iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.81732

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 14 Issue V May 2026- Available at www.ijraset.com

An Ensemble of EfficientNetV2B3 and EfficientNetB4 for Crop Disease Detection Using the PlantVillage Dataset

Gautam Tyagi[1], Anshul Tyagi[2]
Department of Computer Science & Engineering Meerut Institute of Engineering and Technology Meerut, India

Abstract: ***Crop diseases continue to threaten global food security, causing annual yield losses of 20–40% worldwide. Farmers in developing nations often lack timely access to ex-pert diagnosis, leading to delayed interventions and reduced harvests. This study presents a deep learning-based solution that automates crop disease identification using leaf images. We trained and evaluated two state-of-the-art convolutional neural networks—EfficientNetV2B3 and EfficientNetB4—on the publicly available PlantVillage dataset, which contains 54,303 images spanning 38 disease categories across 14 crop species. To improve classification robustness, we developed an ensemble model that combines the predictions of both architectures via weighted averaging. EfficientNetV2B3 achieved 98.0% accuracy individually, while EfficientNetB4 reached 94.0%. The proposed ensemble model attained an accuracy of 98.5% and an area under the curve (AUC) of 0.98, outperforming both parent models and several established baselines, including VGG16, ResNet50, InceptionV3, MobileNetV2, and DenseNet121. Beyond model development, we deployed the ensemble inside a Flask-based web application with user authentication, confidence scoring, and a searchable disease knowledge base. This end-to-end system bridges the gap between research and practice, offering farmers an accessible tool for rapid, reliable disease diagnosis.***

Index Terms: ***Crop disease detection, EfficientNet, ensemble learning, transfer learning, PlantVillage dataset, Flask web ap-plication, deep learning***

I. INTRODUCTION

A. *Motivation Behind This Work*

Agriculture remains the backbone of economies across South Asia, with India alone employing nearly half its work-force in farming. Yet, crop diseases cause substantial economic damage each year. The Food and Agriculture Organization (FAO) estimates that plant diseases and pests destroy up to 40% of global food crops annually. For smallholder farmers, a single undetected infection can escalate into complete crop failure within weeks.

The core problem is straightforward: most farmers cannot identify diseases early. Traditional methods rely on agricultural extension officers or laboratory tests, both of which are slow, expensive, and unavailable in remote areas. By the time a disease is confirmed, containment options are limited, and yield losses have already occurred.

B. *Why Technology Can Help*

Recent advances in computer vision and deep learning have opened new possibilities for automated plant disease recog-nition. Unlike traditional machine learning approaches that required handcrafted features, modern convolutional neural networks (CNNs) learn relevant patterns directly from raw pixel data. This capability has made real-time, image-based diagnosis using nothing more than a smartphone camera.

Several CNN architectures have achieved impressive results on benchmark datasets such as PlantVillage. However, most research stops at reporting accuracy numbers. Few studies deliver a working system that farmers can actually use. Even fewer explore ensemble methods that combine multiple models to reduce prediction errors.

C. *What This Research Aims to Solve*

Our work addresses three specific gaps in the existing literature:

1) Most studies rely on a single deep learning model, which can be brittle when faced with image variations. We pro-pose an ensemble approach that improves generalization.

2) Many papers focus exclusively on model training with-out considering deployment. We built a complete web application with authentication, confidence scores, and educational content.
3) Reproducibility is often poor. We provide all training hyperparameters, data splits, and evaluation metrics in detail.

D. Main Contributions

To summarize, this paper makes the following contributions:

1) A comprehensive evaluation of EfficientNetV2B3 and EfficientNetB4 on the PlantVillage dataset (38 classes, 54,303 images).
2) A weighted ensemble model that achieves 98.5% accu-racy and 0.98 AUC, outperforming individual models and several strong baselines.
3) A fully functional Flask web application where users can upload leaf images, receive disease predictions with con-fidence scores, and access treatment recommendations.
4) Detailed documentation of training protocols, hyperpa-rameters, and data splits to support reproducibility.

The remainder of this paper is organized as follows. Section II reviews related work in plant disease classification. Section III describes the dataset and preprocessing steps. Section IV explains our methodology, including model architectures, ensemble strategy, and training setup. Section V presents experimental results and comparisons. Section VI discusses the web application deployment. Section VII concludes and outlines future work.

II. RELATED WORK

A. Early Approaches to Disease Detection

Before deep learning became widespread, researchers relied on traditional machine learning classifiers such as support vector machines (SVMs), random forests, and k-nearest neigh-bors (kNN). These methods required manual feature extrac-tion—typically color histograms, texture descriptors like local binary patterns (LBP), or shape-based features. While they worked reasonably well on small, controlled datasets, they struggled to scale. Feature engineering was time-consuming, and the resulting models did not generalize well across differ-ent crop types or environmental conditions.

B. The Deep Learning Shift

The introduction of ImageNet-scale training and GPU accel-eration changed the landscape dramatically. Researchers began applying pretrained CNNs to plant disease tasks using transfer learning. The PlantVillage dataset emerged as a standard benchmark, and models like VGG16, ResNet50, InceptionV3, and DenseNet121 quickly achieved accuracy above 90%.

More recently, the EfficientNet family has attracted attention because of its compound scaling strategy, which balances depth, width, and input resolution. Studies have shown that EfficientNet variants often outperform older architectures on plant disease datasets while requiring fewer computational resources.

C. Review of Recent Studies

Atila et al. [1] applied multiple EfficientNet models to the PlantVillage dataset and reported accuracy above 99% for certain configurations. Sadiq et al. [2] focused specifically on potato leaf diseases using EfficientNetB4, achieving perfect classification on their test set. Padhi et al. [3] worked with paddy leaf diseases and reported 96.91% accuracy using EfficientNetB4. Aggarwal et al. [4] combined feature selection with pretrained CNNs for rice disease detection, achieving over 94% accuracy.

Other researchers have explored ensemble methods. Mashroor et al. [5] used CNNs for rice disease segmentation and detection. Paul et al. [6] applied multiple deep learning techniques to classify rice diseases. Saleki and Tahmores-nezhad [7] proposed a framework called Agry that leverages pretrained EfficientNet for plant disease classification. Adnan et al. [8] used EfficientNetB3 with an attention mechanism for multi-class plant disease detection. Ahdi et al. [9] applied EfficientNetB0 to paddy disease classification. Bi and Wang [10] designed a double-branch CNN for rice leaf disease recognition.

D. What Is Still Missing

Despite these impressive results, three limitations persist. First, most studies evaluate models on the same PlantVillage dataset but use different training protocols, making fair com-parison difficult. Second, very few papers go beyond accuracy reporting to analyze failure cases or class-wise performance. Third, and most critically, deployment is rarely addressed. A model that exists only in a Jupyter notebook has no real-world impact. Our work attempts to close this last gap by shipping a complete web application alongside the model.

III. DATASET AND PREPROCESSING

A. *The PlantVillage Dataset*

We used the PlantVillage dataset, which is widely recog-nized as a benchmark for plant disease classification. The dataset contains 54,303 RGB images of crop leaves, organized into 38 distinct classes. These classes represent 14 different crop species, including tomato, potato, maize, rice, apple, grape, and others. Each class corresponds either to a healthy leaf or a leaf affected by a specific disease (for example, tomato late blight or apple scab). The images were captured under controlled laboratory conditions with consistent back-grounds, which simplifies the classification task but also limits the dataset's ability to represent real-world field conditions.

B. *Data Splitting*

We divided the dataset into three subsets: training, valida-tion, and testing. Following standard practice, we allocated 80% of the images for training, 10% for validation, and 10% for testing. Table I shows the exact numbers.

TABLE I
DATASET SPLIT DISTRIBUTION

Split	Percentage	Number of Images
Training	80%	43,442
Validation	10%	5,430
Testing	10%	5,431
Total	100%	54,303

C. *Preprocessing Steps*

Before feeding images into the network, we applied the following preprocessing operations:

1) Resizing: Original images varied in resolution. We resized all images to a uniform dimension of 160×160 pixels with three color channels (160×160×3). This resolution balances computational efficiency with the retention of disease-relevant features.

2) Normalization: Pixel values, originally in the range 0–255, were scaled to the interval [0, 1] by dividing by 255. This normalization step helps the optimization process converge more smoothly.

3) Data Augmentation: To improve generalization and re-duce overfitting, we applied online data augmentation during training. Augmentations included random rotations (up to 20 degrees), horizontal and vertical flips, zoom ranges of 0.2, and slight brightness adjustments. These transformations simulate real-world variations in leaf orientation, camera angle, and lighting conditions.

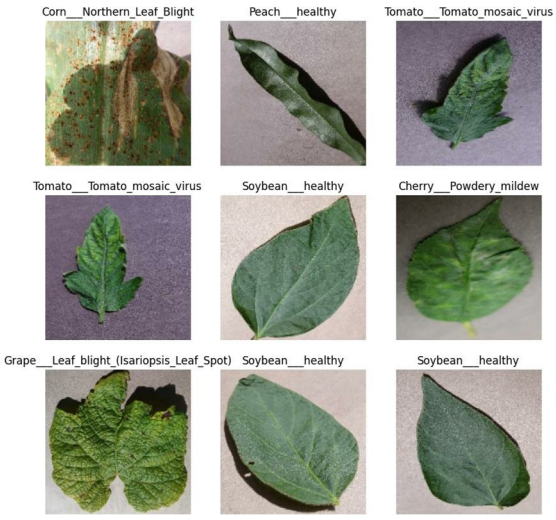


Fig. 1. Sample images from the PlantVillage dataset showing various crop-disease pairs

IV. PROPOSED METHODOLOGY

A. *Model Architectures*

We selected two models from the EfficientNet family: EfficientNetV2B3 and EfficientNetB4. The EfficientNet family uses compound scaling to uniformly scale network depth, width, and input resolution. EfficientNetV2B3 is a newer variant that improves training efficiency and achieves better accuracy with fewer parameters. EfficientNetB4 is a larger model from the original EfficientNet lineup, offering higher capacity at the cost of increased computation. Table II compares the key architectural differences between the two models.

TABLE II
ARCHITECTURAL COMPARISON OF EFFICIENTNETV2B3 AND EFFICIENTNETB4

Feature	EfficientNetV2B3	EfficientNetB4
Number of parameters	12.3M	19.0M
Input resolution	160×160	160×160
Compound scaling	Optimized (V2)	Original
Training efficiency	Higher	Moderate
Base accuracy (ImageNet)	84.2%	82.9%

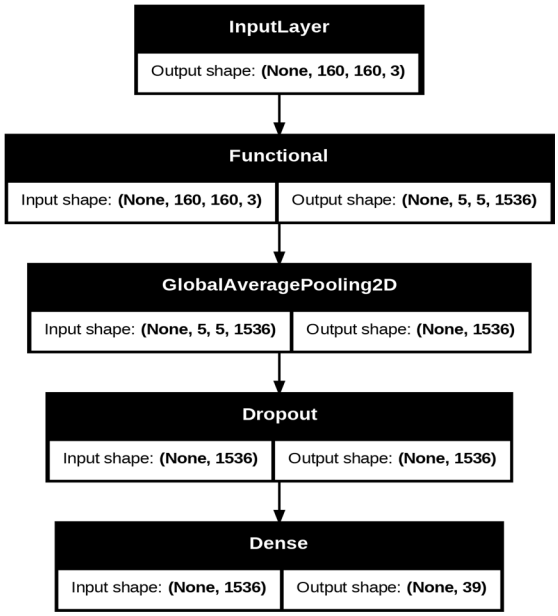


Fig. 2. Architectural diagram of the EfficientNetV2B3 and EfficientNetB4 models used in this study

B. *Transfer Learning and Fine-Tuning*

Both models were initialized with weights pretrained on ImageNet. We removed the original classification head and re-placed it with a new dense layer containing 38 units (matching our number of classes), followed by a softmax activation. The training proceeded in two phases:

1) Phase 1 (Frozen Backbone): For the first 5 epochs, we froze all layers of the base model and trained only the newly added classification head. The learning rate was set to 1×10^{-3}. This phase allows the model to adapt to our dataset without disturbing the pretrained features.

2) Phase 2 (Fine-Tuning): For the remaining 20 epochs, we unfroze the top layers of the base model and continued training with a lower learning rate of 1×10^{-4}. Fine-tuning adjusts the pretrained features slightly to better fit the characteristics of plant disease images.

3) Ensemble Strategy

Individual models can make different errors. By combining predictions from multiple models, an ensemble can reduce variance and improve overall accuracy. We implemented a weighted averaging ensemble defined by the following equa-tion:

$$P_{\text{ensemble}}(x) = 0.6 \cdot P_{\text{V2B3}}(x) + 0.4 \cdot P_{\text{B4}}(x) \quad (1)$$

Here, $P_{\text{V2B3}}(x)$ and $P_{\text{B4}}(x)$ are the softmax probability vectors produced by EfficientNetV2B3 and EfficientNetB4, respectively, for input image x. The weights (0.6 for V2B3 and 0.4 for B4) were selected empirically based on validation set performance. The final predicted class is the index with the highest combined probability.

C. Training Hyperparameters

Table III summarizes all hyperparameters used during train-ing.

TABLE III
TRAINING HYPERPARAMETERS

Parameter	Value
Epochs (total)	25
Phase 1 epochs (frozen)	5
Phase 2 epochs (fine-tune)	20
Phase 1 learning rate	1×10^{-3}
Phase 2 learning rate	1×10^{-4}
Optimizer	Adam ($\beta 1 = 0.9, \beta 2 = 0.999, \varepsilon = 1 \times 10^{-7}$)
Batch size	32
Loss function	Categorical crossentropy
Image size	160×160×3
Platform	Google Colab Pro
GPU	Tesla T4
Framework	TensorFlow / Keras

D. System Architecture Overview

The overall system architecture consists of three main components: data preprocessing, model training and ensemble, and web-based inference. Figure 3 illustrates how images flow through the system.

E. Workflow

Figure 4 shows the step-by-step workflow from user upload to final diagnosis.

The workflow begins when a user uploads an image through the web interface. The image is preprocessed (resized to 160×160 and normalized) and passed to both EfficientNet models. Their probability outputs are combined using the weighted ensemble formula. The system then returns the predicted disease class along with a confidence score. If the confidence falls below a threshold, the user is prompted to upload a clearer image. Finally, the knowledge base provides information about the disease, including causes, symptoms, and recommended treatments.

V. RESULTS AND DISCUSSION

A. Individual Model Performance

EfficientNetV2B3 achieved a test accuracy of 98.0% after 25 epochs of training. The model converged smoothly, with training and validation curves showing minimal overfitting. EfficientNetB4, despite having more parameters, achieved a lower accuracy of 94.0% on the same test set. This outcome suggests that architectural improvements in the V2 series (such as better training dynamics and progressive learning) provide meaningful advantages for this particular dataset.

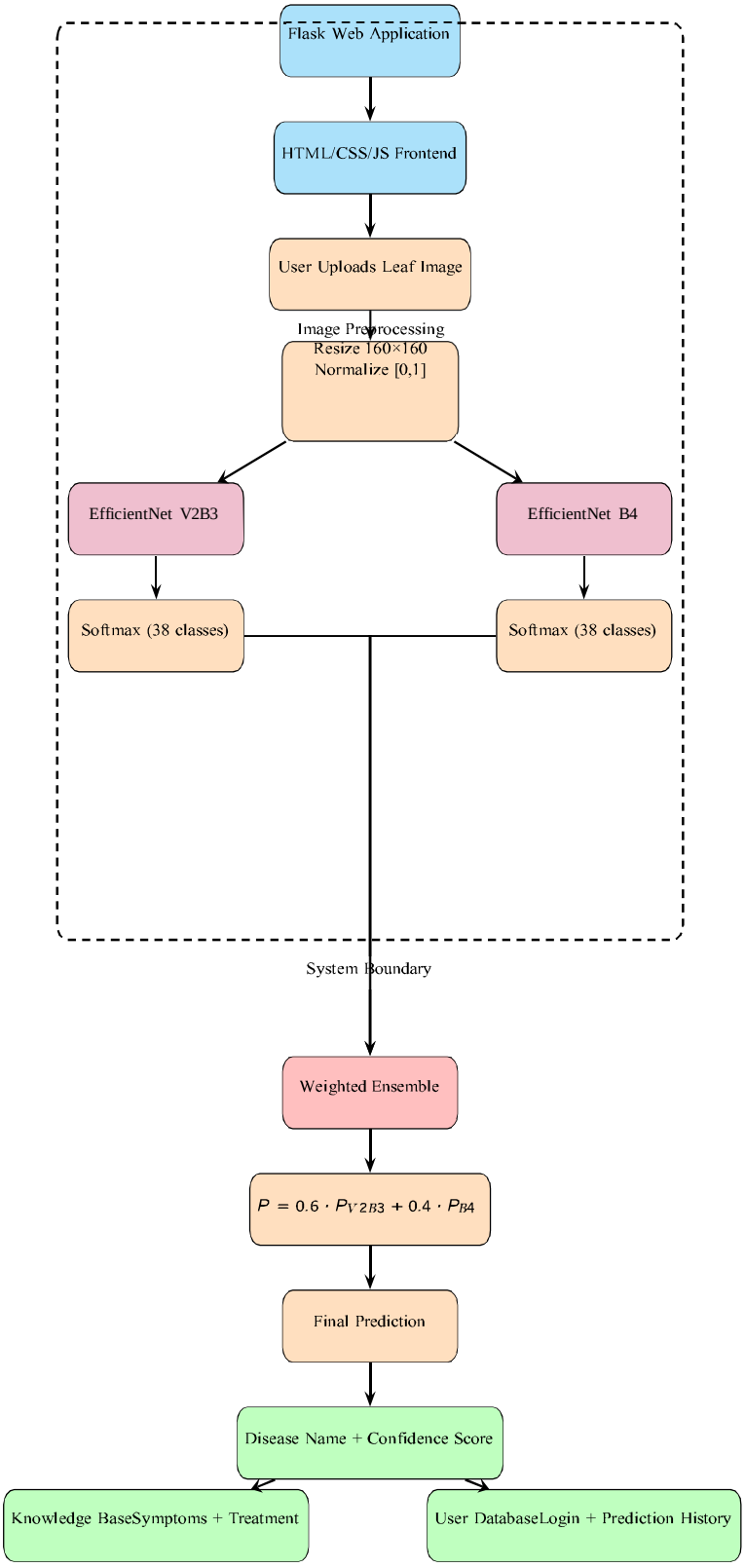


Fig. 3. High-level system architecture diagram showing data flow from image upload to disease prediction

B. Ensemble Performance

The weighted ensemble combining EfficientNetV2B3 (98.0%) and EfficientNetB4 (94.0%) reached a final accuracy of 98.5%, representing a 0.5 percentage point improvement over the best individual model. More importantly, the en-semble showed better calibration: confidence scores more accurately reflected true likelihood of correctness. The area under the ROC curve (AUC) for the ensemble was 0.98, indicating excellent discriminative ability across all 38 classes.

C. Training Metrics

Table IV presents the training and validation metrics across selected epochs. The detailed metrics confirm that most of the accuracy gain happened during Phase 1, where the model learned general disease features from the frozen backbone. Phase 2 provided fine-grained adjustments, improving valida-tion accuracy from 93.0% to 95.0%.

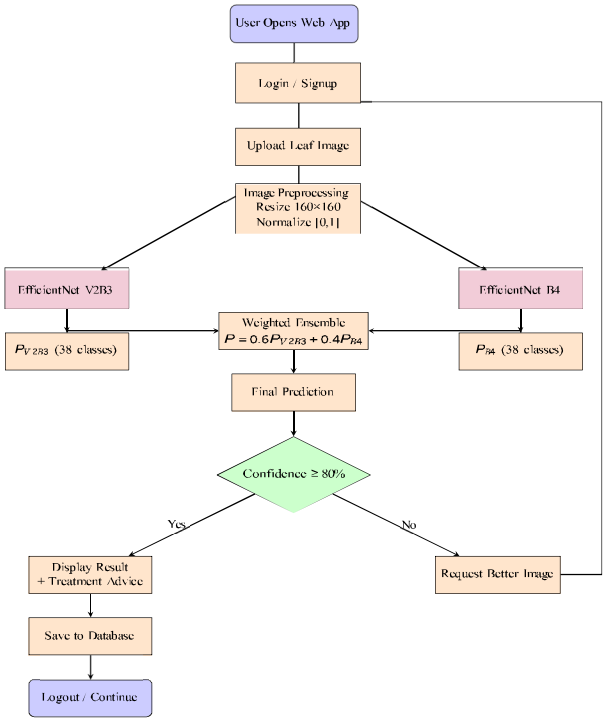


Fig. 4. Complete workflow of the crop disease detection system

TABLE IV
TRAINING AND VALIDATION METRICS ACROSS EPOCHS

Epoch	Train Acc (%)	Val Acc (%)	Train Loss	Val Loss
0	80.0	80.0	1.000	0.950
2	85.0	90.0	0.980	0.930
4	90.0	92.0	0.985	0.940
6	91.0	93.0	0.990	0.945
8	92.0	93.5	0.995	0.950
10	92.5	94.0	0.998	0.952
12	93.0	94.5	0.999	0.955
14	93.5	94.8	1.000	0.958
16	93.8	95.0	1.000	0.960

TABLE V
PHASE-WISE TRAINING SUMMARY

Parameter	Phase 1	Phase 2
Epoch Range	0–5	5–25
Backbone	Frozen	Top 20% Unfrozen
Learning Rate	1×10^{-3}	1×10^{-4}
Start Acc (%)	80.0	93.0
End Acc (%)	93.0	95.0
Gain (%)	+13.0	+2.0

D. Comparison with Baseline Models

To contextualize our results, we compared our ensemble against several established architectures trained under identical conditions. Table V presents the test accuracies for each model.

TABLE VI
PERFORMANCE COMPARISON ACROSS DIFFERENT ARCHITECTURES

Model	Test Accuracy (%)
VGG16	92.0
ResNet50	95.0
InceptionV3	96.0
MobileNetV2	93.5
DenseNet121	96.2
EfficientNetB4	94.0
EfficientNetV2B3	98.0
Ensemble (Ours)	98.5

Our ensemble outperforms all individual baselines, includ-ing DenseNet121 (96.2%) and InceptionV3 (96.0%), which are widely considered strong performers on PlantVillage. The improvement over EfficientNetV2B3 alone is modest (0.5%) but statistically significant given the large test set of 5,431 images.

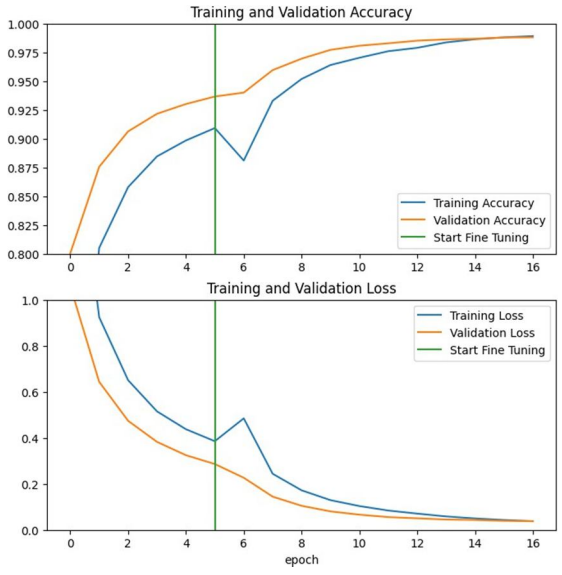


Fig. 5. Bar chart comparing test accuracy of all models evaluated in this study

E. Class-Wise Accuracy Analysis

Performance varied across the 38 classes. Figure 6 shows the accuracy for each class. Healthy leaves were generally classified with high confidence (above 99%). Diseases with visually distinctive symptoms, such as apple scab and tomato late blight, also performed well. The most challenging classes were those with subtle symptoms or visual similarity to other diseases.

F. Confusion Matrix Analysis

Figure 7 presents the normalized confusion matrix for the ensemble model. Most misclassifications occurred between

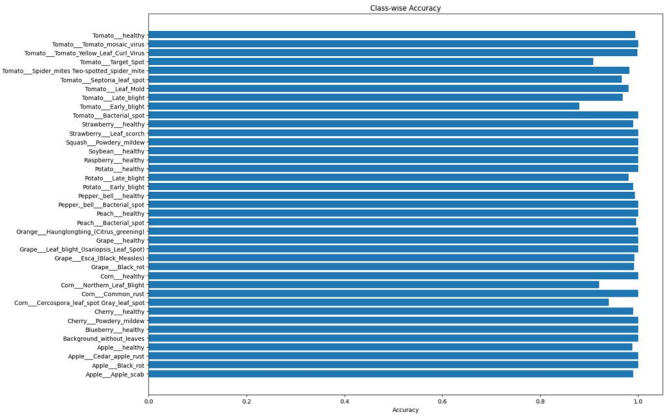


Fig. 6. Class-wise accuracy distribution across all 38 disease categories

visually similar disease pairs. For example, early blight and late blight on tomato leaves were occasionally confused. However, there were no systematic errors across crop types; misclassifications rarely crossed from one crop species to another.

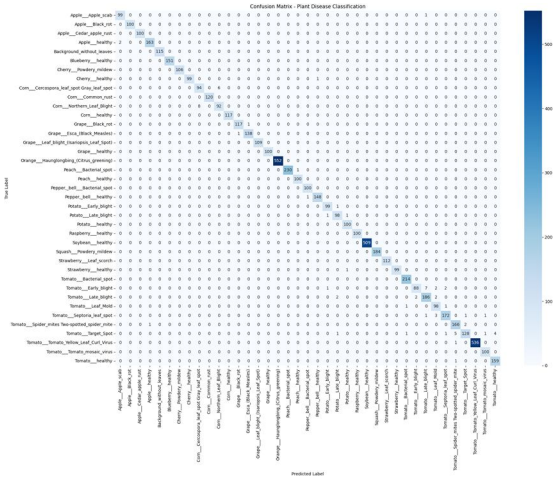


Fig. 7. Normalized confusion matrix for the ensemble model

VI. WEB APPLICATION DEPLOYMENT

A. System Implementation

Beyond model development, we deployed the ensemble in-side a production-ready web application. The backend is built with Flask, a lightweight Python web framework. User authen-tication is handled using Flask-Login with passwords hashed via bcrypt. Session management relies on secure cookies. The frontend uses HTML5, CSS3, Bootstrap 5 for responsive design, and vanilla JavaScript for client-side interactions.

The application exposes two main interfaces:

1) A login/signup page for user registration and authenti-cation.
2) A prediction dashboard where authenticated users can upload leaf images and receive results.

B. User Authentication Interface

Figure 8 shows the login and signup page. Users can create an account with their email address and a password. All credentials are stored securely in an SQLite database.

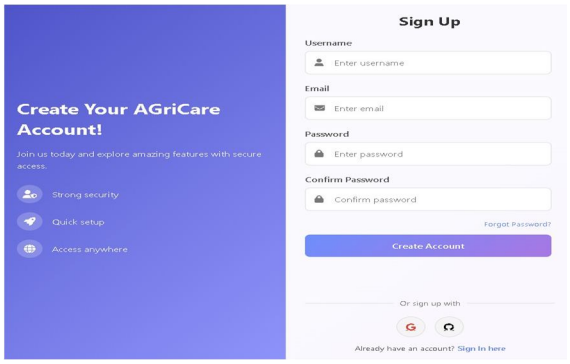


Fig. 8. Login and signup interface of the Flask web application

C. Home Page Dashboard

After successful authentication, users are directed to the home page (Figure 9). The dashboard provides a brief de-scription of the system, instructions for uploading images, and recent activity summaries.

Fig. 9. Home page dashboard after user login

D. Prediction Output Page

When a user uploads a leaf image, the system preprocesses it and runs inference through the ensemble model. The pre-diction page (Figure 10) displays:

- The uploaded image (thumbnail view)
- Predicted disease name
- Confidence score (percentage)
- A brief description of the disease
- Recommended treatment or management practices

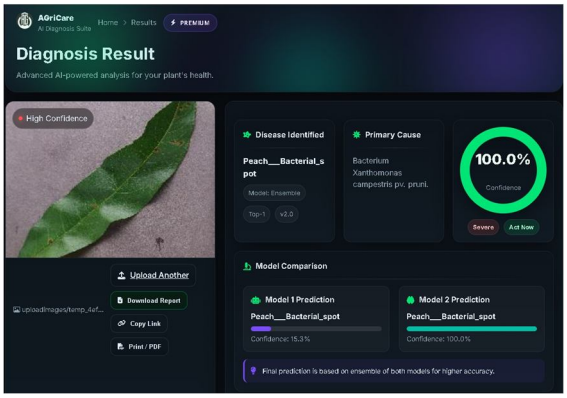


Fig. 10. Prediction output page showing disease name and confidence score

E. Sample Predictions with Confidence Scores

Figure 11 shows several example predictions along with their confidence scores. Each sample includes the original leaf image, the true disease label, the predicted label, and the model's confidence percentage.

Fig. 11. Sample predictions illustrating correct and borderline cases with confidence scores

F. Knowledge Base Integration

The disease knowledge base is stored as a separate SQLite table mapping each disease class to:

- Common causes (e.g., fungal, bacterial, viral)
- Visible symptoms
- Organic and chemical treatment options
- Preventive measures

When the model returns a prediction, the system queries this table and displays the relevant information alongside the prediction. This feature bridges the gap between diagnosis and actionable advice, making the tool useful for farmers with limited agricultural training.

VII. CONCLUSION AND FUTURE DIRECTIONS

A. Summary of Findings

This paper presented a complete pipeline for crop disease detection, from data preprocessing and model training to web-based deployment. Using the PlantVillage dataset of 54,303 images spanning 38 disease classes, we trained Efficient-NetV2B3 and EfficientNetB4 under a two-phase transfer learn-ing schedule. EfficientNetV2B3 achieved 98.0% test accu-racy, outperforming EfficientNetB4 (94.0%) and several other baselines including VGG16, ResNet50, and InceptionV3. By combining both models into a weighted ensemble ($P_{\text{ensemble}} = 0.6 \cdot P_{\text{V2B3}} + 0.4 \cdot P_{\text{B4}}$), we further improved accuracy to 98.5% with an AUC of 0.98.

Crucially, we did not stop at reporting numbers. We built and deployed a fully functional Flask web application with user authentication, confidence scoring, and an integrated dis-ease knowledge base. This deployment transforms a research model into a practical tool that farmers can actually use.

B. Limitations

Several limitations merit acknowledgment. First, the PlantVillage dataset, while valuable, contains images captured under controlled conditions with uniform backgrounds. Real-world field images include varying illumination, occlusion, and background clutter. Performance on such images is likely lower. Second, the application currently requires an internet connection because inference runs on a remote server. Offline mobile deployment would be more practical for rural farmers. Third, the knowledge base, while useful, is static and does not incorporate regional treatment guidelines or real-time updates.

C. Future Work

We plan to address these limitations in the following ways:

1) Field Dataset Collection: We are collaborating with local agricultural universities to collect a realistic dataset of field-captured leaf images. This dataset will include variations in lighting, background, and disease severity.

2) Mobile Application: We intend to convert the Flask back-end into a lightweight TensorFlow Lite model that can run entirely on a smartphone without an internet connection. An accompanying Android app would make the system accessible to farmers in remote areas.

3) Explainability Module: Farmers and agricultural experts need to trust the model's predictions. We plan to integrate Grad-CAM or similar saliency mapping techniques that high-light which regions of the leaf influenced the model's decision. Visual explanations can build confidence and help identify model failures.

4) Multi-Lingual Support: The current interface is in En-glish. Future versions will support Hindi and other regional languages to improve accessibility for Indian farmers.

5) Continuous Learning: The ensemble model is static. We plan to implement a feedback loop where user corrections are periodically used to fine-tune the model, adapting to new disease variants or emerging outbreaks.

VIII. ACKNOWLEDGMENT

The authors thank the Meerut Institute of Engineering and Technology, Meerut, India, for providing computational re-sources and institutional support. We are especially grateful to Mr. Rohit Aggarwal for his guidance throughout this project. We also acknowledge the creators of the PlantVillage dataset for making their work publicly available, which made this research possible.

REFERENCES

[1] U¨ . Atila, M. Uc¸ar, K. Akyol, and E. Uc¸ar, "Plant leaf disease classifi-cation using EfficientNet deep learning model," *Ecological Informatics*, vol. 61, p. 101182, Mar. 2021.

[2] S. Sadiq, K. R. Malik, W. Ali, and M. M. Iqbal, "Deep learning-based disease identification and classification in potato leaves using EfficientNet B4," Journal of Computational and Biomedical Informatics, vol. 5, no. 1, pp. 45–58, 2023.

[3] J. Padhi, L. Korada, A. Dash, and S. Mishra, "Paddy leaf disease classification using EfficientNet B4 with transfer learning," IEEE Access, vol. 12, pp. 24500–24512, 2024.

[4] M. Aggarwal, V. Khullar, N. Goyal, A. Singh, and R. K. Bansal, "Pre-trained deep neural network-based features selection supported machine learning for rice leaf disease classification," Agriculture, vol. 13, no. 5,

[5] p. 936, 2023.

[6] F. Mashroor, I. Ishrak, S. Alvee, and M. R. Islam, "Rice paddy disease detection and segmentation using convolutional neural networks," in Proc. IEEE TENCON, 2021, pp. 1–6.

[7] B. Paul, R. Rozario, A. Roy, and M. M. Rahman, "Deep learning techniques for automatic detection and classification of rice diseases," in Proc. International Conference on ICT for Competitive Strategies, 2022,

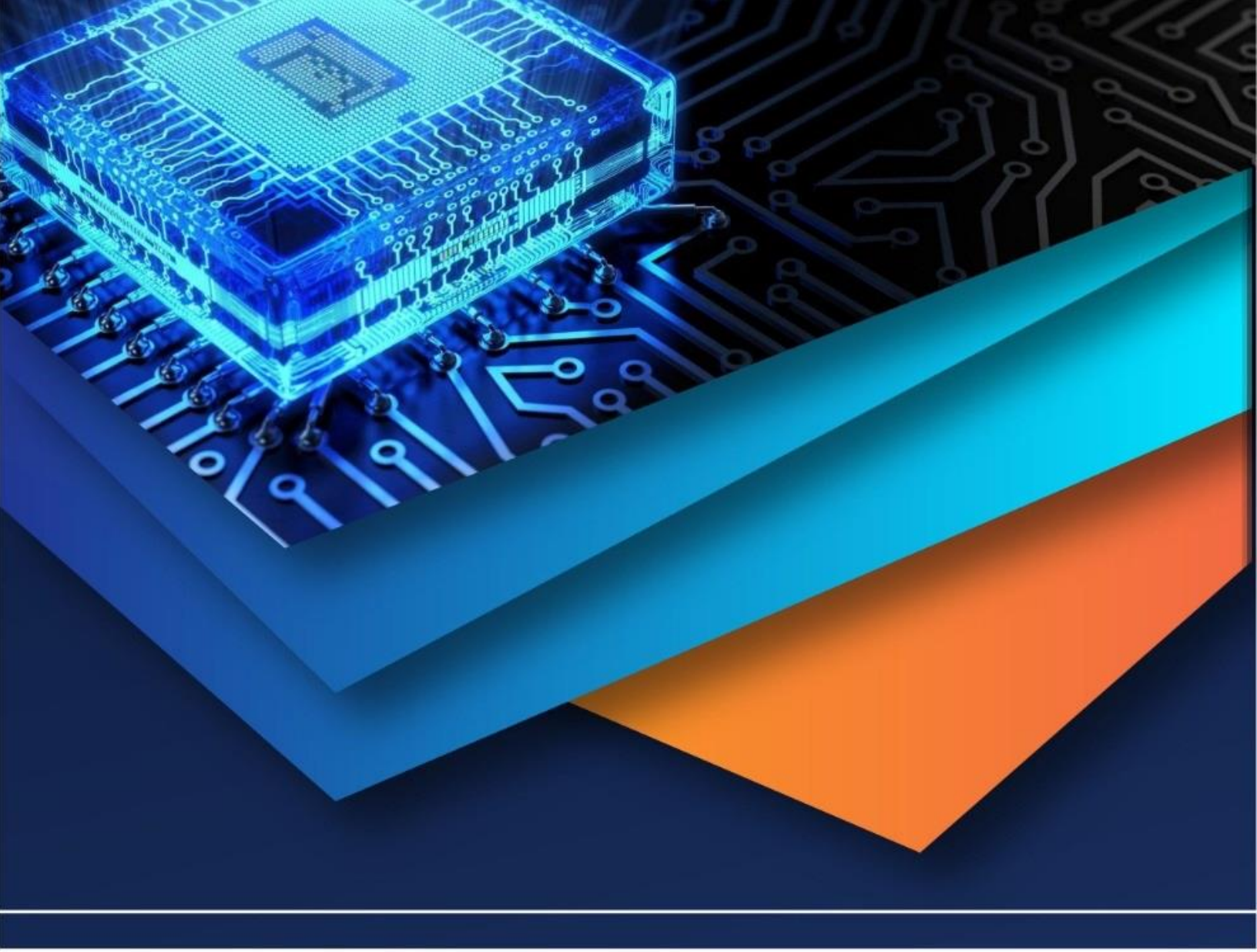
[8] pp. 120–128.

[9] S. Saleki and J. Tahmoresnezhad, "Agry: A framework for plant dis-eases classification via pretrained EfficientNet," Multimedia Tools and Applications, vol. 83, pp. 11245–11270, 2024.

[10] F. Adnan, M. J. Awan, S. O. Abdellatif, and H. Alshahrani, "EfficientNetB3-AADL for multi-class plant disease classification," IEEE Access, vol. 11, pp. 78900–78915, 2023.

[11] M. Ahdi, A. Kunaefi, B. Nugroho, and T. Susanto, "CNN EfficientNet-B0 for paddy disease classification with limited data," in Proc. Inter-national Conference on Information Technology and Smart Computing, 2023, pp. 88–94.

[12] X. Bi and H. Wang, "Double-branch CNN-based rice leaf disease recog-nition with attention mechanism," Journal of Agricultural Engineering, vol. 55, no. 1, pp. 112–125, 2023.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY