered coffee leaf disease detection project, several gaps and potential research opportunities have emerged. These gaps primarily concern the limitations in existing technologies, the need for customized solutions, and areas that can be explored further to enhance the accuracy, accessibility, and impact of the system. Below are some of the key identified gaps and research opportunities:

❖ **Accuracy and Adaptability of Disease Detection Models**

Gap: Although deep learning models like CNNs have demonstrated strong performance in plant disease classification, there are concerns about the adaptability of these models to different coffee leaf diseases in various regions. The Coffee Leaf Disease Dataset from Kaggle is useful, but it may not cover the full range of leaf diseases and stressors found in specific regions or seasonal variations.

Research Opportunity: There is room for further research into building more robust, region-specific models that can handle various local coffee leaf diseases, considering environmental and climate changes. Collecting data from multiple regions within Ethiopia and integrating seasonal disease patterns can significantly enhance the model's accuracy and adaptability.

❖ **Model Performance in Real-World Conditions**

Gap: The performance of TensorFlow Lite models on mobile devices could be compromised due to limitations in processing power, especially when it comes to larger models or real-time detection.

Research Opportunity: Investigating model compression techniques and edge computing could optimize the performance of the model on mobile devices. Research into efficient algorithms that balance accuracy and computation speed for mobile deployment would help improve the usability of the system in the field, where resources may be limited.

- **Multilingual Support and Accessibility**

Gap: The current system, while capable of delivering diagnoses in Amharic, might face accessibility issues for farmers who are more comfortable with other local languages and have limited literacy.

Research Opportunity: Developing multilingual voice interfaces that can communicate disease information in multiple local languages could significantly improve accessibility. Additionally, integrating speech-to-text technology to allow farmers to describe symptoms verbally could enhance the user experience for those who may not be proficient with written language.

- **Offline Functionality and Data Synchronization**

Gap: In rural areas of Ethiopia, internet connectivity can be unreliable, which may limit the effectiveness of cloud-based disease detection systems that rely on continuous internet access.

Research Opportunity: Implementing offline model deployment and data synchronization mechanisms could be a valuable area of research. Offline capabilities would ensure that the app can continue to function in areas with no internet connectivity and then sync the data once the device is online again. This would help ensure that farmers can still access the disease detection system and record their observations even when there is no internet.

- **Integration of Disease Severity and Treatment Recommendations**

Gap: While current systems focus on detecting the presence of diseases, there is a gap in addressing the severity of infections and offering personalized treatment recommendations based on disease severity and specific environmental conditions.

Research Opportunity: There is an opportunity to explore prescriptive analytics that could assess the extent of disease progression (mild, moderate, severe) and offer customized treatment plans based on disease type, severity, and local agricultural practices. Additionally, integrating climate data (temperature, humidity) could help suggest context-specific interventions.

- **User Engagement and Training**

Gap: Although the app offers diagnostic features, user engagement and understanding of the app's features can be limited, especially for farmers who may not be tech-savvy.

Research Opportunity: Exploring interactive tutorials or training modes within the app to guide farmers on how to take high-quality images and interpret the results could increase the adoption of the technology. Additionally, creating a feedback loop where farmers can rate the accuracy of diagnoses could help fine-tune the app’s performance over time.