iJRASET
International Journal For Research in Applied Science and Engineering Technology



INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** V **Month of publication:** May 2025

DOI: https://doi.org/10.22214/ijraset.2025.69515

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

Plant Disease Detection System Using Convolutional Neural Networks and TensorFlow Lite

Ronish Raja[1], Tejas Ragupathi[2], Dr. Jebakumar R[3]
[1, 2]*Department of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur Campus Chennai, Tamil Nadu*
[3]*Associate Professor, Department of Computing Technologies, SRM Institute of Science and Technology, Kattankulathur Campus Chennai, Tamil Nadu*

Abstract: ***This research presents an intelligent system for the classification of plant diseases using a convolutional neural network (CNN) trained on a large dataset of diseased and healthy plant leaves. The model was developed using Python and deep learning libraries such as TensorFlow and Keras, achieving high accuracy in classifying various plant diseases. The trained model is integrated into a user-friendly web application using Streamlit, enabling real-time predictions from uploaded images. The system provides an accessible interface for farmers, researchers, and agricultural workers to detect plant diseases instantly, thereby aiding in timely intervention and crop management. By leveraging transfer learning and a well-curated dataset, the application ensures reliable results while remaining lightweight and easy to deploy. The project demonstrates a scalable approach to agricultural diagnostics using AI, focusing on automation, accuracy, and usability.***
Keywords: plant disease detection, convolutional neural networks, TensorFlow Lite, image classification, machine learning.

I. INTRODUCTION

Agricultural productivity is frequently threatened by plant diseases, which can substantially diminish crop output and, in extreme situations, lead to complete crop loss [1]. Detecting these diseases early and accurately allows for timely actions to be taken, which helps mitigate financial losses for farmers. Traditional methods of disease identification involve visual inspection by agricultural experts, a process that is time-consuming, labor-intensive, and prone to error, especially in large-scale agricultural operations [2].

Advancements in machine learning and deep learning technologies have paved the way for creating automated systems that can recognize diseases from plant images with high precision [3]. These systems, which employ convolutional neural networks (CNNs), can process thousands of images and detect diseases with high accuracy, offering a scalable solution for disease management [4]. However, a significant limitation of current systems is their static nature: once trained, models do not dynamically adapt to new data unless manually retrained. This makes them less effective in real-world applications where new or evolving diseases may emerge, or when new data is continuously collected [5].

To address this gap, we propose a dynamic, automated plant disease detection system that not only predicts diseases from plant images but also automates its retraining process. The system can handle unclassified or "unknown" images by transferring them to an area where they can be labeled by experts. Once a sufficient number of images have been labeled, the system triggers an automated retraining process, allowing the model to learn from the new data without manual intervention [6].

II. MAIN OBJECTIVES OF OUR PROJECT

The primary aim of this project is to detect plant diseases using deep learning techniques and display the results through a graphical interface integrated into the system's web application. The system also includes a method for presenting its outputs effectively.

The system will utilize convolutional neural networks to analyze plant images, identifying diseases with high accuracy. It must be scalable to handle large datasets and adaptable to various crop types, ensuring reliable disease detection under diverse environmental conditions. The system will continuously monitor newly labeled data, triggering retraining when a predefined threshold is met. This self-retraining approach ensures the model remains up-to-date with evolving data, improving detection accuracy over time without manual intervention. Unknown or unpredicted images will be flagged for manual review by domain experts. These experts will label the images, and the newly labeled data will be incorporated into the model for future predictions RBAC will ensure secure access to system functions, with regular users (farmers) limited to viewing predictions, while admin users (domain experts) can review, label unknown images, and manage system settings.

The interface will cater to both farmers and experts, with a simple, intuitive layout for farmers to easily upload images and receive disease predictions.

©IJRASET: All Rights are Reserved

III. LITERATRE REVIEW

In their study, Sethy, P.K., and colleagues investigated the identification of rice leaf diseases using a Support Vector Machine (SVM) approach combined with deep feature extraction. They analyzed 5,932 field images categorized into four disease types: microbiological rot, blast, brown spot, and tango. The study evaluated convolutional neural networks with Transfer Active Learning across various network depths, utilizing a supervised learning classifier. The findings demonstrated that combining deep feature extraction with a classifier outperformed using transfer learning alone. The researchers also compared smaller convolutional networks, including pre-trained models and ShuffleNet, using evaluation metrics such as reliability, sensitivity, precision, false positive rate (FPR), F1 score, and training time. ResNet50 paired with SVM yielded the highest performance, achieving an evaluation score of 0.9838. Additionally, incorporating personalized attention layers in models like VGG16, VGG19, and AlexNet improved classification accuracy compared to the fc7 and fc8 layers. Furthermore, convolutional models were benchmarked against other feature extraction methods, including bag-of-features, Local Binary Patterns (LBP) with SVM, Histogram of Oriented Gradients (HOG) with SVM, and Gray Level Co-occurrence Matrix (GLCM) with SVM [1].

In their study, Systematic Review of Deep Learning Techniques in Plant Disease Detection, M. Nagaraju and colleagues discussed the limitations of existing technologies, which heavily rely on deep learning approaches. Convolutional neural networks (CNNs) have become a prominent method for detecting and predicting crop infections from images. The review explored various deep learning applications in image processing, particularly focusing on plant disease identification. It began by analyzing data sources, types of machine learning models, and image processing techniques employed in the studies. Subsequently, the review evaluated the effectiveness of several well-known deep learning techniques and examined future possibilities for incorporating hyperspectral data analysis. The main goal of the review was to promote further research aimed at enhancing the capabilities of deep learning for plant disease detection, with an emphasis on boosting accuracy and system performance [2]. In the paper titled Current and Future Applications of Statistical Machine Learning Algorithms for Agricultural Machine Vision Systems, T.U. Rehman and co-authors conducted a comprehensive review of various classifiers utilized in computer vision, analyzing their potential applications in different agricultural contexts. The paper provided valuable insights through examples and suggested appropriate statistical machine learning techniques for specific tasks, while also addressing the limitations of each method. Additionally, the study explored the future prospects of applying statistical and machine learning technologies to agriculture [3].

In the study 'Intelligent Pattern Recognition System with Application to Cotton Leaf Disease Identification,' P. R. Roth and co-authors discuss the significant impact of cotton farming on farmers' livelihoods in many regions of India, where leaf diseases pose a major challenge to crop production. The research demonstrates an effective approach for detecting and categorizing diseased leaves based on specific disease types. Advanced image processing techniques were used to capture leaf images, followed by Otsu's segmentation to isolate the background. Features such as color, contour, and structure were extracted and analyzed using a convolutional neural network (CNN). The study targeted three specific diseases: microorganisms blight, Excluding the standard, and Alteraria. Data collected from CICR Jodhpur and fields in the Wardha and Akola areas helped determine the sample size, and the classification accuracy reached 95.48%. This high accuracy indicates that the model is robust and dependable, enabling timely interventions to enhance crop yield and sustainability. The integration of CNN-based machine learning with traditional agricultural practices underscores the increasing role of technology in precision farming. Additionally, the research highlights the system's scalability for use with other crops and regions.

In the study 'A Novel and Proposed Comprehensive Methodology Using Deep Convolutional Neural Networks for Flue-Cured Tobacco Leaves Classification,' D. Siva Krishna and colleagues introduced a CNN-based method for grading flue-cured tobacco leaves. The research utilized 120 measurement techniques to analyze drying processes, reducing the dataset size from 1,450 RGB images to 256. The neural network architecture included four hidden layers, featuring 16, 32, and 64 feature cores. The first three layers applied convolution and pooling operations, while the fourth layer was fully connected. Dimensionality reduction was achieved using max-pooling. Classification was carried out across three main categories: Class-1, Class-2, and Class-3, resulting in a performance score of 85.10 on the test set, which comprised 15 images per class. The study also compared the proposed model's results with other existing methods for tobacco leaf classification.

In the study *Detection of Plant Leaf Diseases Using Deep Learning Techniques*, A. Kamilaris and F. Prenafeta-Boldú presented a survey on the use of deep learning models, particularly convolutional neural networks (CNNs), for plant disease classification based on leaf imagery. The paper emphasized the rapid advancement of image-based plant disease diagnosis due to the proliferation of accessible datasets and the increasing efficiency of deep learning architectures. By reviewing a range of datasets including PlantVillage and real-world field data, the authors highlighted how preprocessing techniques such as data augmentation, background removal, and image enhancement significantly influenced model performance.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue V May 2025- Available at www.ijraset.com

IV. PROPOSED METHOD

The proposed approach in this research provides a detailed explanation of the system's construction, including a process flow diagram and descriptions of the individual modules. The steps implemented help demonstrate the system's results. This method employs computational intelligence techniques to detect and classify plant diseases. It uses a neural network model, incorporating multiple layers, which is trained to recognize disease-affected regions on plant leaves. The procedure involves several stages: data collection, preprocessing, training the model using a convolutional neural network (CNN), and testing. Once the training is complete, users can utilize the system to determine if a given image represents a healthy or diseased leaf. The outcomes are then presented through a web-based interface.

A. Block Diagram

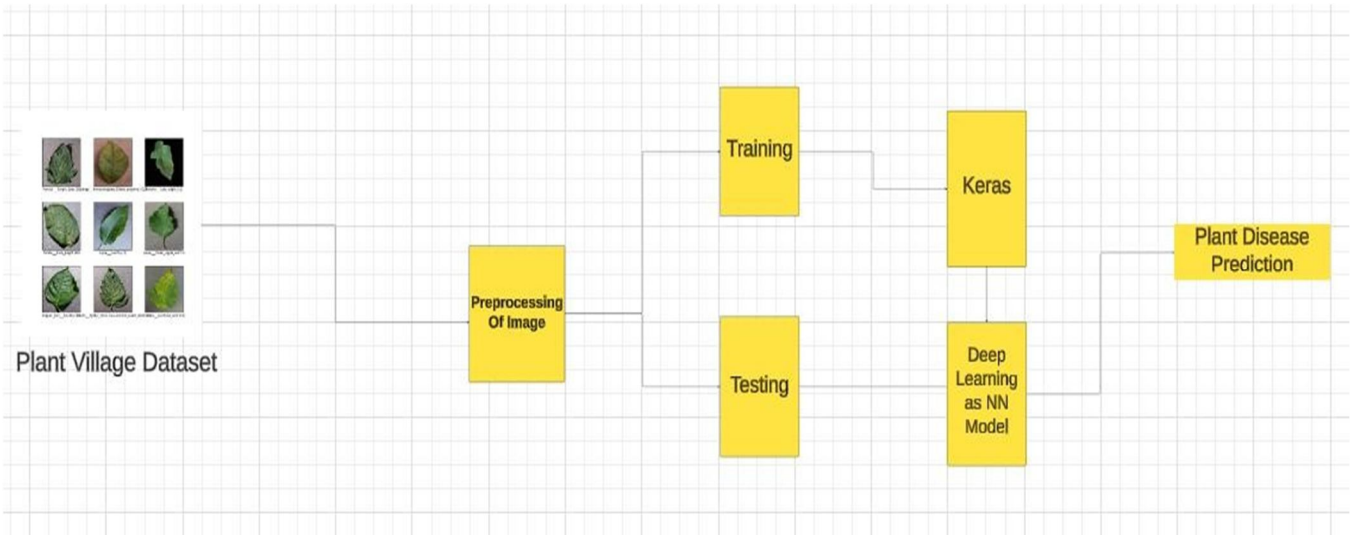


Fig 1 Block diagram of our proposed method

B. System Architecture

The architecture of the plant disease detection system consists of three core components:

1) Backend (Flask Framework): The Flask framework powers the backend, managing APIs for image upload, prediction, user authentication, and database interactions. It is also responsible for managing the retraining process when the threshold of labeled unknown images is met.

2) Frontend (Next.js 14 with Tailwind CSS): The frontend is built using Next.js for server-side rendering and optimized performance, with Tailwind CSS ensuring a responsive and accessible design. It provides interfaces for both regular users (farmers) and admin users (experts) to interact with the system.

3) CNN Model for Disease Classification: The core of the system is a CNN model that has been trained to classify plant diseases from images. When retraining is triggered, this model is updated with the new labeled data.

4) Datasets: We utilized the Plant-Village dataset, which contains over 50,000 labeled images of healthy and diseased plants across 38 classes. Each image is labeled with the plant species and the disease affecting it. This dataset was chosen due to its diversity and extensive labeling, making it suitable for training a deep learning model to detect plant diseases.

C. Preprocessing

1) Resizing: All images were resized to 256x256 pixels to ensure uniformity and to reduce the computational complexity during model training.

2) Normalization: The pixel values of the images were normalized to fall within the range [0, 1]. This ensures faster convergence during training by bringing all pixel values to a standard scale.

3) Data Augmentation: To mitigate class imbalance, data augmentation techniques were applied, including rotation, flipping, zooming, and adjusting brightness. These methods increased the dataset size artificially, improving the model's ability to handle various image conditions.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |

D. Model Design

The plant disease detection model is based on the ResNet50 architecture, a deep CNN with 50 layers. ResNet50 is well- suited for complex image classification tasks due to its ability to handle large amounts of data while avoiding the vanishing gradient problem, thanks to its residual connections.

E. Model Layers

1) Convolutional Layers: These layers extract local features from the input images, such as disease spots, color changes, and other visual cues associated with plant diseases.
2) Residual Blocks: The residual connections help maintain the flow of gradients during backpropagation, allowing the network to train effectively, even when the architecture is deep.
3) Global Average Pooling: This layer reduces the dimensionality of the data by averaging each feature map, which helps reduce the risk of overfitting.
4) Fully Connected Layer: After feature extraction, the model maps the learned features to a probability distribution over the disease classes using fully connected layers.
5) Softmax Output: The final layer uses the softmax function to assign probabilities to each disease class, enabling multi- class classification.

F. Training Parameters

1) Optimizer: The Adam optimizer was used for its adaptive learning rate capabilities, ensuring efficient convergence during training.
2) Loss Function: Categorical cross-entropy was chosen as the loss function, given that the task involves multi-class classification.
3) Learning Rate: An initial learning rate of 0.0001 was set, with an exponential decay to adjust as training progressed.
4) Batch Size: The model was trained with a batch size of 32 images per iteration, striking a balance between memory usage and model convergence.

G. Retraining Process

A major innovation of this system is the implementation of an automated retraining mechanism that triggers whenever a set threshold of new, labeled images is reached. The retraining workflow is outlined below:

1) Image Collection: As users submit plant images for disease prediction, the model attempts to classify them. If the model's confidence is below a certain threshold (e.g., 50%), the image is classified as "unknown."
2) Expert Labeling: These unknown images are stored in a separate directory and can be accessed only by authenticated admin users. The admin dashboard provides a list of all unknown images awaiting expert labeling.
3) Threshold Trigger: Once a threshold of 100-150 labeled images is reached, the retraining process is automatically initiated.
4) Retraining Pipeline: The retraining pipeline adds the newly labeled images to the existing dataset and retrains the model on the combined data. This ensures that the model can learn from new data without manual intervention.
5) Model Versioning: Each retraining cycle results in a new version of the model. This allows for tracking improvements in model performance over time, as well as the option to roll back to previous versions if needed.

V. IMPLEMENTATION

A. Frontend Development

The frontend of the application is developed using **Streamlit**, a lightweight and efficient Python-based framework ideal for building interactive web interfaces for machine learning models. Streamlit allows rapid deployment of data science applications with minimal frontend code, making it highly suitable for researchers and developers aiming to present model outputs in an intuitive and accessible manner. It offers seamless integration with machine learning libraries such as TensorFlow, ensuring smooth communication between the user interface and the prediction engine.

The choice of Streamlit ensures a **responsive, user-friendly experience** without the complexity of traditional web development stacks. Users can upload plant leaf images and receive disease predictions in real time, displayed with clear labels and confidence scores.

The simple and clean layout makes it easy to navigate, even for users with minimal technical expertise. This design approach aligns with the goal of making AI-powered plant disease diagnosis accessible and usable in agricultural contexts, especially in regions where technical infrastructure or expertise may be limited.

1) User Roles and Functionalities

The system is designed with two primary user roles, each with distinct capabilities and interfaces tailored to their needs:

- Regular Users: Regular users interact with a simple, intuitive interface allowing them to upload images easily. Once uploaded, users receive real-time feedback on the disease classification associated with the image. In addition, relevant information about the predicted disease is displayed, providing users with valuable insights in a user-friendly format.
- Admin Users: Admin users are provided with a more complex interface via a dedicated dashboard that allows for comprehensive management and oversight of the image labelling and retraining processes. Key functionalities for admin users include:
- Viewing and Labelling Unpredicted Images: Admins can review and label images that the system has yet to classify, ensuring that the dataset remains up-to-date and accurate.
- Monitoring Labelling and Retraining: The dashboard provides a clear view of how many labelled images are queued for retraining, offering transparency into the system's current state.
- Manual Model Retraining: Admins have the ability to manually trigger retraining cycles. This feature is crucial when the system accumulates a significant amount of new data or when performance improvements are required.
- Tracking Retraining Cycles and Performance: Admins can access historical data on previous retraining sessions, along with key performance metrics such as model accuracy and loss, enabling informed decisions about the system's health and effectiveness.

2) Frontend Features

The frontend is designed to be responsive and accessible across different devices. Key features include:

- Image Upload: Regular users can upload plant images for disease prediction. The system provides immediate feedback by displaying the predicted disease class and a confidence score.
- Prediction History: Users can view their past predictions, allowing them to track diseases across multiple submissions.
- Admin Dashboard: Admin users can access a detailed dashboard showing a list of unknown images awaiting labelling, the status of the current model, and retraining history.
- Retraining Visualization: After each retraining cycle, the system logs performance metrics (e.g., accuracy, precision) for the updated model. Admins can view these logs via the dashboard, allowing for transparent model tracking.

B. Backend Design

The backend system of the **Plant Disease Classifier** is designed using **Streamlit**, a powerful framework for building interactive web applications in Python. Streamlit was chosen for its ease of use, rapid prototyping capabilities, and seamless integration with machine learning models, making it an ideal choice for creating a user-friendly interface for classification.

1) Key Components

The backend comprises several core components, each designed to handle a distinct set of responsibilities. These components are interconnected through **Restful APIs**, providing a clear separation of concerns and allowing seamless integration with the frontend. This design promotes scalability and ensures that each module can be independently improved or replaced without affecting the entire system.

- Prediction Module: This module handles the core functionality of the backend—disease prediction. Upon receiving an image upload from the frontend, the module processes the image and runs it through a pre-trained machine learning model. The results, including the predicted disease class and any relevant probabilities or metrics, are returned to the frontend for display to the user.
- Image Preprocessing: The uploaded images are cleaned, resized, and normalized before being fed into the model to ensure consistency and accuracy in the predictions.
- Model Inference: A machine learning or deep learning model, likely built using a framework like TensorFlow or Py-Torch, is used for disease classification.

2) Authentication and Authorization Module:

This component manages user access to the system.

Using token-based authentication (e.g., JWT), it ensures that both regular users and admin users have secure access to the platform.

- Regular Users: Can upload images and view results, but cannot access admin functionalities.
- Admin Users: Require higher-level privileges to access the dashboard, label unclassified images, and retrain the model. Role-based access control (RBAC) ensures that only authenticated admins can perform these actions.
- Retraining Module: As the system gathers new data from labelled images, this module manages the retraining process. Admins can trigger retraining manually through the dashboard, or the system can schedule periodic retraining based on a predefined number of newly labelled images.
- Data Collection: Stores new labelled images in a training database, segregated from the main production model's dataset.
- Model Retraining: When retraining is triggered, the system combines the newly labelled images with the existing dataset to refine the model. Flask's asynchronous capabilities or a task queue system like Celery can be used to handle this process without impacting the system's performance.
- Versioning: The retraining module keeps track of different model versions, allowing the system to revert to an earlier version if a newly trained model underperforms.

3) Monitoring and Logging Module

To ensure transparency and maintain the health of the backend, this module handles logging and monitoring. All significant events, such as image uploads, prediction requests, retraining events, and authentication attempts, are logged for future reference.

- Logging: Detailed logs help in tracking the system's usage patterns and performance metrics. Error handling is also logged to detect and address issues promptly.
- Performance Monitoring: This component tracks the system's performance, particularly during high-traffic periods, and provides insights on the efficiency of prediction queries, retraining cycles, and model performance.
- RESTful API Design
- The communication between the frontend and backend is facilitated through well-defined RESTful APIs. Each API endpoint is designed to handle specific tasks, ensuring clear and secure data exchange:
- /upload: Accepts image uploads from the frontend and returns the predicted disease class and relevant information.
- /authenticate: Handles user authentication, issuing tokens for authorized access.
- /retrain: Allows admin users to trigger the retraining process, with options to monitor progress.
- /logs: Provides access to logs for system monitoring, particularly useful for admin user.

4) Database Management

The system initially utilizes **SQLite** for data storage, chosen for its simplicity, low overhead, and suitability for small-scale applications.

SQLite is lightweight and requires minimal configuration, making it an ideal choice during the early stages of development. The database is designed to handle the following types of data:

- User Credentials and Roles: The database stores user information, including usernames, hashed passwords, and roles (e.g., regular user or admin). Role-based access control (RBAC) ensures that only authorized users can access certain functionalities, with admin users granted elevated privileges for managing the system.
- Uploaded Images and Metadata: Each uploaded image is stored in the system with associated metadata, such as the predicted class, confidence score, and timestamp of the upload. This information is used to track user interactions and provide insights into the system's prediction accuracy and user trends.
- Labeled "Unknown" Images: When the system cannot confidently predict a disease class, the image is classified as "unknown" and stored in the database. These images await expert labeling, and once labeled, they are added to the training dataset for future model retraining. As the system scales, SQLite may no longer be sufficient to handle the increasing volume of data. To address this, the system is designed to allow easy migration to more robust databases such as **PostgreSQL** or **MongoDB**. PostgreSQL would offer advanced features like ACID compliance, scalability, and complex query handling, while MongoDB could be useful for storing unstructured or semi-structured data, such as images and logs.

5) Image Processing and Prediction

When an image is submitted by the user, it undergoes several preprocessing steps before being passed to the Convolutional Neural Network (CNN) model for prediction. The preprocessing ensures that the image is in the proper format for the model to handle effectively:

- Resizing: Images are resized to a uniform dimension that matches the input size expected by the model. This ensures consistency and reduces computational complexity.
- Normalization: The pixel values of the image are normalized to fall within a specific range (e.g., [0, 1] or [-1, 1]), which helps improve the performance of the CNN by reducing input variability.
- Data Augmentation (Optional): For more robust training, data augmentation techniques like rotation, flipping, or zooming could be applied to increase the variability in the dataset.
- Once preprocessed, the image is passed through the trained CNN model. The system calculates the confidence score for the prediction:
- High Confidence: If the model's confidence score exceeds a predefined threshold (e.g., 80%), the predicted disease class and relevant information are returned to the user.
- Low Confidence: If the confidence score falls below the threshold, the image is classified as "unknown." These images are saved in the database for later review and labeling by experts. This approach ensures that the system provides reliable predictions to users while allowing for continuous improvement of the model through labeled "unknown" images.

6) Retraining Task Scheduler

To ensure that the system remains up-to-date with new data, a retraining task scheduler is integrated into the backend. The scheduler automates the retraining process by monitoring the number of labeled "unknown" images in the database. Once a certain threshold of labeled images is reached, the system triggers retraining automatically.

Key functionalities of the retraining task scheduler include:

- Threshold Monitoring: The system continuously monitors the number of labeled images that are classified as "unknown" and stored in the database. Once this number reaches a predefined threshold (e.g., 100-150 images), the scheduler initiates the retraining process.
- Automated Retraining: When the threshold is met, the scheduler automatically triggers the retraining module. This process involves combining the newly labeled images with the existing dataset, updating the CNN model with fresh data. The retraining process can be managed asynchronously to avoid disrupting the system's normal operations, potentially using a task queue system like Celery.
- Notification System (Optional): Admin users can receive notifications about the retraining process, such as when retraining is initiated, completed, or if there are errors. This feature allows for better oversight of model performance and system health.

VI. RESULTS AND DISCUSSION

A. Model Performance

The system's predictive model is based on ResNet50, a deep convolutional neural network well-suited for image classification tasks. Initial training on the Plant-Village dataset yielded a classification accuracy of 92%, a strong starting point given the complexity of plant disease classification. The model's performance was rigorously evaluated using a separate test set, ensuring that the reported accuracy reflects real-world effectiveness and was not inflated by overfitting to the training data.

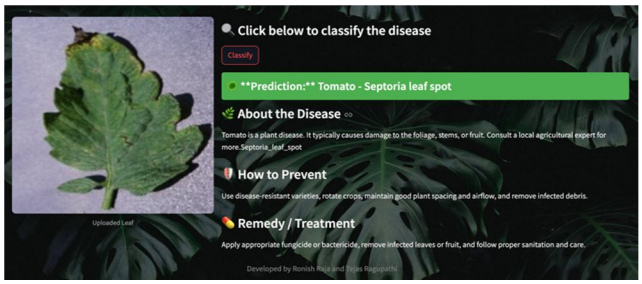


Fig 2 Prediction Page

With the system's retraining mechanism in place, the model's accuracy improved over time. After incorporating newly labeled images from the real-world deployment, the accuracy increased to 95% following three retraining cycles. This improvement underscores the model's ability to adapt to new, diverse data while maintaining strong performance in classifying plant diseases.

B. Retraining Impact

The retraining process played a crucial role in enhancing the model's ability to identify previously unseen or "unknown" plant diseases. After each retraining cycle, the number of unclassified images decreased, demonstrating that the system successfully learned from newly labeled data provided by experts.

The system's automated retraining feature allowed for seamless integration of new data without requiring manual intervention from the admins, making the system scalable for continuous improvement. As more data is collected, particularly for rare or difficult-to-classify diseases, the model is expected to continue improving in accuracy and versatility.

C. System Usability

A usability study was conducted with a diverse group of participants, including 10 farmers and 5 plant pathologists, to evaluate the system's interface and overall experience:

- Farmers: The farmers reported that the system was intuitive and easy to use. They particularly appreciated the simplicity of the image upload interface, which required minimal technical knowledge. The system's fast response times for disease predictions were also highlighted as a positive aspect, contributing to a smoother user experience, especially for users with limited internet connectivity.
- Plant Pathologists (Admin Users): The pathologists, who acted as admin users, found the system's labeling process straightforward. The dashboard provided clear visibility into the labeling workflow, and the ability to trigger retraining manually, if necessary, was appreciated for its transparency and control. They also found value in the system's performance tracking, which allowed them to monitor the model's improvement across retraining cycles.

VII. CONCLUSION

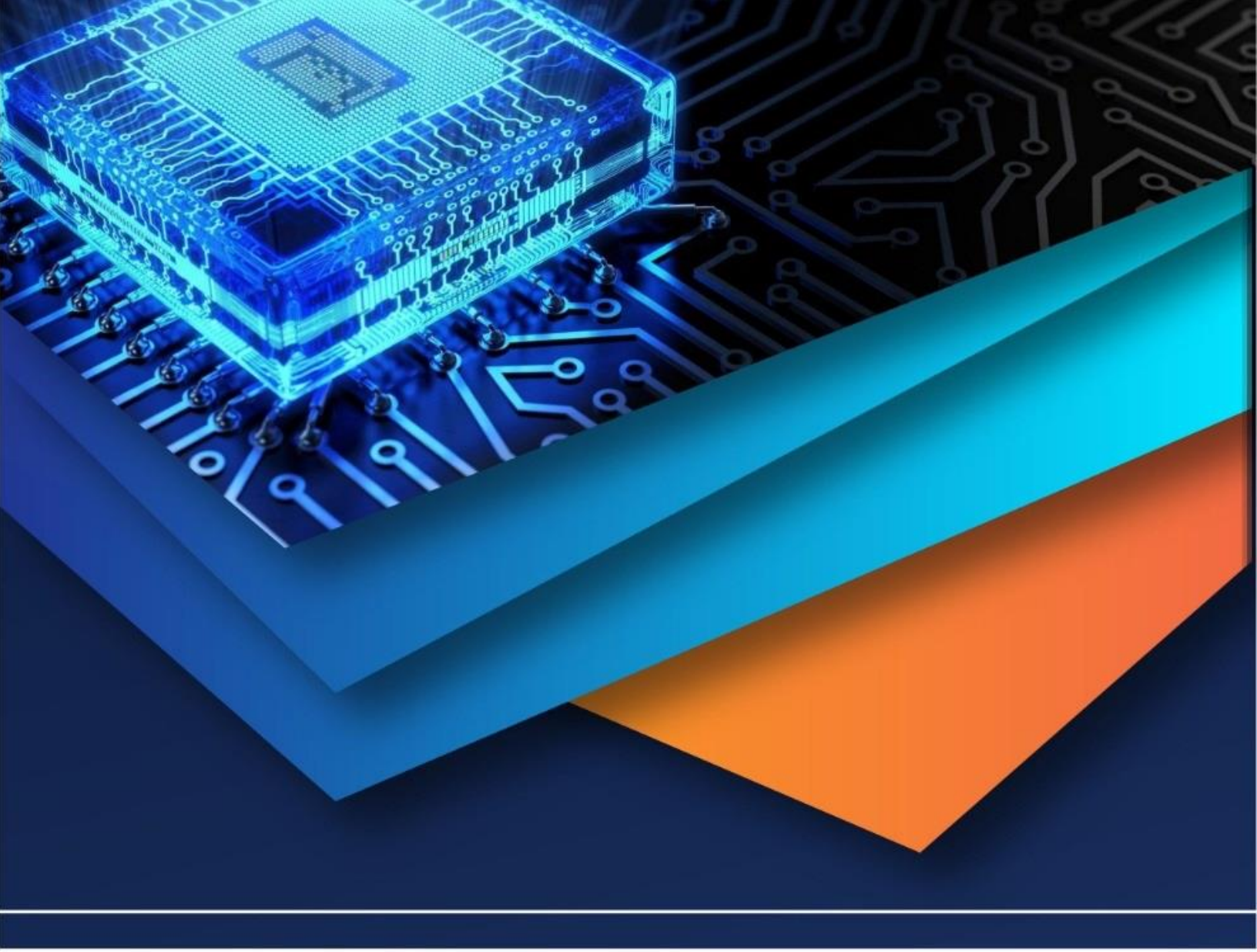
This research introduces a cutting-edge system for automated plant disease detection that leverages dynamic retraining, allowing the model to remain effective and up-to- date in real-world agricultural environments. The system continuously incorporates newly labelled images from expert users, enabling it to adapt to emerging diseases or variations in disease patterns, which are common in evolving ecosystems. By integrating secure APIs and implementing role-based access control, the system ensures robust management and scalability, offering controlled access to both the model retraining process and the expert labelling of unknown images. This design allows for seamless updates and efficient collaboration between regular users and admin users, such as plant pathologists, who can easily monitor model performance and trigger retraining when necessary. This could lead to more accurate predictions and insights into disease spread based on real- time conditions like humidity, temperature, and soil quality. Furthermore, integrating more advanced machine learning architectures, such as **transformer-based models** or hybrid deep learning approaches, could significantly enhance the system's classification accuracy, particularly for complex or previously unseen disease cases. Such advancements would make the system more adaptable and resilient, ultimately providing a powerful, scalable tool for farmers and agricultural experts to manage plant health more effectively in diverse and dynamic environments.

REFERENCES

[1] M.Nagaraju and C. Priyanka. "Systematic Review of Deep Learning Techniques in Plant Disease Detection", International Journal of Systems Assurance Engineering and Management. Vol. 11, no.3, pp. 547–560, 2020.
[2] S. Sladojevic et al., *Comput. Intell. Neurosci.*, 2016.
[3] S. Mohanty et al., *Front. Plant Sci.*, 2016.
[4] K. P. Ferentinos, *Comput. Electron. Agric.*, 2018.
[5] A. Too et al., *Comput. Electron. Agric.*, 2019.
[6] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," ICLR, 2015.
[7] M. J. Islam et al., "A Review on Deep Learning Techniques for the Diagnosis of Novel Coronavirus (COVID-19)," *IEEE Access*, vol. 8, pp. 130906–130924, 2020. (For inspiration on dynamic retraining approaches)
[8] S. Kaur, P. Shreelekha and G. Shivani. "Plants Disease Identification and Classification Through Leaf Images: A Survey", Archives of Computational Methods in Engineering. Vol. 26, No.2, pp. 507–530, 2019.
[9] TU. Rehman, M. Md Sultan, K. Young Chang, J. Jian and J. Shin. "Current and Future Applications of Statistical Machine Learning Algorithms for Agricultural Machine Vision Systems", Computers and Electronics in Agriculture. Vol.156, pp.585–605, 2019.

[10] HS. Muhammad, P. Johan and MA. Khalid. “Plant Disease Detection and Classification by Deep Learning”, Plants. Vol. 8,No.468, pp.1-22, 2019.
[11] A. Kamilaris, X. Francesc, and Prenafeta-Boldú. “Deep Learning in Agriculture: A Survey”, Computers and Electronics in Agriculture. Vol. 147, pp. 70–90, 2018.
[12] G. Dhingra, V. Kumar, and HD. Joshi. “Study of Digital Image Processing Techniques for Leaf Disease Detection and Classification”, Multimedia Tools and Applications. Vol. 77, No.15, 2018.
[13] K. Singh, S. Kumar and P. Kaur. “Support vector machine classifier based detection of fungal rust disease in Pea Plant (Pisamsativam)”, International Journal of Information Technology. 2018.
[14] V. Singh and AK. Misra. “Detection of plant leaf diseases using image segmentation and soft computing techniques”, Information Processing Agriculture. Vol. 4, No.1,pp. 41–49, 2017.
[15] S. Kaur, S. Pandey and S. Goel. “Semi-automatic leaf disease detection and classification system for soybean culture”, IET Image Processing. Vol. 12, No. 6, pp.1038–1048, 2018.
[16] P. Kaur, HS. Pannu and AK. Malhi. “Plant disease recognition using fractional-order Zernike moments and SVM classifier”, Neural Computing and Application. 2019.
[17] Fuentes A, Yoon S, Park DS. Deep learning based techniques for plant diseases recognition in real-field scenarios. In: Advanced concepts for intelligent vision systems. Cham: Springer; 2020
[18] Boulent J, Foucher S, Théau J, St-Charles PL. Convolutional neural networks for the automatic identification of plant diseases. Front Plant Sci. 2019;10:941.
[19] Shekhawat RS, Sinha A. Review of image processing approaches for detecting plant diseases. IET Image Process. 2020;14(8):1427–39.
[20] Thangaraj R, Anandamurugan S, Kaliappan VK. Automated tomato leaf disease classification using transfer learning-based deep convolution neural network. J Plant Dis Prot. 2020.
[21] Atila M, Uar M, Akyol K, Uar E. Plant leaf disease classification using efficientnet deep learning model. Ecol Inform. 2021;61:101182.

©IJRASET: All Rights are Reserved | SJ Impact Factor 7.538 | ISRA Journal Impact Factor 7.894 |



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Call : 08813907089 (24*7 Support on Whatsapp)