Leveraging Computer Vision and Chatbot Tools for Plant Health Detection

Marcus Teck Wey Gan1, a), Shih Yin Ooi1, 2, b), Yee Jian Chew1, 2, c), Ying Han Pang1, 2, d) and Kiu Nai Pau1, 3, e)

1*Faculty of Information Science and Technology (FIST), Multimedia University, Jalan Ayer Keroh Lama, Melaka, 75450, Malaysia*

2*Centre for Advanced Analytics (CAA), COE for Artificial Intelligence, Multimedia University, Jalan Ayer Keroh Lama, Melaka, 75450, Malaysia*

3*Centre for Cybersecurity and Quantum Computing (CCQ), COE for Advanced Cloud, Multimedia University, Jalan Ayer Keroh Lama, Melaka, 75450, Malaysia*

*b) Corresponding author: syooi@mmu.edu.my*

*a) 1201103149@student.mmu.edu.my*

*c) chewyeejian@mmu.edu.my*

*d) yhpang@mmu.edu.my*

*e) knpau@mmu.edu.my*

**Abstract.** Even as artificial intelligence becomes increasingly prevalent and advanced across most industries, plant diseases proceed to threaten global food security, specifically in rural and underdeveloped regions. Furthermore, most of the smallholder farmers still rely on manual observation, which is slow and often inaccurate. This project presents an integrated plant health monitoring system that combines real-time computer vision, deep learning, and generative AI to address these challenges. YOLOv9s is used for plant detection, DeepSORT tracks individual leaves, and ResNet50 classifies them into one of sixteen disease categories. In the real-time monitoring component, classification results are logged and used by a Retrieval-Augmented Generation (RAG) pipeline through a CSV agent and Llama 3.2b to generate treatment prescriptions. In contrast, the chatbot module (Farmiz), driven by Gemma3, provides context-aware responses without relying on RAG or CSV inputs. Experimental testing shows that YOLOv9s achieved a mAP@50 of 0.994, while ResNet50 reached 99.53% validation accuracy. This system integrates the gap between detection along with decision-making which offers farmers fast, accurate, and practical support for disease management, while promoting digital agriculture.

# INTRODUCTION

Recently, the agricultural sector has progressively adopted AI technology as a way to enhance productivity and protect crops from various threats such as pests and diseases. This is because plant diseases remain a major cause of yield loss, especially in rural regions or developing countries. Therefore, early detection and fast intervention are important in managing such issues. Nonetheless, most of the smallholder farmers continue to rely on manual observation. This method is often slow, inconsistent, and inaccurate. As a result, this situation underscores the pressing need for intelligent systems that apply real-time computer vision and AI to improve disease management strategy [2] [3].

This project introduces an integrated plant health monitoring system that addresses these challenges by combining object detection using YOLOv9s, tracking with DeepSORT, and disease classification via ResNet50. The results are processed through a Retrieval-Augmented Generation (RAG) pipeline powered by Llama 3.2b and a CSV agent to produce treatment prescriptions. Additionally, the system includes a chatbot named Farmiz, powered by Gemma3, which allows users to upload images or ask questions. To illustrate, the system operates in two modes which are a live webcam monitoring and chatbot interaction. Both are designed to deliver accurate and timely recommendations to assist farmers. Despite growing advances in digital agriculture, many farmers in developing regions still rely on conventional methods that require frequent field inspections and experience. Thus, these practices are labour-intensive and often lead to delayed responses, worsening crop conditions. Although apps like Plantix and PlantVillage offer image-based detection, yet they lack real-time feedback and personalised advice and a conversational interface for personalized interactions. They typically function in isolation without combining detection, decision-making, and conversational interfaces. Hence, there is a clear need for a more unified and responsive solution.

The main objective of this project is to develop a user-friendly plant health monitoring system that enables early detection and provides instant treatment guidance. This involves real-time object detection and tracking, disease classification into sixteen categories, and prescription generation through LangChain’s RAG using Llama 3.2b. Meanwhile, the Farmiz chatbot supports natural interaction, driven by Gemma3. Lastly, the system is built using FastAPI and Next.js. This is to ensure responsive deployment and integration. Altogether, the solution aims to support better decisions and promote sustainable farming practices.

In addition, these objectives were carefully designed in response to the real-world challenges highlighted in digital agriculture. One of the main priorities was to improve generalisation, especially in the leaf detection stage. The dataset used was a subset of the New Plant Diseases Dataset on Kaggle, which had already been augmented offline by the original uploader. Meanwhile, additional image augmentations were applied in Roboflow, including rotation, flipping, and brightness adjustments to enhance it further. However, although the system has not yet been tested in extreme field conditions, these steps help improve its ability to recognise leaves accurately under typical lighting environments.

Another important goal was ensuring scalability. The system is capable of running efficiently even on entry-level laptops with NVIDIA GTX 1650 GPUs. Its modular structure also allows for future expansion, meaning that additional plant types and disease classes can be added by extending the dataset. In terms of deployment, the system is flexible. It can be hosted on a server or run locally on a computer. Despite the real-time webcam monitoring mode requires internet access to connect to the IP webcam, yet farmers in low-connectivity areas can still use the chatbot feature. For instance, they can receive immediate diagnostic results and treatment suggestions without needing to access the live video feature by uploading an image.

Lastly, the chatbot was designed to go beyond basic prediction. Its purpose is to help users understand the diagnosis and take action. Instead of forcing users to search online or rely on experts, the chatbot explains the issue and provides step-by-step treatment advice in simple language. This makes it easier for people without technical knowledge to manage plant health confidently.

The main contributions of this project are as follows. First, it combines two ways of detecting plant health into one website: a real-time monitoring feature that uses the webcam to detect leaves, identify diseases, and generate short prescription on the spot, along with a chatbot interface that allows users to upload a photo of a plant and receive a diagnosis. Thus, this makes the system flexible and easy to use, whether on a farm or at home. Second, the system uses a combination of deep learning models to handle different tasks. To illustrate, YOLOv9s detects and locates the leaf in an image, DeepSORT tracks it across frames in live video, and ResNet50 classifies the leaf into one of sixteen possible health conditions. These models work together behind the scenes to analyse the plant quickly and accurately. Third, the system uses two types of AI models to help users understand what’s happening to their plants. In the real-time monitoring mode, a small language model called Llama 3.2 3B is used to automatically generate a short prescription based on the detection results. In the chatbot mode, a larger model called Gemma 3 4B powers the Farmiz assistant. After the user uploads an image, the system processes it through the detection pipeline, and then the chatbot can answer questions like “What’s wrong with my plant?” or “What should I do next?” This makes the system feel more interactive and helpful, especially for users who do not have farming experience or technical knowledge.

# LITERATURE REVIEW

## Introduction To Plant Health Detection

Agriculture plays a crucial role in order to ensure security of food as well as supporting rural economies, especially in developing nations. This is essential as plant diseases continue to affect crop yields and food availability despite technological progress. For example, disease detection depends on human inspection or expert consultation in tradition, which is labour-intensive, subjective, and often inaccessible in remote areas. Therefore, researchers have developed automated techniques using machine learning, deep learning, and computer vision in response to improve accuracy, consistency, along with speed. Consequently, these methods enable earlier and more reliable detection, proving more efficient than manual observation.

Similarly, deep learning has become central in plant disease detection systems. This is because among the most used architectures are the You Only Look Once (YOLO) series for object detection and CNNs like ResNet50 and VGG16 for image classification. To illustrate, CNNs were successfully applied by [4] to detect rice leaf diseases through a mobile app, achieving 93% accuracy. [5] demonstrated YOLOv8’s effectiveness in weed detection, reaching 86% accuracy. Furthermore, YOLOv5, YOLOv8, and YOLOv9-C were compared by [6], reporting mAP@50 of 46% for YOLOv9-C. This supports its use for real-time detection. Thus, these results justify the use of YOLOv9s and ResNet50 in this project, offering a good balance between speed and accuracy.

Beyond visual detection, chatbots have emerged as effective tools in providing real-time guidance to farmers. For example, 87% accuracy was reported by [7] in answering crop-related queries. This demonstrates the chatbot’s potential for early diagnosis. Similarly, [8] highlighted how AI chatbots offer relevant advice on pesticide use and nutrient management, with a 91% relevance score. Moreover, NLP integration allows these chatbots to process natural language [3]. Additionally, [9] demonstrated that LLMs could interpret symptoms and return context-aware responses. These capabilities make chatbots ideal for scalable and accessible support in rural agriculture.

Although both technologies are effective independently, but their integration is still rare. As a case in point, [10] proposed a hybrid system using computer vision and image segmentation to assess plant damage. Also, [9] combined YOLOv8 with a RAG-based chatbot. It improves accuracy from 82% to 91% while reducing hallucinated responses. As a result, this integration allows real-time diagnosis, classification, and recommendation within a single system, which not only improves autonomy but also accessibility for farmers, particularly in areas with limited agricultural support.

## Current Challenges

Despite promising progress, several challenges remain. First, many deep learning models struggle to generalise across varied conditions. As [11] noted, environmental changes in lighting or leaf texture can cause accuracy to drop by 15–20%. Second, scalability remains a concern. This is due to the fact that most systems are designed for specific crops or regions. Additionally, [2] reported that deploying AI tools at scale requires strong infrastructure and diverse datasets. Lastly, the lack of labelled data limits model training, especially in underdeveloped areas where data collection is costly and slow. These issues affect both vision-based systems and language models in agriculture. Table 1 presents a summary of these reviews, outlining their methods, key results, and contributions, which together justify the project’s methodological direction and necessity.

**TABLE 1.** Summary of literature review papers

|  |  |  |  |
| --- | --- | --- | --- |
| **Author(s)** | **Method** | **Key Results** | **Contributions** |
| [2] | CNN | 93% accuracy | Confirms CNN reliability in detection |
| [3] | YOLOv8 + (RAG) | accuracy increased 82 to 91% | Proves RAG boosts chatbot accuracy |
| [4] | YOLOv8 | 86% accuracy | Proves YOLOv8 suitability in agriculture |
| [5] | YOLOv5/8/9-C | mAP@50: 46% | Endorses YOLOv9s as optimal choice |
| [6] | Chatbot | 87% query accuracy | Show chatbot effectiveness in diagnosis |
| [7] | Chatbot | 91% advice relevance | Supports chatbot use for treatment aid |
| [8] | AI Advisor | 85% user satisfaction | Validates NLP tools in rural support |
| [9] | CV + Segmentation | 90% IoU accuracy | Confirms value of hybrid CV approaches |
| [10] | CNNs/VGG16 | 15–20% accuracy drop | Highlights need for robust datasets |
| [11] | IoT + Chatbot | Infra and data limitations | Emphasises real-world deployment limits |
|  |  |  |  |

Several platforms assist farmers with disease detection and farm management as shown in Table 2. First, Plantix uses image recognition to diagnose over 400 plant diseases and offers instant advice. Next, PlantVillage combines AI and SMS to support disease, pest, and climate issues in remote areas. Lastly, CropX focuses on soil monitoring through sensors and analytics to help optimise water use and crop yield. Nevertheless, none combines real-time detection, classification, and AI-driven conversation as in the proposed system.

**TABLE 2.** Relevant application comparison

|  |  |  |  |
| --- | --- | --- | --- |
| **Platform** | **Focus** | **Tech Used** | **Unique Feature** |
| Plantix | Disease diagnosis | Image recognition, AI | 400+ diseases, real-time feedback |
| PlantVillage | Crop & climate support | AI, cloud, SMS | AI advice via SMS/mobile, climate focus |
| CropX | Precision soil management | Soil sensors, analytics | Real-time soil & water data |

# Proposed Framework AND Methodology

The proposed system is designed as a two-part solution as shown in Figures 1(a) and (b) below in that integrates computer vision, deep learning, and generative AI. It operates in two primary modes: real-time monitoring through a webcam and an interactive chatbot. In both cases, plant leaves are first detected using the YOLOv9s object detection model. Once detected, DeepSORT assigns a unique ID to each leaf to maintain consistency across frames. The system then classifies each detected plant using the ResNet50 model, which predicts one of sixteen plant disease classes. Following classification, the result is passed to a prescription module. In the case of real-time monitoring, this module uses a CSV file to store results, which are then processed using LangChain's Retrieval-Augmented Generation (RAG) pipeline powered by Llama 3.2b. The CSV Agent serves as an interface that queries the CSV file for relevant disease classifications and feeds this structured data into Llama 3.2b to generate immediate treatment suggestions. Meanwhile, the user can upload a plant image or ask a question for chatbot interactions, which triggers a similar detection and classification process, followed by an AI response powered by Gemma3. This chatbot operates independently of any CSV data or RAG mechanisms.

|  |  |
| --- | --- |
| **A diagram of a plant health monitoring  AI-generated content may be incorrect.** | A diagram of a chatbot  AI-generated content may be incorrect. |
| (a) | (b) |

**Figure 1.** (a) Real-time plant health monitoring block diagram; (b) Farmiz chatbot system block diagram

This modular approach allows both real-time and user-triggered analysis which ensures accessibility and flexibility. Furthermore, it also enables the system to deliver expert-level advice in environments where agronomists are not readily available.

The development process followed Agile methodology due to its iterative and feedback-driven structure. The reason is because the system was built over several sprints, and each focused on different modules such as model training, system integration, backend development, and chatbot enhancement. Also, each sprint ended with testing to ensure component stability and allowed quick refinement based on real-world performance. Therefore, this approach was especially useful for integrating AI components like the LLMs and managing model-API communication efficiently.

Next, the backend was developed using FastAPI due to its lightweight and high-performance support for asynchronous operations. This is essential for real-time image processing and API communication. The frontend was built with Next.js due to its React-based structure and responsive interface. Furthermore, model training was conducted in Google Colab to take advantage of GPU resources, while Roboflow was used for 3200 images dataset annotation, data augmentation and format conversion. Meanwhile, PyTorch and Ultralytics were used to implement YOLOv9s and ResNet50 models. Afterward, LangChain and Ollama were integrated for AI generation, including prescription writing and chatbot conversation. Last but not least, Anaconda was used to maintain a stable virtual environment and manage dependencies efficiently.

However, the early prototype began with a dataset of 1,200 plant images. At this stage, only object detection was implemented using YOLOv9s to test accuracy and responsiveness. There was no classification model integrated yet. The chatbot was tested independently using LangFlow to validate prompt flow and simulate AI interaction before actual deployment. Likewise, the user interface was initially designed in Figma. This helps to visualise key components like webcam feed, chatbot interface, and result display layout. Over time, ResNet50 [12] was added for classification, DeepSORT was included for tracking, and the LangChain RAG pipeline was introduced for generating treatment advice. Each feature was tested as an individual module before full system integration.

# Experimental Results and Discussion

Initially, a prototype of the detection pipeline was tested using a small set of 1,200 images. This early phase focused solely on verifying detection accuracy using YOLOv9s. After validating its feasibility, the project advanced to a larger dataset derived from the publicly available New Plant Diseases Dataset on Kaggle [13]. From this extensive source, 3,200 images were carefully selected and imported into Roboflow for annotation. During this process, bounding boxes were applied around whole plant leaves, rather than individual disease spots. This decision aligns with the system’s architecture, where YOLOv9s is responsible for identifying the type of crop leaf, and ResNet50 is used later to classify the detected leaf into a disease category.

Additionally, the 3,200 annotated images were also augmented using typical techniques including horizontal and vertical flipping, rotation between −15° and +15°, and ±15% changes in brightness to make the detector YOLOv9s more robust. The augmentations increased the dataset to 5,440 images. The images were then resized all to 640×640 pixels to give them uniformity. Moreover, roboflow also managed the division of the dataset, utilizing 82% for training (4,480 images), 12% for validation (640 images), and 6% for testing (320 images). Finally, YOLOv9s was trained to detect six various types of crops: apple, bell pepper, crop (miscellaneous), grape, strawberry, and tomato.

Once the plant leaves were detected in each image, their cropped regions were passed into a ResNet50 model to classify disease. In contrast, ResNet50 was trained solely on the original 3,200 leaf images, with no additional augmentation unlike YOLOv9s which focused on spatial detection. This classification dataset was split using a stratified approach: 60% for training, 20% for validation, and 20% for testing. The model was trained to predict one of sixteen disease classes across multiple crop types. Specifically, these were apple black rot, apple scab, apple healthy; corn common rust, corn northern blight, corn healthy; grape black rot, grape esca, grape healthy; pepper bacterial spot, pepper healthy; strawberry leaf scorch, strawberry healthy; and tomato bacterial spot, tomato septoria leaf spot, and tomato healthy.

Finally, this two-model architecture improves system performance and interpretability. In consequence, the system separates the task of identifying what crop is present from determining what condition it is in by assigning distinct roles to YOLOv9s and ResNet50.Furthermore, this layer wise pipeline also makes it easier to develop user-friendly output, allowing non-technical users to make informed decisions without needing to understand how the model operates internally.

Following Table 3, Model training was conducted in Google Colab using PyTorch with GPU support. All models applied early stopping and cross-entropy loss. For instance, CNN used the Adam optimiser, VGG16 used SGD with momentum and decay, while ResNet50 was fine-tuned by freezing base layers and scheduling learning rate decay. As a result, ResNet50 achieved the highest validation accuracy, justifying its selection as the final classifier.

**TABLE 3.** Deep learning model comparison

|  |  |  |  |
| --- | --- | --- | --- |
| **Platform** | **Initial Accuracy** | **Fine-Tune Method** | **Final Accuracy** |
| CNN | 91.72% | Adam, CrossEntropyLoss | 94.06% |
| VGG16 | 89.69% | SGD + StepLR, CrossEntropyLoss | 94.22% |
| ResNet50 | 94.22% | Adam + StepLR, CrossEntropyLoss | 99.53% |

DeepSORT was implemented using a Kalman filter and cosine distance metric to enable consistent tracking across video frames. As defined in trackers.py, each object detected by YOLOv9s is passed to DeepSORT, which assigns a unique tracking ID based on motion prediction and appearance features. This prevents duplicate classifications for the same leaf across frames, even under partial occlusion. The detection.py script manages this real-time pipeline, handling frame input, performing detection with YOLOv9s, and forwarding bounding boxes to the DeepSORT tracker, which outputs object IDs and coordinates.

Moreover, the chatbot, Farmiz, is powered by the Gemma3 language model served via Ollama and orchestrated using LangChain. According to chatbot\_session.py and farmiz\_chain.py, it handles both image uploads and free-text queries. When an image is submitted, it follows the same pipeline which is detected by YOLOv9s, tracked with DeepSORT, and classified using ResNet50. The result is immediately passed to Gemma3 for a direct, natural language reply. For text-only queries, the chatbot processes intent and generates answers using its embedded prompt context without relying on any external data or CSV storage. This makes Farmiz fast, context-aware, and efficient for real-time support.

In addition, the web application uses Next.js for the frontend and FastAPI for the backend. The reason is because they are supporting asynchronous and modular communication. The interface consists of a real-time monitoring module and a chatbot page (index.js, realtime-monitoring.js, and farmiz-chatbot.js). These interact with backend scripts like main.py, realtime\_detector.py, and realtime\_classifier.py, which process webcam frames or uploaded images. For webcam input, classification results are written to a CSV file (csv\_handler.py) and retrieved using LangChain’s CSV agent to generate treatment responses via Llama 3.2b. In contrast, chatbot responses are triggered separately and processed through Ollama. This design ensures both components operate independently while remaining integrated within a single system.

Figure 2(a) as depicted above illustrates the Farmiz chatbot interface, highlighting how the system processes an uploaded plant image to perform object detection and disease classification. In this example, the leaf is correctly identified as "grape\_leaf" and classified with the associated disease, "grape\_esca", with annotations clearly displayed on the image. Meanwhile, Figure 2(b) demonstrates Farmiz's ability to deliver context-aware responses by providing a detailed diagnosis along with a structured treatment plan tailored to the identified disease.

|  |  |
| --- | --- |
| A screenshot of a computer  AI-generated content may be incorrect. | A screenshot of a chat  AI-generated content may be incorrect. |
| (a) | (b) |

**Figure 2.** (a) Farmiz chatbot plant disease detection; (b) Farmiz chatbot’s diagnosis and treatment plan

Likewise, Figure 3 presents the real-time monitoring interface of the system. To illustrate, the YOLOv9s detects and annotates multiple plant leaves within a single video frame. Moreover, the right-hand panel displays the classification output, including the plant type, disease category, detection confidence, and the recommended treatment prescription. These prescriptions are dynamically generated by the CSV Agent through the “langchain\_experimental” RAG pipeline driven by Llama 3.2b.

A screenshot of a computer

AI-generated content may be incorrect.

**Figure 3.** Real-time detection with yolo and prescription output

# Conclusion

During system development, several integration challenges were encountered. One of the primary issues was ensuring smooth real-time performance across modules with varying processing demands. In some cases, asynchronous operations between YOLOv9s detection and DeepSORT tracking led to mismatched frames, requiring fine-tuning of the pipeline. Additionally, formatting and storing classification results for the real-time prescription pipeline introduced timing constraints, as the system needed to ensure that results were available before the CSV Agent could access them. On the frontend, integrating both webcam monitoring and chatbot features within a unified interface required multiple design revisions to maintain responsiveness and user-friendliness.

Nevertheless, the current system has several limitations despite its effectiveness. First, the chatbot lacks long-term memory. This means it cannot store or recall previous conversations once a session ends. Subsequently, this restricts continuity and historical context in user interactions. Secondly, the system supports only 16 predefined disease classes, limiting its applicability across other crops or less common diseases. Lastly, the absence of a dedicated backend database for logging or analytics reduces the potential for long-term data insights or user behaviour tracking.

Besides, future iterations of the system could integrate a lightweight database or cloud storage to retain user data, plant images, and chatbot conversations. This would enable historical query tracking and personalized interactions, enhancing system memory and user experience. Likewise, expanding the dataset to cover more crops and diseases could improve model generalization. Meanwhile, adding voice-based and multilingual capabilities could extend accessibility for diverse farming communities.

# References

1. P. C., V. B., R. K., and B. S., “Enhancing Citrus Plant Health through the Application of Image Processing Techniques for Disease Detection,” *Journal of Informatics and Web Engineering* 4(2), 53–63 (2025).
2. T. Maginga, J. Nsenga, P. Bakunzibake, and E. Masabo, “Smallholder farmer-centric integration of IoT and Chatbot for early Maize diseases detection and management in pre-visual symptoms phase,” in *2022 IEEE Global Humanitarian Technology Conference (GHTC)*, (IEEE, Santa Clara, CA, USA, 2022), pp. 369–372.
3. J. Omara, E. Talavera, D. Otim, D. Turcza, E. Ofumbi, and G. Owomugisha, “A field-based recommender system for crop disease detection using machine learning,” Front. Artif. Intell. **6**, 1010804 (2023).
4. B. Sharmah, P. P. Phukan, and M. Gogoi, “Design of a Chatbot Enabled Mobile Application for Real-Time Rice Leaf Disease Identification Using Deep Learning Framework,” in *2024 15th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, (IEEE, Kamand, India, 2024), pp. 1–6.
5. P. Kumar, and U. Misra, “Deep Learning for Weed Detection: Exploring YOLO V8 Algorithm’s Performance in Agricultural Environments,” in *2024 2nd International Conference on Disruptive Technologies (ICDT)*, (IEEE, Greater Noida, India, 2024), pp. 255–258.
6. Y. Natij, H. El Karch, A. Maafiri, and A. Mezouari, “Evaluating the Performance of YOLO Object Detectors for Plant Disease Detection,” in *2024 11th International Conference on Wireless Networks and Mobile Communications (WINCOM)*, (IEEE, Leeds, United Kingdom, 2024), pp. 1–6.
7. S. R, V. V. A, and S.R. T, “A Chatbot Assistant for Early Detection and Management of Sugarcane Diseases,” in *2024 Second International Conference on Advances in Information Technology (ICAIT)*, (IEEE, Chikkamagaluru, Karnataka, India, 2024), pp. 1–5.
8. C. Bhuvaneswari, H.S. Pokhariya, P. Yarde, V. Vekariya, H. Patil, and N. L, “Implementing AI-Powered Chatbots in Agriculture for Optimization and Efficiency,” in *2024 2nd International Conference on Intelligent Data Communication Technologies and Internet of Things (IDCIoT)*, (IEEE, Bengaluru, India, 2024), pp. 1–7.
9. S. Selva Kumar, A. K. M. A. Khan, I. A. Banday, M. Gada, and V. V. Shanbhag, “Overcoming LLM Challenges using RAG-Driven Precision in Coffee Leaf Disease Remediation,” in *2024 International Conference on Emerging Technologies in Computer Science for Interdisciplinary Applications (ICETCS)*, (IEEE, Bengaluru, India, 2024), pp. 1–6.
10. M. J. Montagut Marques, L. Mingxin, K. T. Shiojiri, T. Hagiwara, K. Hirose, K. Shiojiri, and S. Umezu, “Plant Doctor: A hybrid machine learning and image segmentation software to quantify plant damage in video footage,” Measurement 249, 117094 (2025).
11. G. S. Damaraju, S. Siddharth Cilamkoti, L. Venugopal, V. Warrier, D. S Ashok, G. G, P. Seethapathy, and S. Manalil, “PlantHealthAI: An Integrated System for Plant Disease Detection, Severity Prediction with Knowledgebase Chatbot Support,” in *2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, (IEEE, Bangalore, India, 2023), pp. 1–7.
12. J. -R. Lee, K. -W. Ng, and Y. -J. Yoong, “Face and Facial Expressions Recognition System for Blind People Using ResNet50 Architecture and CNN,” *Journal of Informatics and Web Engineering* **2**(2), 284–298 (2023).
13. S. Bhattarai, “New Plant Diseases Dataset” (Kaggle, 2018). Available at: https://www.kaggle.com/datasets/vipoooool/new-plant-diseases-dataset