ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov1153

Plant Disease Detection and Intelligent Agricultural Advisory System

Likhith P.[1]; M. J. Tejas Gowda[2]; Pradeep Patel P. K.[3]; Praveen Kumar R.[4]

[1,2,3,4]Department of Computer Science and Engineering, East West Institute of Technology, Bengaluru, Bengaluru, India

Publication Date: 2026/11/25

Abstract: Plant diseases reduce crop quality and productivity, while timely diagnosis may be difficult when growers have limited access to plant pathology expertise. This paper presents the design of a Plant Disease Detection and Intelligent Agricultural Advisory System that combines image-based disease classification with a structured agronomic knowledge module. The proposed workflow accepts a leaf photograph through a Streamlit interface, prepares the image using a computer-vision preprocessing stage, and passes the resulting tensor to an EfficientNet-based deep-learning classifier. The classifier returns a ranked disease label and associated confidence score. The predicted class is then used as a key to retrieve crop-specific information from a Python advisory module, which organizes the response into eight parts: host crop, disease identity, health status, description, visible symptoms, organic management options, chemical-management guidance, and preventive practices. The interface presents the uploaded specimen, prediction information, and advisory content in a form intended for farmers, students, and agricultural extension users. The paper details the system architecture, image-processing pipeline, transfer-learning rationale, mathematical formulation, module responsibilities, and evaluation protocol. Published PlantVillage research demonstrates the potential of deep learning on controlled leaf images, while later work highlights the difficulty of transferring such performance to natural field conditions. Because the project materials available for this manuscript do not include verifiable training logs, a documented test split, or reproducible timing measurements, no project-specific accuracy or latency values are claimed. The contribution is therefore presented as an integrated prototype design and implementation workflow, together with a transparent evaluation plan and safety considerations for agricultural recommendations.

Keywords: *Plant Disease Detection, EfficientNet, Deep Learning, Computer Vision, Transfer Learning, Streamlit, Agricultural Advisory, Image Classification.*

How to Cite: Likhith P.; M. J. Tejas Gowda; Pradeep Patel P. K.; Praveen Kumar R. (2026) Plant Disease Detection and Intelligent Agricultural Advisory System. *International Journal of Innovative Science and Research Technology*, 11(11), 1623-1631. https://doi.org/10.38124/ijisrt/26nov1153

I. INTRODUCTION

Agriculture depends on crop health at every point in the growing process. Diseases caused by fungi, bacteria, viruses and other pathogens can harm leaves, stems, fruits and roots. Often, changes in the leaves show that a plant is under stress.. Spotting these signs is not always easy. Many diseases cause symptoms—spots, color changes, leaf curling or dead areas.. How these symptoms appear can vary depending on the crop type, the stage of the disease, the weather and how the farm is managed.

Traditional diagnosis usually needs help from an expert agronomist, lab testing or advice from a farming extension service. These are tools.. They may not be quickly available to every farmer. A mobile or web-based image analysis tool can give a fast screening result. Help guide what to do next. This tool should not be seen as a replacement for professional diagnosis. Instead, it should support the decision-making process by making early information more accessible.

Deep convolutional neural networks can learn features directly from labeled images. In plant pathology, these features might include the shape of lesions, how colors are spread, surface texture, yellowing edges around damaged areas, rust-like bumps or patterns of how the disease spreads. Transfer learning helps by using a model trained on a general image dataset and adapting it to a smaller agricultural dataset. EfficientNet is especially useful because it balances the depth, width and input resolution of the network through a method called compound scaling [1].

The system we propose combines this classification power with a rule-based system. Of just showing a predicted disease label, the application turns that label into a clear, structured report. This design helps bridge the gap between image classification and practical management advice. The system is built from four parts: the Streamlit user interface, image preprocessing, the EfficientNet inference engine and a knowledge base that gives advice.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov1153

➢ *Scope*

The project is about classifying leaf images and providing guidance based on the results. It includes taking in image files, turning those images into a format the model can use, getting a predicted class and confidence score and then showing an eight-part report. The project does not cover pesticide use, lab testing to confirm diseases, measuring how severe a disease is or making field-level forecasts that have been proven accurate.

➢ *Contributions*

What we bring to the table is joining a deep learning prediction step with a workflow, all in one app. The paper walks through the data path from start to finish. It covers what each software module is responsible for, the math behind the classification and the checks you need for evaluation. The paper also points out the spots that need fixing before anyone uses this system as a decision-support service for farming.

II. LITERATURE WORK

Hughes and Salathé created a collection of plant-health images to help with mobile disease diagnostics [2]. Mohanty, Hughes and Salathé then tested networks on 54,306 PlantVillage images that covered 14 crop species and 38 crop-disease or healthy classes [3]. Hughes and Salathés experiment with controlled images showed 99.35% accuracy. Hughes and Salathés tests with images from sources had much lower accuracy. This difference matters a lot: doing well on a controlled dataset does not mean the same level of success will happen with photos taken in farms.

Singh et al. Created PlantDoc, a dataset meant to show real-life conditions with 2,598 labeled images across 13 plant species and up to 17 disease classes [4]. The dataset shows how hard it is to collect field images with lighting, messy backgrounds, things blocking the view and different angles. These problems make it important to test carefully, not to split images randomly.

Recent studies of plant-disease datasets point out class imbalance, changes in label quality and new classes as big challenges for real-world use [5]. The current project takes this path by looking at confidence display and field testing as checks, not assuming that one prediction is always right.

A. Problem Statement

A person can look at a leaf. Notice its colour or texture, but the person may not have quick access to a plant pathology specialist. Current image-classification demos can give a disease label,. They do not explain what that label means or how a grower should use that information. On the hand, generic advisory material may not match the exact crop-disease category that the image model has identified. The problem addressed in this work is to link classification with a crop-aware advisory presentation while keeping the model's certainty limits visible.

The system receives a leaf image. Predicts one class from the set of classes that the training labels represent. The predicted class is then mapped to a record. The key engineering challenge is to keep the model's class-index ordering, the class names and the keys in the module in sync. A mismatch at this interface can cause the system to show a crop or disease description even if the model prediction itself is correct.

➢ *Existing Approach and Identified Gaps*

Manual visual diagnosis relies on experience. Manual visual diagnosis can be delayed by travel, high workload or limited availability of specialists. A standalone classifier adds automation,. A standalone classifier can produce results that are difficult for non-technical users to interpret. A static information page gives management guidance,. A static information page does not adjust its content to match the leaf that the user has uploaded. The proposed workflow links these functions while still requiring a review step.

A second gap concerns dataset realism. PlantVillage offers a benchmark for controlled leaf images, whereas PlantDoc and later studies reveal the complexity of images. A classifier that is trained on backgrounds may notice background or capture-condition cues in addition to disease symptoms. Therefore the system must be tested with images that reflect the environment where the system will be used.

➢ *Proposed System Overview*

The proposed application follows a step-by-step workflow. First, the user uploads a leaf image using the Streamlit interface. Second, the image is converted into the color format that the model expects. Third, the image is resized to the configured input dimensions. Turned into a numerical tensor. Fourth, the trained EfficientNet-based model produces class scores. Fifth, the class with the score is selected and its confidence is calculated. Sixth, the class label is used to fetch an entry. Finally, the interface displays the image, the predicted crop-disease class, the confidence value and the advisory report.

The project notes mention a 224 × 224 input size and a saved model file named plant_disease_model.h5. These details are provided as implementation references. The final input dimensions and preprocessing scale must be confirmed against the training script and the saved model before deployment. The model variant is described generically as EfficientNet because the project notes refer to both B0 and B3 in places.

➢ *Objectives*

- Build a way to upload pictures of leaves.
- Get the images ready the way every time for the neural network to read.
- Use a model that sorts plant diseases into many classes.
- Show a confidence score beside the class it predicts.
- Link the class labels to an advisory record with eight parts.
- Display the diagnosis and management details on a Streamlit dashboard.
- Set up a repeatable test process that checks performance for each class, how long it takes and how well it holds up in the field.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov1153

➢ *Functional Requirements*

Table 1 Functional Requirements

ID	Requirement	Expected behaviour
FR-1	Image upload	Accept supported JPG/JPEG/PNG leaf images. Reject unsupported formats.
FR-2	Preprocessing	Decode images, convert them to RGB, resize them and transform image data consistently.
FR-3	Inference	Load the trained model. Return a probability score for each supported class.
FR-4	Advisory lookup	Map the predicted class to a structured advisory entry.
FR-5	Result display	Display the image, the predicted label, the confidence and the advisory sections.
FR-6	Error handling	Deal with missing model files, invalid images and unmatched advisory keys.

III. PROPOSED METHODOLOGY

The methodology is split into parts: image acquisition, data preparation, model inference, confidence handling, advisory retrieval and interface presentation. Breaking it down this way makes it easier to test each part. It also helps in finding errors because you can look at where one module ends and another begins.

➢ *Image Acquisition and Dataset Preparation*

The input image may be taken with a smartphone. Chosen from local storage. For model development, each training image must be given a label that shows a crop-disease pair or a healthy class. The dataset should be inspected for files, broken images, labels that do not match, an uneven spread of classes and images that do not contain a usable leaf. Training, validation and test groups should be made before any augmentation. If several photos come from the plant, field or original image, related samples should stay in one group to prevent data leakage.

The project notes a PlantVillage-style dataset and many crop types. It does not give a clear list of the exact dataset used in the final training. PlantVillage's published 54,306-image, 38-class version is a known benchmark [3],. It does not prove the exact size or makeup of this project's local set. A final report should add where the dataset came from, the class list, how images each class has, how the data is split and the license details after checking the real folders.

➢ *Preprocessing and Augmentation*

The preprocessing stage takes the file and turns it into something the model can use. First, it decodes the image. Makes sure it can be read properly. Then, it changes the image to RGB format, which's standard for most models. Next, it resizes the image to match the size the model expects. After that, it. Normalizes the pixel values the same way they were during training. This helps the model understand the image better.

During training only, we can add some variation to help the model learn better. We do this by rotating the image a little, flipping it left to right or top to bottom.. Only if it makes sense for the type of image we're working with. We also zoom in or out, move the image slightly and change the brightness.. We have to be careful. The changes should not mess up the disease label. We also avoid any changes that might make the image look fake or create symptoms that don't exist. The goal is to help the model learn from natural-looking examples.

➢ *Model Training and Inference*

The EfficientNet-family backbone extracts hierarchical features from the leaf image. Early convolutional blocks of the backbone learn low-level edges and color transitions. Deeper blocks of the backbone represent more complex patterns. A task-specific classification head maps the extracted representation to the supported crop-disease labels. Transfer learning reduces the need to train all network parameters from initialization. The trained weights of the backbone are referenced in the project notes as model/plant_disease_model.h5. At inference time, the application loads the model once. Reuses the model for subsequent requests. The input tensor is passed through the backbone in evaluation mode. The output vector from the backbone is converted to a predicted class and confidence score. The class-index-to-label mapping must be stored with the model or versioned alongside the model; relying on a sorted directory listing can silently change class ordering.

➢ *Confidence and Unknown-Input Handling*

A confidence threshold can be used to separate high-scoring predictions from outputs. The confidence threshold must be selected using validation data. Reported with its trade-off between coverage and error. A low score should lead to a message such as "unable to determine reliably" rather than a forced disease label. The system should also detect crop types, non-leaf photographs, severe blur and images whose visual content differs substantially from the training data.

➢ *Advisory Mapping*

The predicted label is sent to the module to be used as a lookup key. Exact key matching is better because it stops a crop or disease record from being returned. If there is no matching entry, the system should show an explicit 'advisory unavailable' message of creating a treatment recommendation from a guessed label. The knowledge base must be version-controlled. Must be reviewed by a qualified agricultural specialist.

➢ *Interface and Output*

The Streamlit interface is made to show the specimen next to the prediction. The result panel can show the predicted crop, the disease name, the confidence score and the status. Different tabs or expandable sections can keep the content tidy so that the user can read the diagnosis first, then look at symptoms, management and prevention. The interface should also give a limitation notice that explains the image screening aid.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov1153

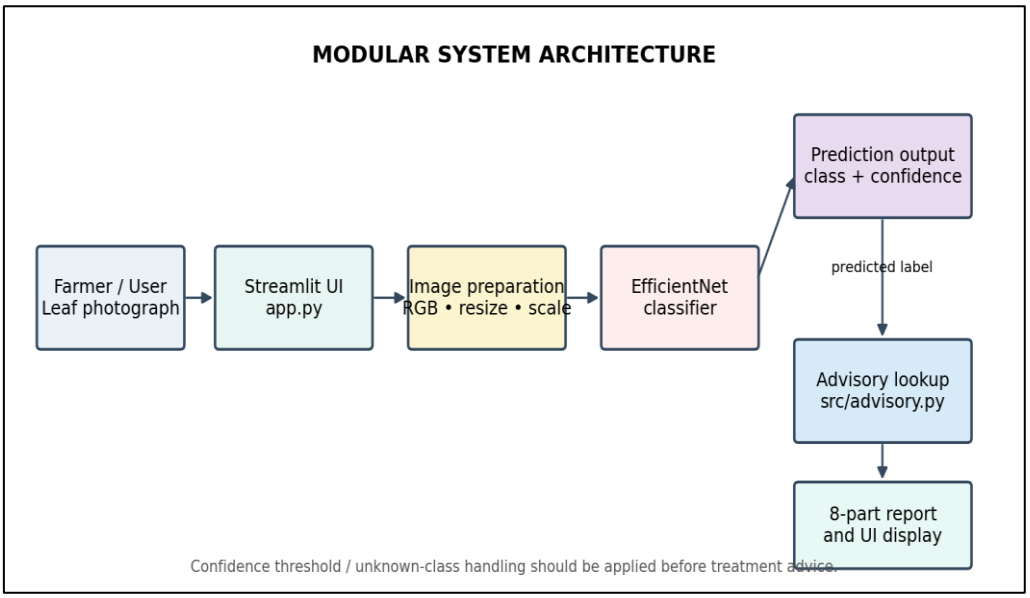


Fig 1 High-Level Architecture of the Proposed Plant Disease Detection and Advisory Workflow.

IV. SYSTEM ARCHITECTURE AND MODULE DESIGN

The architecture keeps the presentation, image processing, inference and advisory knowledge separate. This separation lowers coupling. A change in the dashboard does not need a rewrite of the classifier. Advisory content can be updated without retraining the network. Each module connects using inputs and outputs.

➢ *Presentation Layer: App.py*

The Streamlit application is the point that end users see. The Streamlit application sets up the page layout, shows project details, allows users to upload JPG, JPEG or PNG files and then shows the chosen image. After the image appears, the Streamlit application starts the prediction workflow. Model loading should be cached, so the model weights are not read again each time a new image is uploaded. The Streamlit application must also show states: loading, success, invalid input, missing model files and advisory lookup failures.

➢ *Preprocessing Layer: Utils/Preprocessing.py*

The preprocessing module handles image decoding, changes colors, resizes images in space and turns them into numbers. It needs to have a repeatable function that takes an image and gives back a tensor with the exact shape, color channel order and number range the model expects. Putting preprocessing in its module lets us test tricky cases one by one. For example, we can check how it deals with black-and-white images, images with transparency layers, odd shapes or broken uploads. This helps catch problems and keeps things reliable.

➢ *Inference Layer: Model.py*

The model directory holds the serialized trained network, which the project notes call plant_disease_model.h5. The inference layer loads the model, checks that the tensor shape is correct, runs the prediction and gives back an object that contains the class index, class label and confidence score. The model metadata must record the architecture variant, input dimensions, class order, framework version, training date and dataset identifier.

➢ *Advisory Layer: src/Advisory.py*

The advisory module holds information for every supported class. The advisory module must give a schema, even when optional fields are missing. The advisory module needs matching between predicted labels and advisory keys. A fallback parser can help show a class name,. The advisory module must not use it to invent a treatment protocol. The advisory module should mark any entries clearly.

➢ *Module Interaction*

Table 2 Module Interactions

Module	Input	Output / responsibility
app.py	Uploaded image	User interaction, validation, status and result rendering.
preprocessing.py	Image bytes or image object	RGB tensor at the model's expected dimensions.
model	Preprocessed tensor	Class scores and predicted class.
class mapping	Class index	Stable human-readable crop-disease label.
advisory.py	Predicted label	Eight-part structured management record.

A. Sequence of Operation

The sequence diagram in Fig. 2 Shows a request process. The user uploads a specimen. The interface checks the file to make sure it is valid. Then it starts preprocessing the file. After that, the prepared tensor is sent to the classifier. The classifier gives back class scores. These scores are turned into a label

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov1153

and a confidence value. The interface then uses the predicted label to get the record. Finally, the system creates a response and shows it to the user. If something goes wrong—like if the image cannot be decoded, the model is not available, the prediction is too uncertain. The advisory key is missing—the process stops safely and handles the error properly.

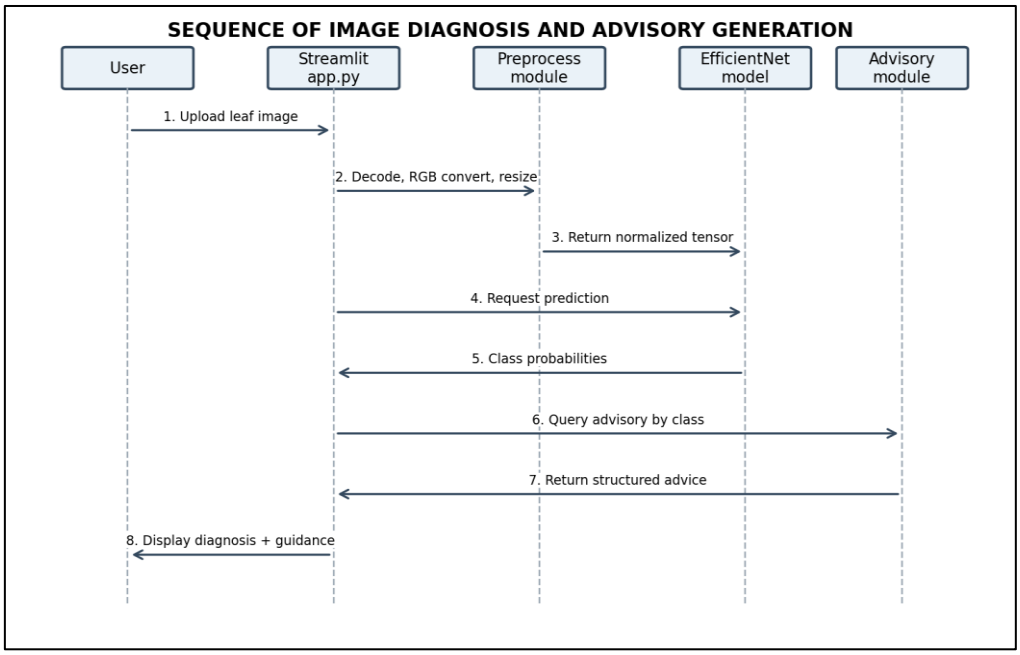


Fig 2 Sequence Diagram for Image Upload, Model Inference and Advisory Retrieval.

➢ *Data Contracts*

A practical implementation should use data contracts between modules. The preprocessing function should return a tensor and metadata describing the applied transformation. The inference function should return the predicted class, score vector and model version. The advisory function should return a record with named fields. This approach prevents the UI from depending on implementation details and makes each component easier to test.

B. Eight-Part Agricultural Advisory Framework

The advisory component is meant to turn a model label into organized information that helps a user make the next decision.

Below are eight parts that form a content framework.

For each disease, the content must be checked against the crop, the stage of the disease, the local climate and the relevant agricultural rules.

Table 3 Advisory Framework Categories

Part	Information presented	Purpose
1. Host crop	Crop or plant species associated with the label.	Confirms the crop context.
2. Disease identity	Human-readable disease or healthy-state label.	Explains the model's predicted category.
3. Health status	Healthy, diseased, or uncertain status.	Provides a concise interpretation.
4. Description	Short explanation of the disease and affected plant tissue.	Adds context beyond the class name.
5. Visible symptoms	Observable signs such as spots, lesions, discoloration or growth.	Helps the user compare the output with the specimen.
6. Organic options	Appropriate cultural or biological management options.	Presents lower-impact measures where supported.
7. Chemical guidance	Approved active ingredients and label-based use guidance, if applicable.	Supports safe consultation with local experts.
8. Prevention	Sanitation, spacing, monitoring and other preventive practices.	Encourages longer-term crop health.

➢ *Advisory Knowledge Design*

Each record should be keyed by the exact model class name. Should include the crop, disease, health state, description, symptoms organic options, chemical guidance and prevention. The database should include a source or reviewer field so that advice can be traced to an university, extension publication, product label or other trusted source. The advisory module should not treat a models confidence score as proof of disease severity; image classification and severity estimation are tasks.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov1153

➢ *Safety of Treatment Recommendations*

Chemical management depends a lot on the crop, the pathogen, the formulation, the concentration, how it's applied, the time before harvest, the gear used and local rules. Using a one-size-fits-all dosage from a code dictionary can be risky if the product or crop is different. That's why this paper doesn't suggest a grams-per-litre amount for everyone. Instead, the advisory record should only show the crop-specific instructions from the product label. If someone is unsure, they should be told to reach out to agricultural extension services for help.

V. RESULTS AND DISCUSSIONS

The project notes talk about a model-training session. They mention a saved checkpoint called model/plant_disease_model.h5. The notes also explain how the system uses an EfficientNet-based classification workflow and includes a lookup module. The materials we have for writing this manuscript don't include the original training log. We don't have the dataset manifest either. There's no class-test predictions, no confusion matrix and no controlled timing report. Because of this, the results in this section can only cover the system design as described and the evaluation requirements. We can't say anything about the numbers. There's no project accuracy, no precision, no recall, no F1-score and no inference latency to report.

➢ *Functional Behaviour to Verify*

You should test the flow by looking at each working part on its own. The image upload step needs to take the file types we support and turn away files. The preprocessing step should give back the tensor shape and value range we expect. When the model runs, it should hand back a score vector with the same length as the class list. The top predicted class needs to point to the advisory record.. At the end, the dashboard should show the uploaded image and the result, while keeping any internal errors hidden from the user.

Table 4 Functional Behavior Verification

Subsystem	Functional check	Evidence to retain
Image input	Supported formats and invalid-file handling.	Test images and validation log.
Preprocessing	Shape, channel order, pixel range.	Unit-test output and configuration.
Classifier	Correct label order and finite probabilities.	Model metadata and prediction file.
Advisory mapping	Every supported label maps to the correct record.	Coverage report and reviewed content.
Dashboard	Clear display and error states.	Screenshots and application log.

➢ *Quantitative Evaluation Protocol*

We recommend that the evaluation use a held-out test set that was not used for training, augmentation selection or threshold tuning. Accuracy should be reported together with macro-averaged precision, recall and F1-score. The report should also include class-wise metrics and a confusion matrix. For classes, macro-F1 is especially informative because macro-F1 gives each class equal weight. The report should state the number of images in each split. Explain whether images from the same plant or source were grouped to prevent leakage. Latency should be measured on a hardware and software configuration. Time for image decoding and preprocessing, model inference, advisory lookup and total response rendering should be reported separately. Repeated trials should be. A median and a spread measure such as interquartile range should be reported. A single timing measurement is not enough to establish a performance claim.

➢ *Field Generalization*

The system must be tested using photographs taken outside the training environment. Important variations to consider include sunlight, shaded areas, background vegetation, soil, overlapping leaves, blur, different camera sensors, various disease stages and multiple symptoms appearing on a single leaf. Mohanty et al. Found a difference in performance between images from the controlled PlantVillage dataset and real-world images [3]. PlantDoc was created to help with the lack of natural-scene data [4]. These findings show that we need a test set focused on field conditions, instead of just using validation accuracy to judge performance.

➢ *Interpretation*

The main practical value of the system is to merge classification with structured information. When a label is correct, it helps users find the advice in a record; if the label is wrong, it can give people bad guidance. Because of this, the advisory layer makes it more important to get class mapping right, to handle uncertainty and to carry out agronomic review. The interface must make it clear that a prediction is a first suggestion and not a final laboratory diagnosis.

VI. LIMITATIONS

The system is limited by the quality and coverage of the training dataset, the similarity of disease symptoms and possible class imbalance. And the classifier can be overconfident when an uploaded image belongs to a crop or disease. We realize that image-level classification does not identify the lesion location or estimate the percentage of affected leaf area. We can understand that image-level classification also cannot determine pathogen presence from symptoms where visual symptoms overlap with nutrient deficiency, environmental stress or multiple infections.

The advisory knowledge base requires domain review and regular updates. Recommendations may vary by crop variety, geography, season, disease pressure, product formulation and regulation. The project materials also contain references to EfficientNet-B0 and EfficientNet-B3; The exact variant should be verified from the model architecture or training script before publication. And also see that the dataset size and class count must be documented from the local dataset rather than borrowed from a published benchmark.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov1153

➢ *Future Enhancements*

Several extensions can improve reliability and usability. I believe that a larger field-image collection should be built with labels, metadata and expert review. Model calibration and unknown-class detection can reduce the risk of presenting a label as certain. Image segmentation or object localization could identify regions and support a separate severity-estimation module. A mobile or edge deployment path could reduce dependence on connectivity, provided the model is optimized and validated on target devices.

The advisory component can be expanded with language support, including Kannada and Hindi and with verified content from agricultural universities and extension agencies. Weather information may support risk alerts, but such functionality should be introduced as a separate forecasting module with its own data sources and validation. Integration with drones or IoT sensors may extend monitoring from leaf photographs to crop-level observation, although these extensions require new data, hardware and evaluation procedures.

VII. CONCLUSION

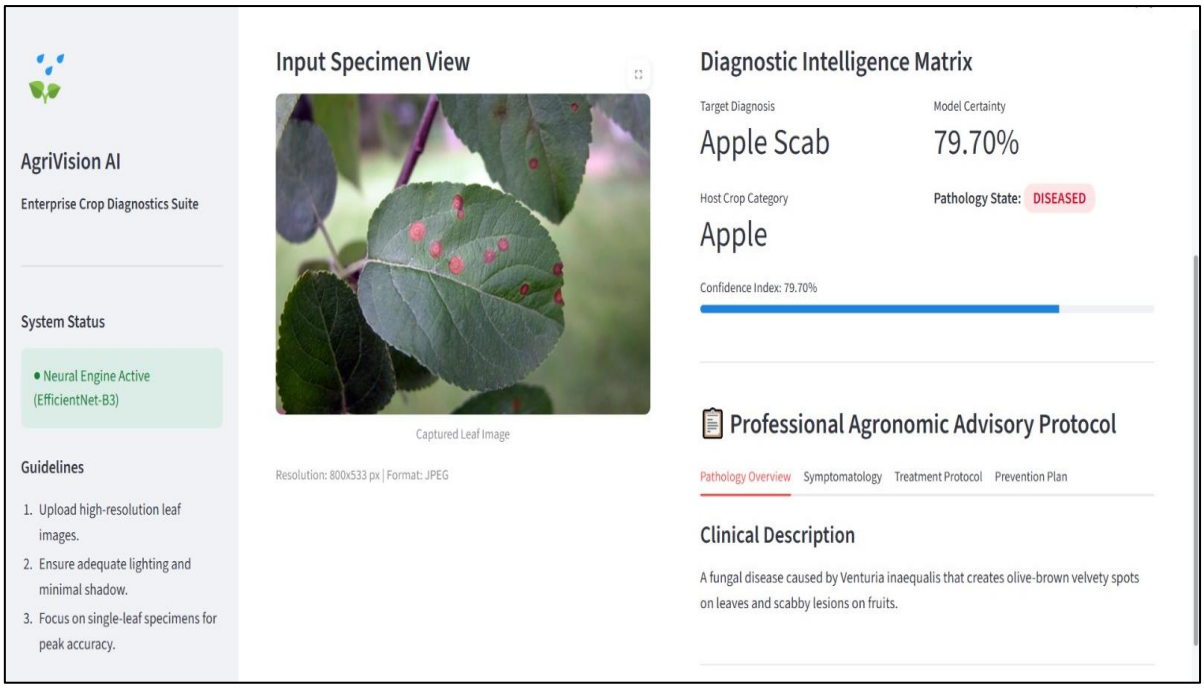


Fig 3 Diagnostic Interface View



Fig 4 System Upload Dashboard

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov1153

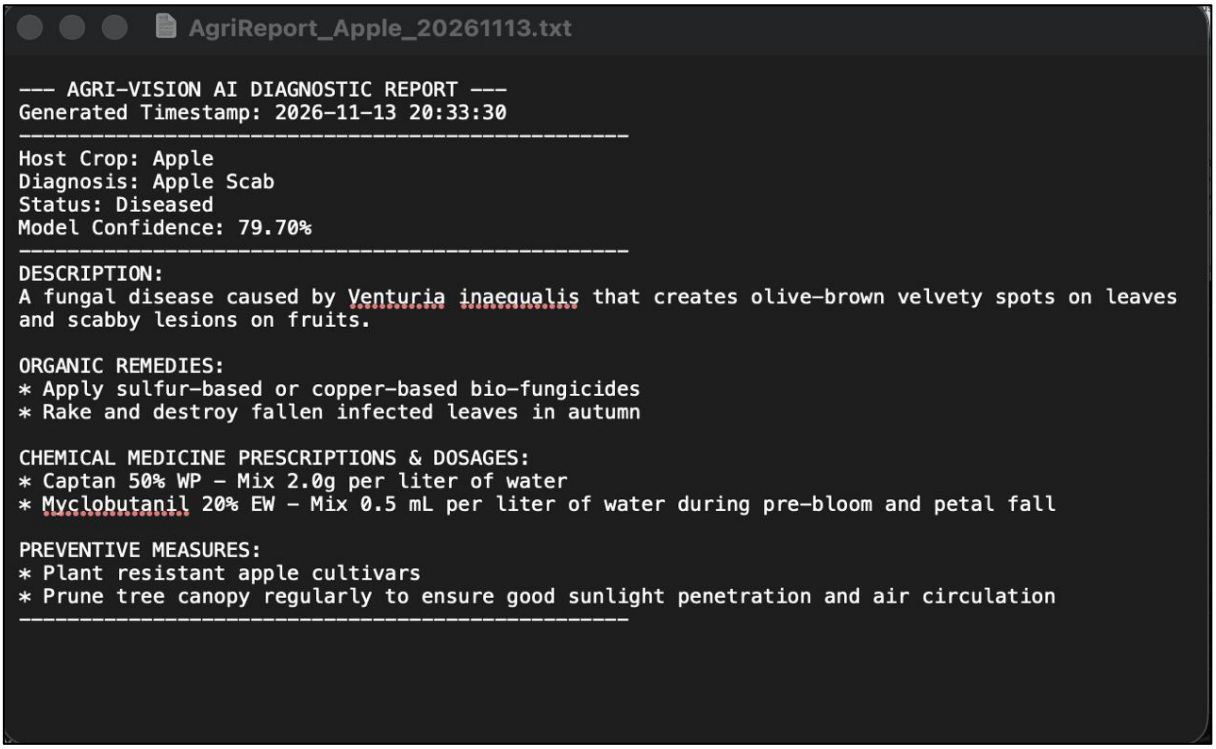


Fig 5 Generated Diagnostic Text Report

This paper presents the design of a Plant Disease Detection and Intelligent Agricultural Advisory System that links image-based classification to an eight-part agronomic information workflow. The system pairs a Streamlit interface, image preprocessing steps, an EfficientNet-family classifier, class-label mapping and a Python advisory module. The architecture is meant to make early crop-health information easier to reach while keeping the classifier and knowledge base as testable parts. From my perspective, Plant Disease Detection becomes more approachable when the system remains modular and user-friendly.

The paper also stresses that publication-quality claims must be backed by evidence. Published work shows performance on controlled leaf-image datasets, yet external and field photographs bring big challenges. Because the project materials do not include project-specific evaluation logs, no numerical performance claim is made in this document. Future work should record the dataset and EfficientNet variant, create class-wise test metrics, test field generalization, measure end-to-end latency and confirm all treatment guidance with qualified agricultural experts. In short, Plant Disease Detection will benefit from these steps to become truly reliable.

REFERENCES

[1.] M. Tan and Q. V. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” in Proc. 36th International Conference on Machine Learning, PMLR, vol. 97, pp. 6105–6114, 2019.

[2.] D. P. Hughes and M. Salathé, “An open access repository of images on plant health to enable the development of mobile disease diagnostics,” arXiv:1511.08060, 2015.

[3.] S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using Deep Learning for Image-Based Plant Disease Detection,” Frontiers in Plant Science, vol. 7, Art. no. 1419, 2016, doi: 10.3389/fpls.2016.01419.

[4.] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, “PlantDoc: A Dataset for Visual Plant Disease Detection,” in Proc. 7th ACM IKDD CoDS and 25th COMAD, pp. 249–253, 2020, doi: 10.1145/3371158.3371196.

[5.] M. Xu, H. Kim, J. Yang, A. Fuentes, Y. Meng, S. Yoon, T. Kim, and D. S. Park, “Embracing limited and imperfect training datasets: opportunities and challenges in plant disease recognition using deep learning,” Frontiers in Plant Science, vol. 14, 2023, doi: 10.3389/fpls.2023.1225409.

[6.] G. Bradski, “The OpenCV Library,” Dr. Dobb’s Journal of Software Tools, 2000.

[7.] M. Abadi et al., “TensorFlow: A system for large-scale machine learning,” in Proc. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp. 265–283, 2016.

[8.] Streamlit Inc., “Streamlit Documentation,” https://docs.streamlit.io/, accessed Nov. 2026.

[9.] Dr. Pushpa Ravikumar; Gowrav A S; Arpitha C N “A Smart Agriculture Framework for Coffee Plant Disease Detection and Remediation Using YOLOv5” 2026 International Conference on Natural Language Processing and Computer Vision (ICNPCV) 2026 10.1109/ICNPCV69434.2026.1170874

[10.] Pedapudi. Nagababu; Shaik. Nageena; Veeranki. Dharani; Darsi. Naveen “Plant Disease Detection and Diagnosis” 2024 5th International Conference for Emerging Technology (INCET) 2024 10.1109/INCET61516.2024.10593371

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26nov1153

[11.] Nitin Lokhande; Vijaya Thool; Pratap Vikhe "Comparative analysis of different plant leaf disease classification and detection using CNN" 2024 International Conference on Recent Innovation in Smart and Sustainable Technology (ICRISST) 2024 10.1109/ICRISST59181.2024.10921975

[12.] Pranav P Pathak; Thanmai S; Preetham Bhat; Akshatha Manjunath "Multi-Plant Disease and Severity Detection In Single Species Using Deep Learning" 2026 1st International Conference on AI, Data Science, Cyber Security and Smart Manufacturing for Sustainable Development (ICADCS) 2026 10.1109/ICADCS70036.2026.115835

[13.] Aadesh Shalikram Borey; Shirish Kulkarni "Deep Learning-Based Multi-Class Classification of Plant Leaf Diseases with CNN and VGG16" 2026 IEEE International Conference for Convergence in Computing Technology (I3CTCON) 2026 10.1109/I3CTCON68242.2026.11507703

[14.] Gaurav Tuteja; B. Ravindar; Mukesh Kumar; Shivashish Dhoundiyal; Saurabh; Mayank Deep Khare "EfficientNetB0-Based Multi-Class Plant Leaf Disease Detection for Scalable Precision Agriculture Applications and Monitoring" 2026 1st International Conference on Computer, Mechatronics, Robotics & Artificial Intelligence (ICCMRAI) 2026 10.1109/ICCMRAI67732.2026.11723377

[15.] Nishad Mahmud Opu; Md Saiful Islam; Ruma Akter; Kazi Maisha Jannath; Shahriar Ahmed; Rakibul Islam Rafi "Deep Learning-Based Leaf Diseases Detection in Indoor Ornamental Plants: A Comparative Study of YOLOv8, YOLOv9, and YOLOv11" 2025 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE) 2025 10.1109/WIECON-ECE69386.2025.11526313

[16.] Appana Naga Lakshmi; Bathala Nandini; S. Praveen Kumar Reddy; Rakshitha S; Noor Saba Khanam A M "A Lightweight YOLO-Based Plant Leaf Disease Detection and Embedding Space Statistical Framework for Severity Estimation" 2026 IEEE International Conference on Advances in Computing Research On Science Engineering and Technology (ACROSET) 2026 10.1109/ACROSET70960.2026.11742453

[17.] Manpreet Bhatty; Shalu Gupta "Computer Vision and Artificial Intelligence to Precision Plant Pathology: A Review of Leaf Disease Detection Systems" 2026 International Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC) 2026 10.1109/ASSIC70037.2026.11717185

[18.] Khambhampati P Chaitanya; A. Maryposonia "An Efficient Detection Transformer with Discriminative Attention-based Res-Senet Model for Early Plant Disease Detection and Classification" 2026 9th International Conference on Computing Methodologies and Communication (ICCMC) 2026 10.1109/ICCMC69250.2026.11624986

[19.] Fahim Musa Ssenyonjo; Adam A. Alli; Excellence Favor "A Machine Learning Plant Disease Detection Model: A Case of Cocoa Plant" 2026 2nd International Symposium on AI-Driven Engineering Systems (ISADES) 2026 10.1109/ISADES69945.2026.11608106

[20.] Kadava. R. N. Karthik; Bangaru Eswari; Gunji PremKumar; Gunturu Veena Livingstone; Muddapati Sai Jahnavi "Plant Leaf Disease Classification Using EfficientNet-V2B0 with Treatment Recommendation" 2026 2nd International Conference on Computing, Communication and Green Engineering (CCGE) 2026 10.1109/CCGE671422026.11581469

[21.] Seerat Singla; G Sunil; Shraddha Kaparwan; Neetu Sharma; Harmanpreet; Ajay Kumar "Plant Leaf Disease Classification Using EfficientNetV2-Based Transfer Learning on an Egyptian Plant Dataset" 2026 1st International Conference on Computer, Mechatronics, Robotics & Artificial Intelligence (ICCMRAI) 2026 10.1109/ICCMRAI67732.2026.11724230.