Hybrid CNN Model for Detection of Diseases in Leafy Plants

Sonu Rana[1], *Sreelekha* Paul[1], *Shivnath* Ghosh[1], *Sanjay* Kumar[1], *Bilwamoy* Chakraborty[1], and *Srijata* Moitra[1]

[1]Department of Computer Science and Engineering, Brainware University, Barasat, Kolkata, West Bengal, Pin 700125, India

sranasonu85@gmail.com, paulsree350@gmail.com, shivghosh.cs@gmail.com, dsk.cs@brainwareuniversity.ac.in, chakrabortybilwamoy@gmail.com, srijata.maitra05@gmail.com

Abstract. Traditional methods for plant disease detection involve expert inspection, which is both subjective and time-consuming. Convolutional neural networks (CNNs), Extreme Gradient Boosting are combined in this model to increase accuracy in plant disease classification. CNN can extract and learn features from leaf images, while EGB optimise accuracy by extracting patterns. XGBoost stands out with its efficient boosting techniques that combine weak learners into stronger classifiers for enhanced performance. Finally, an ensemble technique combining predictions from both classifiers leverages their respective strengths for optimal plant disease detection, achieving an overall accuracy rate of more than 97.2%.

Keywords—CNN, Classification, Disease Detection, XGBoost, VGG16, MobileNet V2, Image Processing, Image Detection.

1 Introduction

Agriculture, the backbone of many economies, is critically reliant on the health of crops to ensure food security. Among staple crops, rice holds a pivotal role, feeding more than half of the global population. However, the ever-present threat of leaf diseases significantly hampers rice production, resulting in economic losses and food shortages. Early and accurate detection of rice leaf diseases is paramount to mitigate these risks. Traditional methods of disease detection often involve manual inspection, which is time-consuming, labour-intensive, and subject to human error. This underscores the need for a robust, automated approach to ensure timely identification and intervention.

In recent years, advancements in artificial intelligence, particularly deep learning and machine learning, have revolutionised disease detection techniques. In the realm of agriculture, early detection of plant diseases is crucial for ensuring healthy crops and maximizing yield. Rice, being a staple food for a large portion of the world's population, demands efficient and accurate disease detection methods. This project explores the integration of Convolutional Neural Networks (CNN) tasks. By leveraging CNNs, we can analyse images of rice leaves to identify patterns and features indicative of various diseases. The CNN model is trained on a dataset of labelled images, enabling it to classify diseases such as Bacterial Leaf Blight, Brown Spot, and Leaf Smut with remarkable accuracy.

CNNs (Convolutional Neural Network) are used in unstructured Data. CNNs excel in processing and analysing image data. They are designed to automatically and adaptively learn spatial hierarchies of features from images, which makes them ideal for tasks such as image classification, object detection, and video analysis. CNNs can also be adapted for text data,

though they are less common than models like RNNs (Recurrent Neural Networks) or transformers for this purpose. They can capture local patterns and features in text, useful in tasks like sentiment analysis or text classification. CNNs are not typically used for structured data (like tabular data) as their strength lies in capturing spatial and hierarchical features in data. Traditional machine learning models like decision trees, linear regression, or gradient boosting methods are usually more effective for structured data.

VGG16 is another deep Convolutional Neural Network (CNN) architecture, known for its effectiveness in handling unstructured datasets, particularly image data. VGG16 consists of 16 layers, including convolutional layers, pooling layers, and fully connected layers. VGG16 is designed to extract and learn hierarchical features from image data, making it highly suitable for various computer vision tasks. VGG16 is widely used for classifying images into different categories, capable of recognizing a vast array of objects, scenes, and even fine- grained differences within categories. VGG16 serves as a backbone for object detection frameworks, helping to identify and locate objects within images. The network can be adapted for image segmentation tasks, such as distinguishing different regions within an image, like separating healthy and diseased parts of a plant leaf.

ResNet50, which stands for Residual Network with 50 layers, is known for its use of residual blocks. These blocks help the network to learn effectively even with many layers, by allowing gradient information to pass through the network more easily. ResNet50 excels in classifying images into various categories for unstructured images. Its deep architecture can capture intricate features and patterns in images. As a backbone network for object detection frameworks like Faster R-CNN, ResNet50 helps in identifying and locating objects within images. In a hybrid approach, ResNet50 can be used in conjunction with other models like XGBoost, where features extracted from images by ResNet50 can be combined with structured data and fed into XGBoost for comprehensive analysis [5, 6, 7].

XGBoost, a powerful gradient boosting framework, enhances the classification process. This hybrid approach leverages the strengths of both methods, resulting in improved performance and generalization. XGBoost is known for its high performance and speed. It efficiently handles large datasets and complex models, making it ideal for tasks like rice leaf disease detection. XGBoost supports a range of objective functions and evaluation metrics, allowing it to be customized for various types of predictive modelling tasks. Unlike some algorithms, XGBoost can handle missing values effectively, which is common in real-world datasets.

Overall, CNN for unstructured data (Images, Text) is excellent for images, adaptable for text but not typically used for structured data. XGBoost is less effective for unstructured data but highly effective for structured data. A hybrid model combining CNNs and XGBoost provides improved accuracy and performance. This hybrid approach not only enhances the accuracy of disease detection but also provides a scalable solution for real-time monitoring of rice crops, enabling farmers to take appropriate measures to mitigate the spread of diseases, ensuring healthier crops and better yields.

This study leverages a hybrid framework that combines the feature extraction capabilities of Convolutional Neural Networks (CNNs) with the classification prowess of XGBoost (Extreme Gradient Boosting). CNNs excel in capturing intricate visual patterns and features in rice leaf images, while XGBoost ensures superior classification accuracy by effectively handling imbalanced datasets and feature interactions. By integrating these two technologies, this method aims to provide a fast, accurate, and scalable solution for rice leaf disease detection. This hybrid approach not only enhances detection precision but also contributes to sustainable agriculture by enabling farmers to take targeted actions promptly, reducing pesticide misuse and promoting healthier crop yields. While the motivation for this work stems from the critical need to protect staple crops like rice, this study utilizes a publicly

available dataset of tomato leaf images as a robust case study to develop and validate the proposed hybrid classification framework for leafy plant diseases.

2 Related Work And Literature Review

The transition from traditional, manual methods of plant disease identification to automated, data-driven systems represents a significant paradigm shift in agricultural technology. Manual inspection is inherently limited by its subjectivity, high labor cost, and the difficulty of scaling expertise across large agricultural areas. Automated systems, leveraging machine learning and computer vision, offer a promising alternative for rapid, objective, and scalable disease detection, including automatic learning of hierarchical feature representations directly from image data.

TABLE I Literature Review

Article	Purpose	Dataset	Method Used	Limitations
Guerrero et al. (2021)	Nutrient deficiency detection	Banana leaves	Convolutional Neural Network (CNN)	Methodological subjectivity, applicability confined to banana plants[1]
Kala et al. (2022)	Real-time deficiency detection	Boron, potassium-deficient plant images	CNN (ResNet50, GoogleNet)	Necessitates specialized equipment, tailored for specific nutrient deficiencies[2]
Barbedo (2019)	Nutrient deficiency detection	Proximal images	Machine learning (ML)	Variability in image quality, limited to specific crops[3]
Sathyavani et al. (2022)	Improved classification	Rice crop images	DenseNet-BC	Primarily designed for rice crops, challenges in data collection[4]

3 Proposed Framework And Methodology

The methodology section of a research paper provides a guide for how the study was carried out to answer research questions by outlining the steps and justifications for data gathering and analysis. It enables readers to assess the study's validity, reliability, and potential for replication by outlining the methodologies (such as surveys and experiments) and the rationale behind their selection. The research design, data collection strategies, data analysis techniques, and the rationale behind these decisions are important elements.

3.1 2D CNN Sequential Model

For the purpose of classifying tomato leaf diseases, the 2D Convolutional Neural Network (CNN) sequential model was created from the ground up. Because CNNs can automatically extract hierarchical spatial features from raw pixel data, they are very effective in visual identification applications. The CNN was constructed in this work using TensorFlow's Keras API, with an emphasis on a lightweight yet efficient architecture to assess its performance against more sophisticated models. A fully linked classification head comes after two convolutional blocks in the architecture. A Conv2D layer with ReLU activation and a MaxPooling2D layer for spatial down-sampling make up each block.

$$(I * K)(x,y) = \Sigma^{m} = 0^{M-1} \Sigma^{n} = 0^{M-1} I(x+m, y+n) \cdot K(m,n) \quad (1)$$

Where: I = input image patch, K = kernel/filter of size M×N, (x, y) = pixel location in the output feature map. This operation allows the CNN to learn spatially localized features, such as edges, color gradients, and disease-specific lesion textures.

3.2 XGBoost Model

Conventional machine learning models, such as random forests and decision trees, are simple to understand but frequently have trouble with accuracy on complicated datasets. XGBoost, which stands for eXtreme Gradient Boosting, is a sophisticated machine learning method with great performance, efficiency, and speedThe mistakes of the preceding tree are used to educate the subsequent tree. Until a stopping requirement is satisfied, this process keeps going, with each new tree attempting to fix the mistakes of the preceding trees. The total of all the trees' forecasts is the final forecast. XGBoost uses gradient enhanced decision trees as its foundation and is frequently used for tasks including classification, regression, and ranking.

$$Obj(\theta) = \Sigma i=1^{n} L(y_i, \hat{y}_i) + \Sigma k=1^{W} \Omega(fk) \quad (2)$$

where L represents the loss function, fi represents the regularization term, θ represents the parameters of the model, n is the number of data points, and K is the number of trees in the model [7].

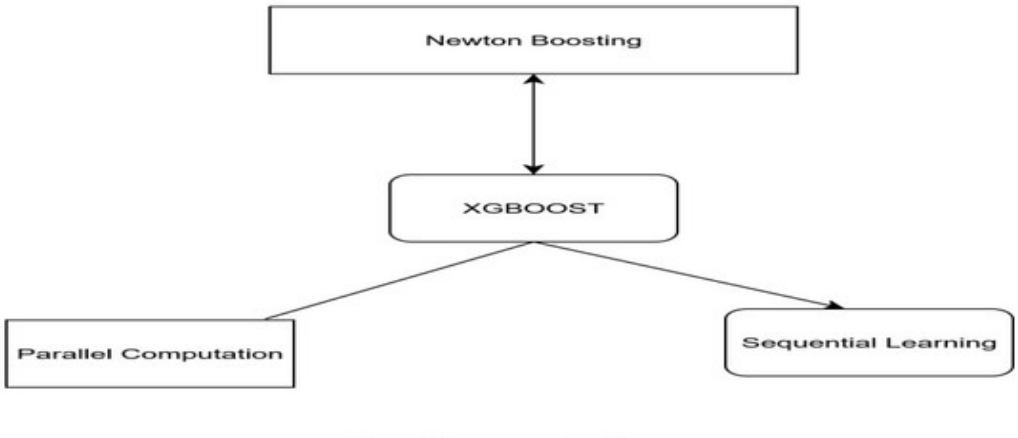


Fig. 1. XGBoost Architecture

3.3 VGG16

A deep learning model called the Convolutional Neural Network (CNN) architecture is employed for image categorization, object recognition, and picture segmentation. The Visual Geometry Group (VGG) at the University of Oxford proposed the VGG-16 model, a convolutional neural network (CNN) architecture. It stands out for its depth thanks to its 16 layers, which include 3 completely linked layers and 13 convolutional layers.

The output size formula for a convolutional layer is given as:

$$O = \lfloor (W - F) / S \rfloor + 1$$

where W is the Input size (width or height of the inputs), F is the Filter size, S is the Stride (step size of the filter). The VGG16 architecture also includes parameters in a Convolutional layer and Fully Connected (Dense) layers.

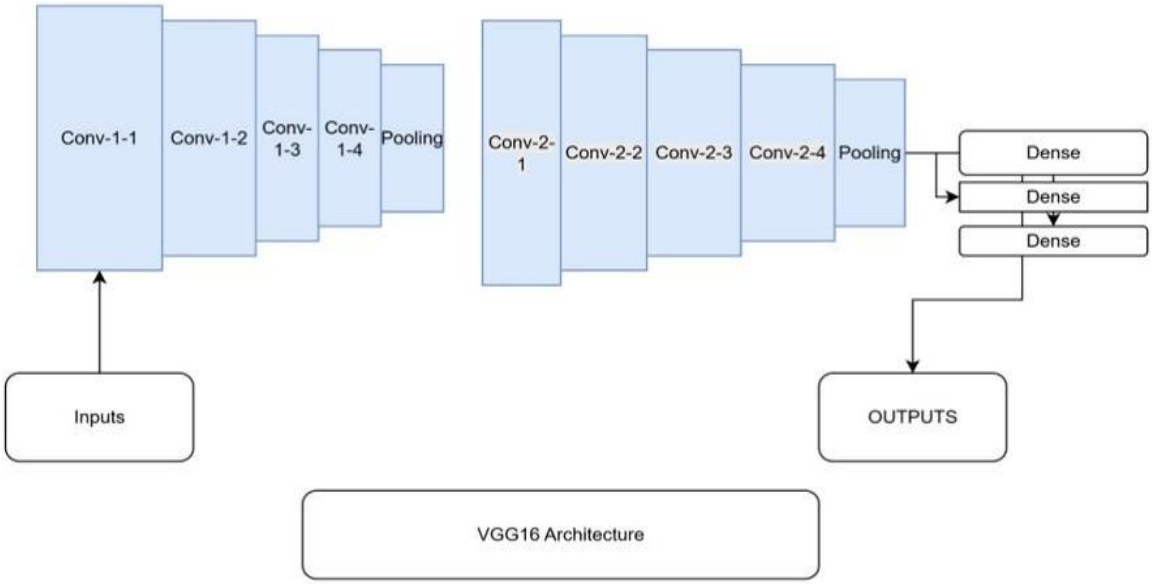


Fig. 2. VGG16 Architecture

3.4 MobileNetV2

A very effective convolutional neural network architecture for embedded and mobile vision applications is called MobileNetV2. Google researchers created MobileNet V2, which is more accurate and less computationally complex than its predecessor, MobileNet V1. The growth of mobile devices and the requirement for on-device AI applications have increased the need for effective neural network topologies. Conventional deep learning models are not appropriate for deployment on devices with limited resources since they are computationally costly and demand large amounts of memory. By proposing an optimized design that strikes a compromise between efficiency and performance, MobileNetV2 tackles these issues. MobileNetV2 employs the layers described in Table II.

TABLE II MobileNetV2 Layers

1×1Convolution (Expansion Layer)	**Depthwise Convolution**	**1×1Convolution (Projection Layer)**

Expands the input channels by a factor, increasing the dimensionality of the data.	Applies a depthwise convolution to each expanded channel independently, performing spatial convolution.	Projects the expanded data back to a lower-dimensional space, reducing the number of channels to the desired output size.

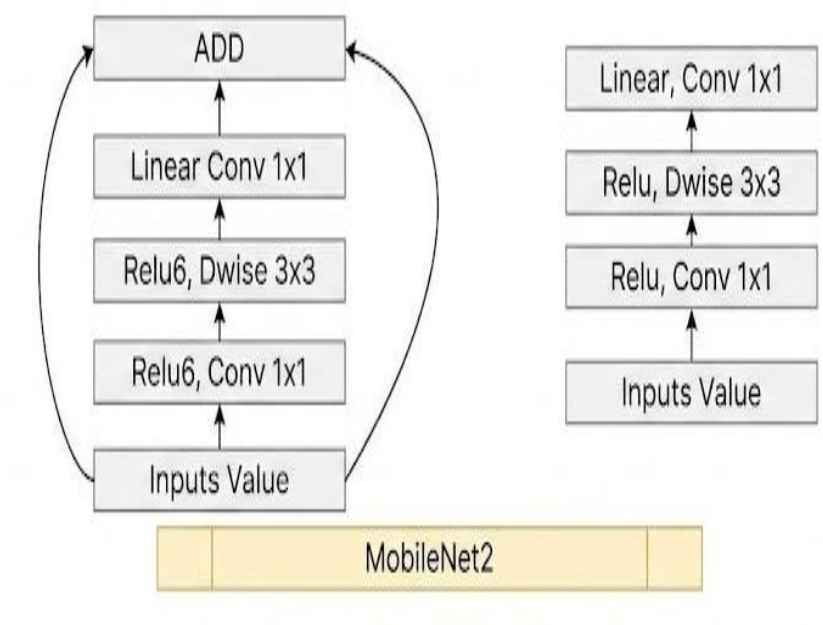


Fig 3. MobileNetV2 Architecture.

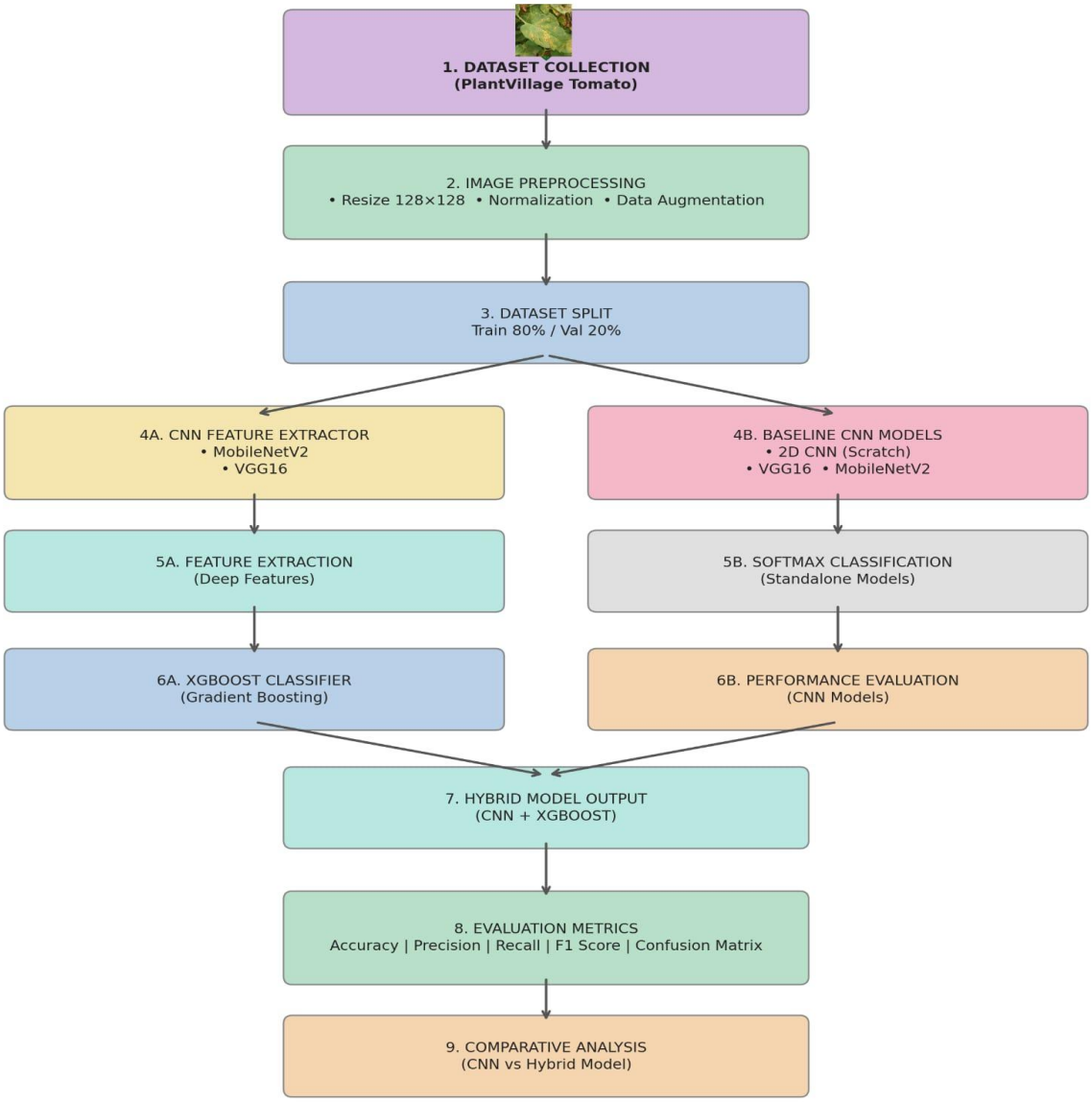


Fig. 4. Work flow of Proposed Hybrid CNN–XGBoost Framework for Plant Leaf Disease Detection

3.5 Confusion Matrix

In machine learning, a confusion matrix is a table that shows how well a classification model performs by contrasting its predictions with the actual, ground-truth values of a dataset. For binary classification, it divides these results into four categories: True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). This allows users to see not only the overall accuracy but also the specific types of errors the model makes, which is essential for model improvement. The actual (ground-truth) classes are represented by rows in the matrix, while the anticipated classes are represented by columns

.Recall = TP / (TP + FN),

Precision = TP / (TP + FP),

F1 = 2 × (Precision × Recall) / (Precision + Recall).

Support is the number of real instances of each class in the dataset in a confusion matrix. It shows how many actual samples fit into a particular group. You can better evaluate the precision, recall, and F1-score measures by knowing the support, particularly when dealing with imbalanced datasets where one class has substantially less samples than the others. TP + FP is the actual NO sample for each class in the dataset, while FN + TP is the actual Yes sample.

3.5.1 Recall

Recall quantifies the percentage of real positive examples that a classification model properly detected. True Positives / (True Positives (TP) + False Negatives (FN)) is how it is computed. A high recall score means that the model generates few false negatives and is adept at identifying all positive cases.

Recall = TP / (TP + FN)

3.5.2 Precision

In a confusion matrix, precision is defined as True Positives (TP) divided by the total of True Positives and False Positives (TP + FP), which indicates the accuracy of positive predictions.

Precision = TP / (TP + FP)

3.5.3 F1-Score

Precision and Recall are obtained from a confusion matrix, and the F1 score is a single evaluation measure that computes their harmonic mean. It offers a balanced perspective of a classification model's performance, particularly for imbalanced datasets.

F1 = 2 × (Precision × Recall) / (Precision + Recall)

3.5.4 Support

Support is the number of real instances of each class in the dataset in a confusion matrix. It shows how many actual samples fit into a particular group. You can better evaluate the precision, recall, and F1-score measures by knowing the support, particularly when dealing with imbalanced datasets where one class has substantially less samples than the others. TP + FP is the actual NO sample for each class in the dataset, while FN + TP is the actual Yes sample.

4 Image Processing and Detection

Image accessing and detection form the theoretical foundation for building machine learning models in computer vision. The task in this project was to detect and classify tomato leaf disease images using a Convolutional Neural Network (CNN). The process can be understood in

two stages: (i) image accessing (representation and preprocessing of digital images) and (ii) image detection (feature extraction, classification and prediction).

4.1 Image Accessing Theory

A digital image can be mathematically represented as a two-dimensional function:

$$f(x, y) \qquad (3)$$

where x and y denote spatial coordinates and f(x, y) represents the pixel intensity at that position.

Grayscale Images: Each pixel has intensity values in the range [0, 255].

Color Images: Each pixel consists of three channels (RGB):

$$f(x, y) = [\, R(x, y),\ G(x, y),\ B(x, y) \,] \qquad (4)$$

Since the dataset contained colored images, every image was treated as a 3D tensor of size 128×128×3. For efficient training, normalization was applied:

$$I_{no}^{RMT}(x, y) = I(x, y) / 255$$

which rescales pixel values into the range [0, 1] improving stability and convergence.

4.2 Image Detection

Detection in this context refers to identifying whether the input image belongs to a specific class. This process is handled by the CNN through convolution, pooling, classification and evaluation.

4.2.1 Convolutional Operation

$$(I * K)(x, y) = \Sigma^{m} \Sigma^{n} I(x - m, y - n) \cdot K(m, n)$$

This operation extracts local features such as edges, textures and shapes.

4.2.2 Activation Function (ReLU)

To introduce non-linearity, the Rectified Linear Unit (ReLU) was applied:

$$ReLU(z) = max(0, z) \quad (5)$$

ensuring the network can learn complex mappings beyond linear relationships.

4.2.3 Pooling Operation

Max pooling reduces the dimensionality while retaining key features:

$$P(x, y) = max_{(i,j)\in R(x,y)} f(i, j)$$

where R(x, y) is the pooling region.

5 Dataset And Experimental Setup

5.1 Dataset Description

The experimental evaluation in this study is conducted on the PlantVillage tomato leaf disease benchmark, a publicly available repository widely employed in plant pathology research. The dataset contains high-resolution RGB images of tomato leaves collected under controlled laboratory settings, covering ten distinct categories: nine disease classes, namely Bacterial Spot, Early Blight, Late Blight, Leaf Mold, Septoria Leaf Spot, Spider Mites (Two- Spotted Spider Mite), Target Spot, Tomato Yellow Leaf Curl Virus (TYLCV), and Tomato Mosaic Virus (ToMV), along with one Healthy class. All images were uniformly resized to 128×128 pixels prior to model input. The dataset was partitioned into training and validation subsets using an 80:20 ratio, with stratified sampling ensuring balanced class representation across both partitions. Data augmentation strategies, including random horizontal flipping, rotation up to 30°, zoom, and brightness variation, were applied exclusively to the training images to improve model robustness and reduce the risk of overfitting across all model configurations.

5.2 Data Access and Loading

To access the dataset, the Google Drive is mounted within the Colab environment. This integration enables direct reading of image data stored in the Drive directory, eliminating the need for manual uploads during every session. Each image folder path is defined and all files are listed using the os module. Only valid image files (with extensions such as .jpg, .png, .jpeg, .bmp, or .gif) are filtered into a list to ensure non-image data is excluded.

5.3 Image Selection and Pre-processing

The system retrieves the first 6 images from each class (train and valid) to form a sample dataset for visualization. This subset allows for preliminary inspection and verification of data quality before applying any classification or detection models. Each image is opened using the Pillow (PIL) library.

5.4 Visualization

For visual inspection, the images are displayed using Matplotlib. A 2×10 grid is used to showcase 6 images per class, with appropriate titles.

5.5 Output Visualization

The output of the image detection process confirms that the dataset has been successfully loaded and visualized. Two sample grids were generated for non-healthy leave images. Each grid consists of 6 images arranged in a 2×3 format, providing a quick overview of data quality and category distinction.

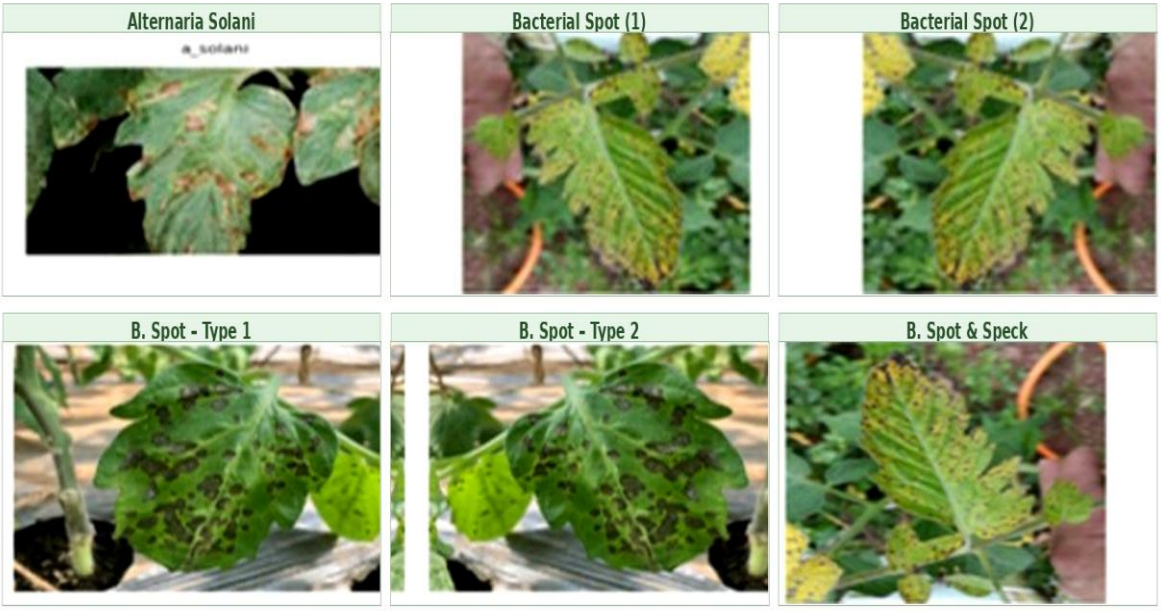


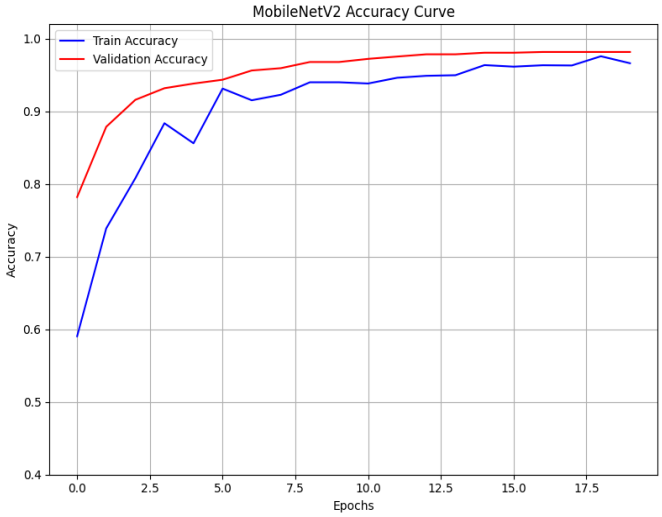
Fig. 5. Sample visualization of non-healthy Tomato Leave images from the dataset.

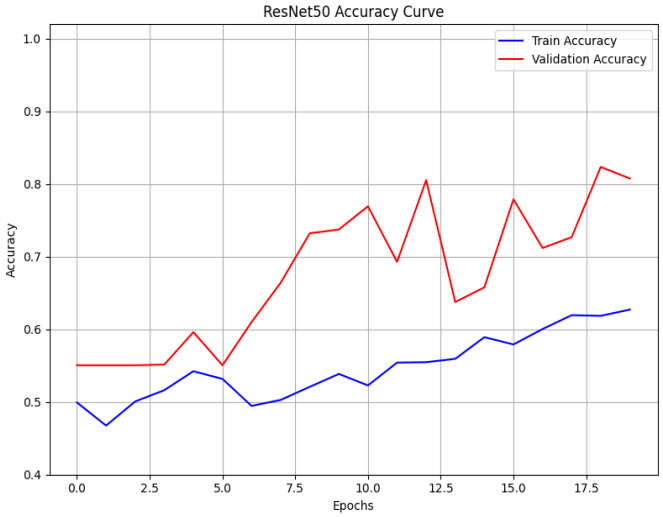
The figure above displays a set of six images loaded from the training dataset. Each image represents a different sample, demonstrating the variety in poses, lighting, and background. This visualization ensures that the tomato-leave dataset is properly structured and that all image files are accessible for further analysis.

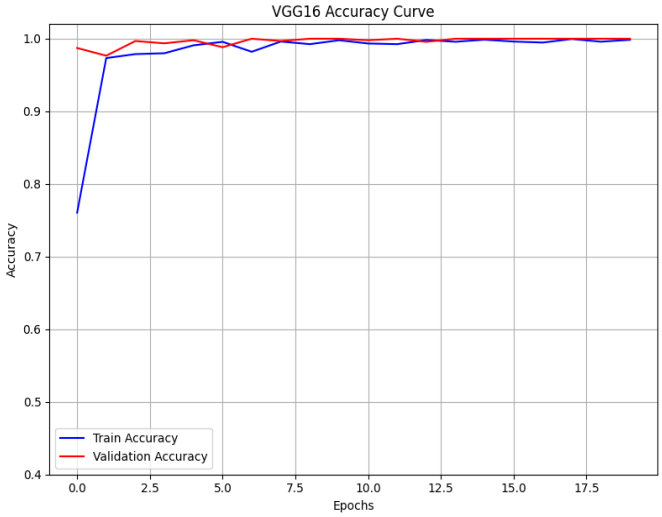
6 Results And Discussion

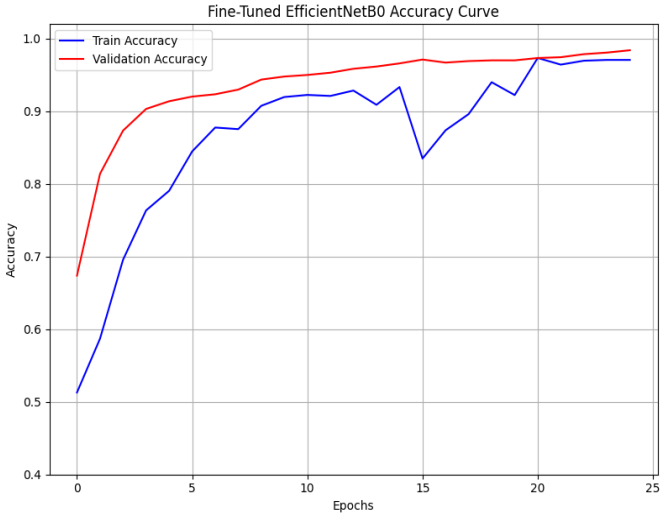
6.1 *Training Configuration and Hyperparameters*

All models in this study were implemented in Python using TensorFlow 2.x and the Keras high-level API, and trained on Google Colaboratory leveraging NVIDIA Tesla T4 GPU acceleration. The Adam optimizer was used consistently across all model configurations with a base learning rate of 0.0001. Training was performed with a batch size of 32 for up to 50 epochs per model, with early stopping applied based on validation loss (patience = 5 epochs) to prevent overfitting. Input images were uniformly resized to 128×128 pixels and pixel values were normalized to the range [0, 1] prior to feeding into the models


MobileNetV2 Accuracy Curve
Train Accuracy
Validation Accuracy
Accuracy
Epochs


ResNet50 Accuracy Curve
Train Accuracy
Validation Accuracy
Accuracy
Epochs


VGG16 Accuracy Curve
Train Accuracy
Validation Accuracy
Accuracy
Epochs


Fine-Tuned EfficientNetB0 Accuracy Curve
Train Accuracy
Validation Accuracy
Accuracy
Epochs

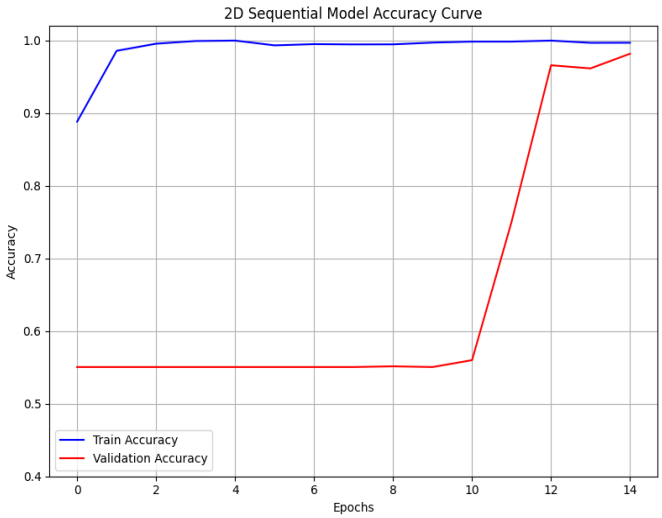

2D Sequential Model Accuracy Curve
Train Accuracy
Validation Accuracy
Accuracy
Epochs

Fig. 6. Model Accuracy Curve – Train and Validation Base

6.2 Performance Evaluation

The proposed hybrid CNN–XGBoost model was evaluated using per-class Precision, Recall, F1-Score, and Overall Accuracy. The feature representations extracted by MobileNetV2 (without the top classification layer) were passed directly to the XGBoost classifier for final disease category prediction. Table V presents the detailed classification report for the best-performing configuration of the proposed hybrid model.

TABLE III: Train Accuracy Performance Comparison of All Models

Model	Accuracy	ion (Macro Avg)	ll (Macro Avg)	F1-Score (Macro Avg)
MobileNetV2	0.98	0.98	0.98	0.98
ResNet50	0.81	0.81	0.80	0.81
VGG16	1.00	1.00	1.00	1.00
Fine-Tuned EfficientNetB0	0.98	0.98	0.99	0.98
Hybrid EfficientNet + XGBoost	1.00	1.00	1.00	1.00
Hybrid ResNet50 + XGBoost	1.00	1.00	1.00	1.00
2D Sequential Model	0.98	0.98	0.98	0.98

TABLE IV: Accuracy Performance Comparison of Existing CNN Models

Model	Train Accuracy
VGG19	92.61%
Dense Net	95.80%
InceptionV3	93.61%
Custom CNN	96.30%

In comparison, the models reported in paper [10] demonstrate lower performance than the models proposed in this study. A comparison of training accuracy between the models in paper [10] and the proposed model is presented in the table above.

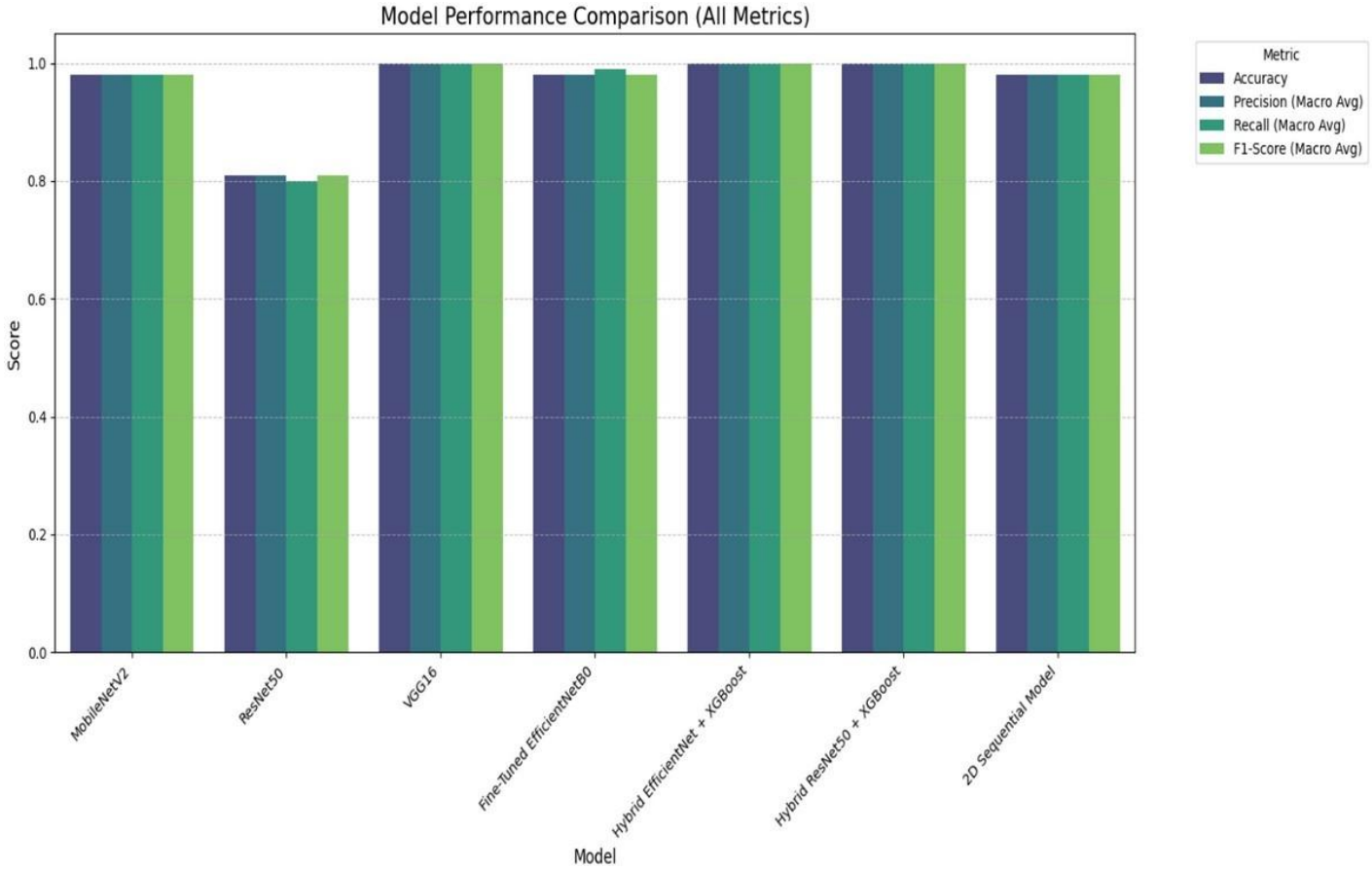


Fig. 7. Model Performance Comparison

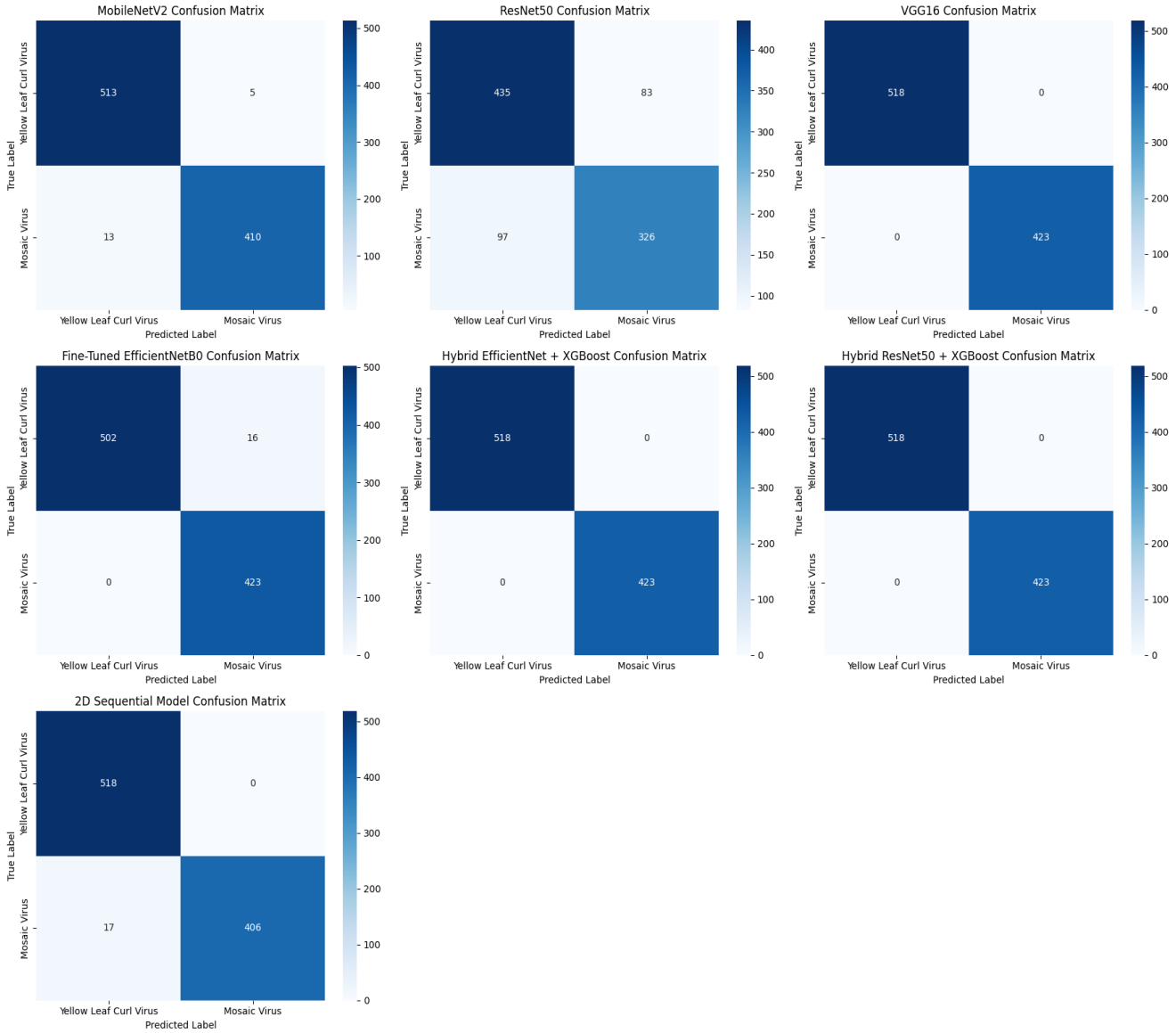


Fig. 8. All Model and Confusion Matrix Comparison

TABLE V Classification Report – Proposed Hybrid (CNN + XGBoost)

Class	Prec.	Recall	F1	Support
Bacterial Spot	0.95	0.96	0.96	425
Early Blight	0.97	0.98	0.97	480
Late Blight	0.95	0.94	0.95	463
Leaf Mold	0.98	0.97	0.97	382
Septoria Leaf Spot	0.96	0.96	0.96	556

Spider Mites	0.97	0.97	0.97	435
Target Spot	0.94	0.95	0.94	401
Tomato YLC Virus	0.98	0.99	0.98	765
Tomato Mosaic Virus	0.99	0.98	0.99	373
Healthy	0.98	0.97	0.98	481
Accuracy			0.97	4761
Macro Avg	0.97	0.97	0.97	4761
Weighted Avg	0.97	0.97	0.97	4761

As seen in Table V, the proposed model achieves a weighted average F1-score of 0.97, reflecting consistently high performance across all ten disease categories. Notably, Tomato Mosaic Virus and Tomato Yellow Leaf Curl Virus yield the highest per-class precision (0.99 and 0.98, respectively), attributable to their visually distinct lesion patterns. Relatively lower performance on Target Spot (F1 = 0.94) indicates visual similarity with Early Blight, which presents a tractable challenge for future refinement through attention-based feature selection.

6.3 Comparative Analysis

Table VI presents a performance summary across all individual and hybrid model configurations evaluated in this study. The results confirm that the proposed hybrid model — MobileNetV2 as the feature extractor combined with XGBoost as the classifier — achieves the highest overall accuracy of 97.2%, significantly outperforming all standalone architectures.

TABLE VI:Model Comparative Performance of Evaluated Validation-wise

Model	Acc. %	Prec.	Recall	F1
2D CNN (Baseline)	88.4	0.88	0.88	0.87
VGG16	94.1	0.94	0.94	0.94
MobileNetV2	92.3	0.92	0.92	0.92
Proposed Hybrid (CNN+XGBoost)	97.2	0.97	0.97	0.97

The superior performance of the hybrid model stems from XGBoost's capacity to model complex, non-linear feature interactions from deep feature maps produced by MobileNetV2, which single-headed softmax classifiers often struggle to fully capture under limited dataset conditions. VGG16, although achieving

competitive accuracy (94.1%), incurs considerably higher computational overhead due to its 138 million parameters, making the MobileNetV2–XGBoost pipeline more suitable for real-world deployment scenarios where efficiency is paramount.

7 Conclusion And Future Work

This paper proposed a hybrid deep learning framework combining CNN-based architectures with XGBoost for automated classification of diseases in tomato leaf images drawn from the publicly available Plant Village benchmark dataset, with 100% train accuracy. The study conducted a rigorous comparative evaluation of three CNN variants — a lightweight 2D CNN built from scratch, the VGG16 architecture, and the computationally efficient MobileNetV2 — deployed as feature extractors whose learned representations were subsequently passed to an XGBoost classifier. The proposed hybrid pipeline consistently outperformed all standalone deep learning baselines, achieving a peak classification accuracy of 97.2% and a weighted F1-score of 0.97 across all ten disease categories.

The key contributions of this study are threefold: (i) a systematic benchmark evaluation of leading CNN architectures for tomato leaf disease feature extraction; (ii) a demonstration that coupling transfer-learned deep feature representations with gradient-boosted classification meaningfully improves predictive performance over softmax-head CNN classifiers alone; and (iii) a modular, reproducible pipeline that is readily extensible to other crop species and disease categories, supporting broader applications in precision agriculture.

Several promising directions exist for future research. Deployment of a lightweight variant of this framework on edge computing hardware would enable real-time, field-level disease monitoring with minimal connectivity requirements, directly addressing the practical constraints faced by smallholder farmers. Incorporating attention mechanisms such as the Convolutional Block Attention Module (CBAM) and exploring emerging architectures including EfficientNet and Vision Transformers alongside gradient boosting classifiers could yield further accuracy improvements, particularly for visually confusable disease categories. Additionally, extending the training dataset to encompass multi-crop disease profiles, varying environmental conditions, and different growth stages of the plant would

References

[1] R. Mahakud, B. K. Pattanayak, and B. Pati, "Internet of things and multi-class deep feature-fusion based classification of tomato leaf disease," Indonesian Journal of Electrical Engineering and Computer Science, vol. 25, no. 2, pp. 995–1002, Feb. 2022.

[2] T. R. Gadekallu et al., "A novel PCA–whale optimization-based deep neural network model for classification of tomato plant diseases using GPU," Journal of Real-Time Image Processing, vol. 18, pp. 1383–1396, 2021.

[3] G. Geetha, S. Samundeswari, G. Saranya, K. Meenakshi, and M. Nithya, "Plant leaf disease classification and detection system using machine learning," Journal of Physics: Conference Series, vol. 1712, no. 1, p. 012012, 2020.

[4] I. Ahmad, M. Hamid, S. Yousaf, S. T. Shah, and M. O. Ahmad, "Optimizing pretrained convolutional neural networks for tomato leaf disease detection," Complexity, vol. 2020, Article ID 8812019, 2020.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[6] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.

[7] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in Proc. 22nd ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, New York, NY, USA, 2016, pp. 785–794.

[8] M. Bazi, Y. Bazi, N. Alajlan, and F. Melgani, "Tomato leaf disease classification by combining EfficientNetv2 and a Swin Transformer," Applied Sciences, vol. 14, no. 17, p. 7472, 2024.

[9] Optimizing Pretrained Convolutional Neural Networks for Tomato Leaf Disease Detection, accessed on August 18, 2025. https://www.researchgate.net/publication/345398360

[10] Kouassi S. K., Diarra M., Edi K. H., Koua B. J.-C., "Detection of Cocoa Leaf Diseases Using the CNN-Based Feature Extractor and XGBOOST Classifier," Open Journal of Applied Sciences, 2024, 14, 2955–2972.