International Journal of Science, Technology, Engineering and Mathematics
Volume 5 Issue 2 June 2025
DOI: https://doi.org/10.53378/ijstem.353186

Edge-optimized multimodal cross-fusion architecture for efficient crop disease detection

[1]Thomas Kinyanjui Njoroge, [2]Kelvin Mugoye Shindu & [2]Rachael Kibuku

Abstract

Accurate and timely crop disease detection is critical for reducing agricultural losses and ensuring food security in low-resource settings. Traditional diagnostic methods, such as manual inspections, are often inefficient and error-prone. Existing deep learning models (e.g., ResNet50, Inception V3) struggle with computational inefficiency and poor generalizability in real-world farming contexts. This study proposes a lightweight multimodal fusion model integrating EfficientNetV2 and MobileNetV2, optimized for edge deployment. The architecture leverages compound scaling and feature fusion to recognize subtle disease patterns, and it was fine-tuned on a globally diverse dataset (PlantVillage and field-collected leaf images). The proposed model achieved state-leading metrics (99.0% accuracy, 0.993 precision, 0.990 F1-score, AUC = 0.999997), outperforming benchmarks like ShuffleNet and DenseNet50 (ranked 2nd–6th). Statistical validation via the Kruskal-Wallis test confirmed significant performance differences across models ($H=614.90$, $p=1.4237e{-}129$), with Bayesian analysis showing a 100% superiority probability over DenseNet50. Notably, the model exhibited the lowest confidence variance (0.000012) compared to alternatives (0.000014–0.000032), demonstrating unmatched prediction stability. Deployment on low-end mobile devices posed challenges such as computational constraints and offline usability. However, the TensorFlow Lite-powered mobile app addressed these limitations, offering real-time, offline disease classification with 0.094-second inference latency on devices with ≤2GB RAM. Validated on 249 unseen field images (95.98% accuracy), this solution bridges the gap between high-performance deep learning and real-world agricultural needs, empowering smallholder farmers with an accessible and scalable tool.

Keywords: *crop disease detection, multimodal fusion model, transfer learning, edge computing, EfficientNetV2, MobileNetV2*

Article History:

***Received**: January 28, 2025* ***Revised**: March 9, 2025*
***Accepted**: March 11, 2025* ***Published online**: May 15, 2025*

Suggested Citation:
Njoroge, T.K., Shindu, K.M. & Kibuku, R. (2025). Edge-optimized multimodal cross-fusion architecture for efficient crop disease detection. *International Journal of Science, Technology, Engineering and Mathematics*, 5(2), 1-37. https://doi.org/10.53378/ijstem.353186

About the authors:
[1]Corresponding author. Department of Computer Science and Informatics, Karatina University, Kenya. Email: tnjoroge@karu.ac.ke
[2]Software Development & Information Systems (SD&IS) Department, School of Technology, KCA University, Kenya.

1. Introduction

Recent technological advancements have enhanced disease detection through machine learning (ML), computer vision, and deep learning (DL) innovations, transforming the domain and making disease identification precise and effective. Kaleem et al. (2021) demonstrated that these modern techniques use sophisticated algorithms to analyze plant leaf photographs, enabling more accurate detection of disease patterns. Given the challenges of distinguishing between similar plant diseases, integrating image processing with ML and computer vision is crucial for achieving reliable diagnoses. DL has made remarkable strides in image classification, achieving notable accuracy and speed in detecting and classifying crop leaf diseases. Despite its advancements, Abdu et al. (2020) illustrated that DL methods face challenges, like the need for extensive training data. Transfer learning (TL) has been demonstrated to be a powerful answer to these limitations, enhancing learning efficiency by leveraging knowledge from pre-trained models. TL also helps reduce overfitting, improve interpretability, and lower computational costs, which is essential for practical model training and performance. Mohammed and Yusoff (2023) also concur that TL models are lightweight and compatible with low-end graphics processing units (GPUs), which leads to shorter training times and lower computational costs. Moreover, TL reduces the likelihood of models making incorrect assumptions about new and unfamiliar data, enabling deep learning models to perform more efficiently. Zhao et al. (2024) explain that collecting and organizing large image databases for training machine learning models comes with significant costs, both in time and finances. Static methods, which rely on fixed datasets and algorithms, often struggle with adaptability and can lead to decreased performance when faced with new or varied data.

As Nguyen et al. (2023) also observe, these static approaches may become outdated as conditions change or as new types of plant diseases emerge. There is a growing need for dynamic detection methods that adapt to evolving data and situations to address these issues. Data regularization and augmentation are integral to these dynamic approaches, improving model generalization and accuracy. These methods offer a more flexible and robust solution than static methods by incorporating descriptive features from diverse sources and dynamically updating models. Sarker (2021) demonstrates that this transition to dynamic detection approaches is crucial for effectively handling the complexities and variabilities of real-world data. Hence, this study addressed the challenges of crop disease detection by utilizing advanced image classification techniques, leveraging the globally recognized PlantVillage dataset and

locally sourced crop images, and the need for models to capture both low-range and long-range dependencies in agricultural data. Existing models struggle with generalization and overfitting, particularly when trained on small or imbalanced datasets, and often fail to balance accuracy with computational efficiency. The main contributions of this study are as follows:

We propose a hybrid architecture that integrates MobileNetV2 and EfficientNetV2 to effectively capture both low-range and long-range dependencies in the data.

We implement feature fusion via concatenation, combining extracted features from both CNN models to form a comprehensive representation, leveraging fine-grained details from both networks.

We incorporate dropout and batch normalization to enhance model generalization, prevent overfitting, and ensure stable feature learning during training.

2. Literature Review

2.1 Plant Disease Detection

Effective plant disease identification is vital for precision agriculture, impacting plant health and productivity. With increasing disease outbreaks, timely detection is crucial for diagnosis, control, and damage assessment. Early identification allows targeted treatments and prevents significant economic losses. Abdu et al. (2020) describe key metrics that include disease incidence (proportion of affected plants), severity (extent of damage), and consequence (impact on yield). Computer vision has enhanced plant disease detection beyond traditional human methods, which are labour-intensive and subjective (Kemi et al., 2022). Automated systems use machine learning to analyze plant images, recognizing patterns and features like texture, color, and shape. According to Zheng et al. (2019), classifiers detect diseases by comparing extracted features with categorized datasets. DL models like CNNs offer improved accuracy by processing large datasets and identifying subtle patterns. These advancements facilitate more effective disease segmentations on images, crucial for precise diagnosis and treatment. Color analysis is pivotal in plant disease detection, especially when identifying discoloration caused by various diseases. Color is often one of the earliest visible symptoms, and analyzing color variations can provide critical information about the plant's health. Sala et al. (2020) identify that the common color spaces like Hue-Saturation-Value (HSV), CIELAB (LAB), and RGB are frequently used to detect these changes, with HSV and LAB being particularly robust against variations in lighting conditions. RGB is an additive color model

where colors are created by combining different intensities of red, green, and blue light (Abbasi et al., 2023). Each component can range from 0 to 255, with (255, 255, 255) producing white and (0, 0, 0) resulting in black. LAB, which separates lightness from color-opponent dimensions, further refines this analysis, helping to differentiate healthy from diseased tissues under various conditions. Texture-based disease detection is highly effective for diagnosing plant diseases by quantifying surface characteristics and identifying subtle changes that may not be readily visible through color or shape alone.

2.2 Multimodal Fusion Models

Multimodal fusion, as demonstrated by Liu et al. (2024), involves integrating different types of data or features from multiple sources. This is crucial for improving the reliability and precision of disease detection. Advanced frameworks for detecting plant leaf diseases have achieved considerable success in early diagnosis. Researchers are developing and refining various fusion methods for automated disease classification, including feature fusion and deep learning. Feature fusion is a machine learning and computer vision technique that enhances classification tasks by combining features from different sources or models. Mi et al. (2020) introduced C-DenseNet, an enhancement of DenseNet architecture. C-DenseNet't incorporates the block attention module into DenseNet, which uses attention mechanisms to refine feature extraction by focusing on essential regions and channels. This fusion enhances DenseNet's ability to capture and emphasize critical features relevant to crop disease detection, as shown in the works of Sladojevic et al. (2016). C-DenseNet can potentially identify subtle disease symptoms more effectively by focusing on key features in plant images.

Dong et al. (2020) incorporated a coordinate attention mechanism into MobileNet to boost the Model performance while reducing its parameter count. MobileNetV2, known for its efficient depthwise separable convolutions, was further improved with coordinate attention, which allowed the Model to focus on spatially significant regions. This combination helped MobileNetV2 to better capture disease-related features in plant images. Mousavi and Farahani (2022) proposed an improved VGG16 grape disease detection model incorporating transfer learning. The VGG16 architecture, known for its deep convolutional layers, was optimized for mobile devices using transfer learning techniques. This fusion allowed the system to process images captured via mobile phones and provided real-time disease identification.

2.3 Features Extraction Process

Feature extraction using two parallel models involves independently processing input data through each model to derive distinct feature representations. These extracted features are then fused, often through concatenation or weighted averaging, to leverage the unique strengths of each architecture, ultimately enhancing overall model performance. An image is represented as a tensor with dimensions.(n_h, n_w, n_c), where n_w is the height, n_w and n_c represents the number of channels. The feature extraction process relies on convolution operations, where a filter (or kernel) of size is used. (f, f, n_c), slides across the image to capture patterns such as edges and textures. This process is mathematically expressed as:

$$\text{Conv}(I, K)_{(x,y)} = \sum_{i=1}^{f} \sum_{j=1}^{f} \sum_{k=1}^{n_c} K_{(i,j,k)} \times I_{(x+i-1, y+j-1, k)} \qquad (1)$$

To introduce non-linearity, a Rectified Linear Unit (ReLU) activation function is applied element-wise to the feature maps, ensuring that negative values are set to zero while retaining positive values. The dimensions of the activated feature maps remain, $\left(n_h', n_w', n_f\right)$. Pooling operations, such as max pooling, are then applied to reduce the spatial dimensions while preserving key features. Given a pooling window of size $p \times p$, the pooled feature map at position (x, y, c) is computed as:

$$P_{(x,y,c)} = \max_{i=0}^{p-1} p - \sum_{j=0} 1 A_{(x \cdot p+i, y \cdot p+j, c)} \qquad (2)$$

This process downsamples the feature maps, reducing computational complexity while retaining the most essential information. These steps are repeated across multiple convolutional and pooling layers, progressively refining and abstracting features. After several layers, the resulting pooled feature maps denoted as P^L, are flattened into a single feature vector V, with dimensions and are calculated as:

$$V \in \mathbb{R}^{\left(n_h^L \times n_w^L \times n_f^L\right)} \qquad (3)$$

The flattened feature vector is then passed through fully connected (FC) layers, where each layer applies linear transformations followed by activation functions. The transformation at the i-th layer is defined as:

$$FC_i = \sigma\left(\sum_{j=1}^{N} V_j \cdot W_{ij}^D + b_i^D\right) \qquad (4)$$

Where, W^D and b^D represent the weights and biases of the dense layer. In classification tasks, a SoftMax function is typically applied to the output of the final FC layer to convert activations into class probabilities, as shown:

$$\hat{y}_i = \frac{e^{FC_i}}{\sum_{j=1}^{K} e^{FC_j}}, \text{ for } i = 1,2,\dots,K \quad (5)$$

Where K is the number of output classes. The predicted class, $\hat{y}_{\text{pred}}$ is determined by selecting the index i corresponding to the highest probability, as shown:

$$\hat{y}_{\text{pred}} = \arg\max_i \hat{y}_i \quad (6)$$

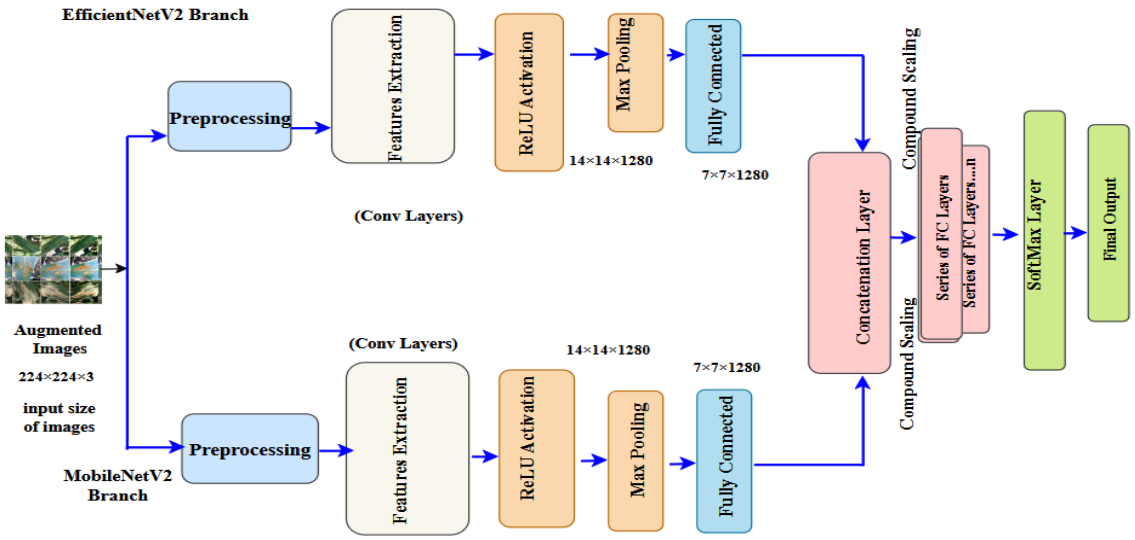
This approach enhances the model's ability to capture diverse feature representations by leveraging parallel feature extraction pathways, improving classification accuracy and efficiency.

3. Methodology

The proposed architecture, as shown in figure 1, comprises two parallel branches: EfficientNetV2 and MobileNetV2, each designed to optimize input data through tailored preprocessing steps. The ReLU (Rectified Linear Unit) activation function is employed in the convolutional layers to introduce non-linearity, effectively addressing the vanishing gradient problem. ReLU activates only positive values while zeroing out negative ones, ensuring the Model captures critical non-linear features and achieves faster convergence. The architecture incorporates compound scaling, uniformly scaling network depth, width, and resolution to balance accuracy and computational efficiency. This approach ensures consistent performance across all layers, maintaining a stable trade-off between feature extraction quality and computational demands. The outputs from both branches are fed into dedicated fully connected layers *(FC1)* with dimensions specified by fc_dim. These outputs are concatenated to form a combined feature set, allowing the Model to integrate unique feature representations from EfficientNetV2 and MobileNetV2. EfficientNetV2 captures complex patterns through its advanced scaling, while MobileNetV2 provides lightweight feature extraction, enhancing efficiency for edge environments. The concatenation preserves the full feature space from each branch, resulting in a comprehensive representation for downstream classification tasks. This concatenated output is processed through an additional set of fully connected layers *(FC2)* and a SoftMax layer for classification, producing the final output.

Figure 1

Proposed model architecture



After loading the dataset, pixel values underwent normalization to a range of [0,1]. This step was crucial for enhancing the Model convergence during training. The normalization of pixel values was mathematically expressed as follows:

$$\text{Normalized\_pixel} = \frac{\text{pixel\_value}}{255.0} \qquad (7)$$

Next, a stratified sampling technique was employed to ensure that the distribution of classes within the training and validation datasets remained consistent with the original dataset. We calculated the number of samples designated for the training set based on stratified sampling as follows:

$$\text{Train\_size} = \left(\frac{\text{number of samples in class}}{\text{total samples}}\right) \times \text{total samples} \times (1 - \text{test\_size}) \qquad (8)$$

Train_size represented the number of samples allocated to the training set, while total samples denote the total number of images within the dataset. The variable test_size specifies the proportion of the dataset reserved for validation, with a value of 0.2 indicating an 80–20 split between training and validation sets. To ensure the reproducibility of the data split across multiple runs, a random state of 42 was specified. The following algorithm in figure 2 iterated through the dataset, extracting images and labels.

Figure 2

Dataset preparation and stratified split algorithm

Algorithm 1 Dataset Preparation and Stratified Splitting

Input: Directory path containing images categorized into different disease classes.
Output: Training and validation datasets for model training.
Initialize an empty dictionary class_dict ← defaultdict(list)
Load dataset from directory and derive class_names
for each batch of images, labels in all_dataset **do**
  for each image, label in (images, labels) **do**
    class_idx ← argmax(*label*)
    class_dict[class_names[class_idx]] ← append(*img*)
  end for
end for
Initialize all_images ← []
Initialize all_labels ← []
for each class_name, images in class_dict.items() **do**
  all_images.extend(images)
  all_labels.extend([class_names.index(class_name)] for _ in range(len(images)))
end for
all_images ← np.array(all_images)
all_labels ← np.array(all_labels)
Perform stratified split:
(train_images, val_images, train_labels, val_labels) ← train_test_split
  with test_size = 0.2, stratify = all_labels, random_state = 42

Figure 3

Model training and evaluation algorithm

Algorithm 2 Model Training

Data Preparation
Split dataset into training and validation sets: $D_{\text{train}}, D_{\text{val}} \leftarrow \text{Split}(D)$
Create TensorFlow datasets, preprocess images, and one-hot encode labels:
$\mathcal{D} \leftarrow$ tf.data.Dataset.from_tensor_slices(I, Y),
$I' \leftarrow \text{Resize}(I, 224 \times 224)$, $I'' \leftarrow \frac{I'}{255}$, $Y_{\text{one-hot}}(y) \leftarrow \begin{cases} 1 & \text{if } y \text{ is the class} \\ 0 & \text{otherwise} \end{cases}$
Create batches and prefetch data: $\mathcal{D} \leftarrow \mathcal{D}.batch(16).prefetch(1)$
Model Architecture
Define input tensors and extract features:
$I_1, I_2 \in R^{224\times224\times3}$,
$F_1 \leftarrow \text{MobileNetV2}(I_1), F_2 \leftarrow \text{EfficientNetV2B0}(I_2)$
Apply Global Average Pooling (GAP) and concatenate:
$F_{\text{GAP}}(F) \leftarrow \frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W} F(i,j), x \leftarrow \text{concat}(F_1, F_2)$
Dense Layer Transformation
Apply Dense layer, Batch Normalization, and Dropout:
$x \leftarrow \text{ReLU}(Wx + b), z_{\text{norm}} \leftarrow \frac{z-\mu}{\sqrt{\sigma^2+\epsilon}}, x \leftarrow x \cdot \text{mask}$
Final Output Layer
Define output layer with softmax activation: $p_i \leftarrow \text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}$
Model Compilation
Compile model with Adam optimizer: $w_{t+1} \leftarrow w_t - \alpha\frac{\hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}$
Loss Function
Calculate Categorical Cross-Entropy with label smoothing:
$y_{\text{smooth}} \leftarrow y \times (1-\alpha) + \frac{\alpha}{C}, L \leftarrow -\sum_{i=1}^{C} y_{\text{smooth},i}\log(p_i)$
Training and Evaluation
Monitor performance with callbacks (Model Checkpoint, Early Stopping, ReduceLROnPlateau) and track metrics:
$A_{\text{train}}, A_{\text{val}}, L_{\text{train}}, L_{\text{val}}$
Final Model Evaluation
Predictions on validation dataset: $y_{\text{pred\_class}} \leftarrow \text{argmax}(y_{\text{pred}}, \text{axis} = 1)$
Calculate evaluation metrics: Confusion Matrix, Precision, Recall, F1-Score

For the dual-input model architecture, as shown in figure 3, the training dataset ensured that both models received the same preprocessed image as input. The dataset originated from a structured directory, each subdirectory representing a specific class label. Using categorical labeling, each image was assigned a one-hot encoded vector, providing a unique identifier for its class. Additionally, the large variability in crop disease images required the model to adapt effectively without overshooting optimal parameter values during training. Initially, we set the learning rate to 1×10^{-3}, commonly used as a starting point for many models. However, this learning rate was too high for our model, leading to unstable training and causing the loss to fluctuate significantly, which hindered proper convergence. The larger learning rate resulted in "overshooting," where the model failed to settle into the local minima of the loss function, making it difficult to achieve optimal training results. To address these issues, we opted for a lower learning rate of 1×10^{-5}. Furthermore, the reduced learning rate provided better stability during training, preventing the Model from oscillating around the optimal values.

In the one-hot encoding, each class label was represented as a binary vector where only the entry corresponding to the target class was set to 1, and all others were set to 0. Mathematically, for a label y belonging to one of the C classes, the one-hot encoding was calculated as:

$$\text{one}_{\text{hot}\,(y)} = \begin{cases} 1 & \text{if class } = y \\ 0 & \text{otherwise} \end{cases} \qquad (9)$$

If the dataset contained N samples, the label matrix after one-hot encoding, Y, will have the shape $N\times C$. The dataset creation process involved pairing each image with its corresponding label using from_tensor_slices, allowing seamless mapping between the input data and labels. Each image-label pair was then processed through a series of transformations, including resizing to the specified input dimensions, normalizing pixel values, and ensuring compatibility for dual input model architectures. To leverage both MobileNetV2 and EfficientNetV2, two input layers were created for dual input processing. Each input layer accepted images with dimensions 224×224×3. Mathematically, if an input image is represented as a 3D tensor I, its dimensions are defined as follows:

$$I = [224,224,3] \qquad (10)$$

The first two dimensions (224, 224) represent the height and width of the image, and the third dimension (3) corresponds to the RGB color channels. This setup allowed each model to process the input image independently before combining their outputs for further fusion and analysis. Both models, MobileNetV2 and EfficientNetV2, were pre-trained on the ImageNet dataset and removed their top (classification) layer. The feature maps produced by each model were denoted as F_1 and F_2 respectively as shown:

$$F_1 = \text{MobileNetV2}(I_1), F_2 = \text{EfficientNetV2B0}(I_2) \quad (11)$$

Global average pooling (GAP) was then applied to the feature maps F_1 and F_2 . GAP reduced each feature map to a single value by averaging each channel's spatial dimensions (height and width). For instance, if F_1have dimensions $h \times w \times d$, GAP produced an output of dimension *d*, which was calculated as follows:

$$\text{GAP}(F_1) = \frac{1}{h \times w} \sum_{i=1}^{h} \sum_{j=1}^{w} F_1(i, j, k) \quad (12)$$

Where *k* is the index of each channel. This pooling operation compressed spatial information, retaining only the most essential features from each channel, facilitating a more compact representation of the input data while preserving key characteristics. The resulting pooled output features of both models, x_1 and x_2, were then concatenated to create a combined feature vector *x* which was defined as:

$$x = \text{concat}(x_1, x_2) \quad (13)$$

Initially, activation functions like sigmoid and tanh were experimented with during model training. However, these posed significant challenges. Both functions suffered from the vanishing gradient problem, where gradients became very small during backpropagation. This slowed down the training process and hindered convergence, particularly in the deeper layers of the network. We ultimately opted for the ReLU (Rectified Linear Unit) activation function to address these challenges. This combined vector was then passed through a dense layer with ReLU activation, which performed an affine transformation followed by a non-linear activation and was calculated as:

$$x = \text{ReLU}(Wx + b) \quad (14)$$

Where *W* and *b* are the weights and biases of the dense layer. By normalizing the output of each layer using the mean and variance calculated across the mini-batch, batch normalization helps stabilize the learning process, accelerates convergence, and mitigates issues related to internal covariate shifts. Given an input z, we computed batch normalization as shown:

$$z_{\text{norm}} = \frac{z-\mu}{\sqrt{\sigma^2+\epsilon}} \quad (15)$$

Where, μ and σ^2 are the batch mean and variance, and ϵ is a small constant. Dropout was then randomly set as a fraction *p* of input units to zero, helping to prevent overfitting, and was defined as:

$$\text{Dropout}(x) = x \cdot \text{ mask} \quad (16)$$

The mask is a binary vector with elements set to 0 with probability p=0.3. We used a dense layer with a softmax activation in the final output layer. By selecting the class with the highest probability, we could make more confident and accurate predictions, with the softmax activation ensuring that the sum of all predicted probabilities equalled 1, providing a clear and interpretable decision and was calculated as follows:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \quad (17)$$

Z_i is the i-th logit (raw model output), and *C* is the number of classes. The model was then compiled using the Adam optimizer, which adjusted weights based on the gradient's first and second moments. This allowed the optimizer to adapt the learning rate dynamically during training, ensuring more efficient convergence. The update rule for each weight *w* at time step *t* was calculated as follows:

$$w_{t+1} = w_t - \alpha \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t+\epsilon}} \quad (18)$$

where α is the learning rate, $\widehat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second-moment estimates of the gradient, and ϵ is a small constant to prevent division by zero. The loss function used was categorical cross-entropy with label smoothing. For each class, the smoothed label y_{smooth} was calculated as follows:

$$y_{\text{smooth}} = y \times (1 - \alpha) + \frac{\alpha}{C} \quad (19)$$

where y was the original label, α is the smoothing factor, and N is the number of classes. The loss L was then calculated by incorporating label smoothing to adjust the error calculation during training, leading to more stable and effective learning, and was calculated as follows:

$$L = -\sum_{i=1}^{C} y_{\text{smooth},i} \log(p_i) \quad (20)$$

where p_i is the predicted probability for class i. This approach reduced overconfidence in predictions, making training more stable. Callback mechanisms—specifically model checkpoint, earlyStopping, and ReduceLROnPlateau—were configured to reduce overconfidence. These model checkpoints monitored the validation loss and saved the model whenever an epoch achieved a new minimum validation loss, and was defined as:

$$\text{Save Model: } \begin{cases} \text{Save if val\_loss}_{\text{new}} < \text{val\_loss}_{\text{best}} \\ \text{Update val\_loss}_{\text{best}} = \text{val\_loss}_{\text{new}} \end{cases} \quad (21)$$

EarlyStopping monitored the validation loss and halted training when no improvement was observed for a specified number of epochs, helping to prevent overfitting, which was calculated as.

$$\text{Stop Training: } \begin{cases} \text{if val\_loss} > \text{val\_loss}_{t-1} \text{ for } p \text{ epochs} \\ \text{Restore Best Weights} \end{cases} \quad (22)$$

Reduce Learning Rate on Plateau was used when the validation loss remained unchanged for three consecutive epochs. The learning rate was halved. This adjustment encouraged the Model to converge more precisely by allowing for finer weight updates, facilitating better optimization in the later stages of training, and was calculated as follows:

$$\text{Update Learning Rate: } \begin{cases} \text{if val\_loss}_t \geq \text{val\_loss}_{t-1} \\ \text{then } \eta_t = \eta_{t-1} \cdot f, \text{ where } f < 1 \end{cases} \quad (23)$$

The model was trained over 24 epochs. Where, L_{train} and L_{val} represented training and validation losses, respectively, and A_{train} and A_{Val} represented training and validation accuracy. Performance was evaluated per epoch to track improvements and avoid overfitting by observing the trends and was defined as:

L_{train} , L_{Val} , $A_{\text{train,}}$ and A_{Val} (24)

The final model was evaluated on the validation dataset using accuracy, precision, recall, and F1-score metrics. To align the true labels, we represented them as:

$$Y_{\text{true}} = \sum_{i=1}^{N} Y_1^i \quad (25)$$

N was the number of batches in the validation dataset, and Y_1 represented the one-hot encoded labels. Predictions were generated on the validation dataset to further analyze the model's accuracy. The model output probabilities for each class were converted to class predictions using the argmax function. This yielded the predicted class. $y_{\text{pred\_class}}$ for each sample and was computed as follows:

$$y_{\text{pred\_class}} = \text{argmax}\left(y_{\text{pred}}, \text{axis} = 1\right) \quad (26)$$

True labels, y_{true} were similarly obtained by converting one-hot encoded labels from the validation dataset to class indices and were calculated as shown:

$$y_{\text{true}} = \text{argmax}(y_{\text{labels}}, \text{axis} = 1) \quad (27)$$

This process allowed for a direct comparison between the predicted and true class labels, facilitating the evaluation of the Model performance. With both $y_{\text{pred\_class}}$ and y_{true} aligned, it became possible to calculate additional metrics.

4. Results and Discussions

4.1 Dataset Description

The study combined the Kaggle dataset (Saleem et al., 2020) and the FieldPlant datasets to create a comprehensive resource named the DEMF dataset. The Kaggle dataset, with 38 distinct classes and 60,343 images, provided a globally diverse and well-structured resource ideal for training and validating the models. The FieldPlant dataset, developed in this study, contributed 25,775 annotated images of plant leaves, primarily collected from farms in central Kenya. Data augmentation techniques were applied to classes with fewer images to address imbalances and strengthen underrepresented classes. Augmented images were physically generated and stored in their respective directories, increasing the dataset's diversity and

ensuring a more balanced representation across all classes. The final dataset, as shown in table 1, comprised 22 crop types, 76 individual classes, and 99,551 images, divided into 79,601 training images and 19,950 validation images.

Table 1

Dataset distribution

Crop Type	Total Images	Training Images	Validation Images
Apple	4,651	3,719	932
Banana	4,008	3,204	804
Beans	8,096	6,475	1,621
Blueberry	1,502	1,201	301
Cassava	4,894	3,914	980
Cherry	2,054	1,642	412
Corn	4,358	3,484	874
Grape	4,641	3,711	930
Maize	1,002	801	201
Maize-leaf	1,239	991	248
Maize	4,985	3,986	999
Orange	5,507	4,405	1,102
Peach	3,299	2,638	661
Pepper	2,480	1,983	497
Potatoes	3,006	2,403	603
Raspberry	1,002	801	201
Rice	5,010	4,005	1,005
Squash	1,835	1,468	367
Strawberry	2,111	1,688	423
Sugarcane	5,010	4,005	1,005
Sunflower	4,008	3,204	804
Tea	6,012	4,806	1,206
Tomatoes	18,841	15,067	3,774
Total	99,551	79,601	19,950

4.2 Experimental Parameters and Environment

Table 2 outlines the experimental setup, where data was stored and accessed via Google Drive. The dataset was organized for efficient preprocessing and stratified splitting into training and validation sets. The images were resized to a fixed dimension of 224×224,

normalized, and prepared for dual inputs required by the MobileNetV2 and EfficientNetV2 branches. Data augmentation and batching were performed to enhance model robustness, with a batch size 16 ensuring manageable computational loads. Key parameters, such as categorical labels and class names, were extracted and processed to ensure compatibility with the classification pipeline. The Model architecture combined outputs from MobileNetV2 and EfficientNetV2 through concatenation, followed by dense layers with ReLU activation, batch normalization, and dropout for regularization. The model was compiled with the Adam optimizer and a learning rate of $1\times10-^{5}$, using categorical cross-entropy loss and label smoothing to improve classification performance. Multiple callbacks were integrated for checkpointing, early stopping, and adaptive learning rate reduction, optimizing the training process. The pipeline concluded with evaluation metrics such as validation accuracy, classification reports, confusion matrices, and ROC-AUC curves to assess model performance comprehensively. The experiments were conducted on an NVIDIA RTX 3090 GPU, providing reliable and efficient performance. The setup included a virtualized Intel Xeon CPU with access to virtualized GPUs (NVIDIA T4, Tesla P100, K80) and an operating system based on Linux (Ubuntu). Python was the primary language, supported by frameworks such as TensorFlow, PyTorch, Keras, and OpenCV.

Table 2

Hyperparameter Configurations

Hyperparameter	**Value**
Image size	224 × 224
Image channels	3
Batch size	16
Number of MobileNetV2 layers	Feature extraction only
Number of EfficientNetV2B0 layers	Feature extraction only
Hidden dimension	256
Dropout rate	0.3
Number of epochs	17
Learning rate	1e-5
Optimizer	Adam
Loss function	Categorical Cross entropy (label smoothing = 0.1)
Callbacks	EarlyStopping, ModelCheckpoint, ReduceLROnPlateau

4.3 Evaluation Approach

A comprehensive set of metrics was chosen to evaluate the proposed model and assess its predictive accuracy. Accuracy measured the correctly classified instances (positive and negative) from the dataset. This metric reflected the model overall effectiveness across all disease categories and was computed as follows:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+TN+FN} \quad (28)$$

TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively. Precision is essential for reducing false positives, a critical factor in real-time crop disease detection, and was as calculated as:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (29)$$

Recall, also known as sensitivity, measures the proportion of positives the model successfully identifies. This metric is crucial in disease detection as it minimizes false negatives, ensuring that diseased plants are not overlooked. and was calculated as follows:

$$\text{Recall} = \frac{TP}{TP+FN} \quad (30)$$

F1-score combines precision and recall into a single metric, using the harmonic mean to balance both. The F1 score is especially valuable when false positives and negatives carry significant implications and was calculated as follows:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (31)$$

The ROC-AUC (Receiver Operating Characteristic - Area Under the Curve) measures the Model effectiveness in distinguishing between classes by examining the relationship between true positive and false positive rates. AUC values closer to 1.0 indicate stronger discriminatory power, critical in distinguishing subtle differences between healthy and diseased samples. The AUC is typically calculated from the area under the ROC curve.

4.4 Classification Results of the Proposed Model

The Model training and validation results, as shown in table 3, demonstrate a consistent improvement in performance across 17 epochs. Training accuracy increased significantly from 61.31% in Epoch 1 to 99.52% in Epoch 17, while validation accuracy improved from 89.56% to 98.63%. Similarly, training loss decreased from 2.1238 to 0.8382, and validation loss dropped from 1.1773 to 0.8368, indicating the model ability to minimize errors effectively. The consistent learning rate of 0.00001 throughout the training process contributed to these steady improvements. With a total training time of 23h 37m 24s, the results highlighted the model's robustness and generalization capability, ensuring no signs of overfitting.

Table 3

Training and validation performance metrics analysis

Epoch	Loss	Accuracy	Validation Loss	Validation Accuracy	Learning Rate	Time (s/step)
1	2.1238	0.6131	1.1773	0.8956	0.00001	5066
2	1.2297	0.8815	1.0571	0.9489	0.00001	5009
3	1.0932	0.9302	0.9983	0.9637	0.00001	4997
4	1.0306	0.9506	0.9615	0.9695	0.00001	4996
5	0.9886	0.9630	0.9462	0.9754	0.00001	4997
6	0.9581	0.9718	0.9218	0.9794	0.00001	4995
7	0.9347	0.9785	0.9061	0.9806	0.00001	5004
8	0.9171	0.9823	0.8912	0.9827	0.00001	5007
9	0.9030	0.9851	0.8836	0.9831	0.00001	4989
10	0.8899	0.9876	0.8763	0.9837	0.00001	4998
11	0.8783	0.9903	0.8697	0.9837	0.00001	4994
12	0.8711	0.9915	0.8648	0.9845	0.00001	4995
13	0.8626	0.9927	0.8556	0.9856	0.00001	5004
14	0.8559	0.9933	0.8503	0.9852	0.00001	4999
15	0.8496	0.9942	0.8472	0.9855	0.00001	4999
16	0.8443	0.9943	0.8453	0.9855	0.00001	4995
17	0.8382	0.9952	0.8368	0.9863	0.00001	5000

Figure 4 illustrates the progressive improvement in training and validation accuracy across 17 epochs, showcasing the model's enhanced performance and generalization. On the other hand, figure 5 depicts the steady decline in training and validation loss, highlighting the model's effective error minimization over time.

Figure 4

Training and validation accuracy

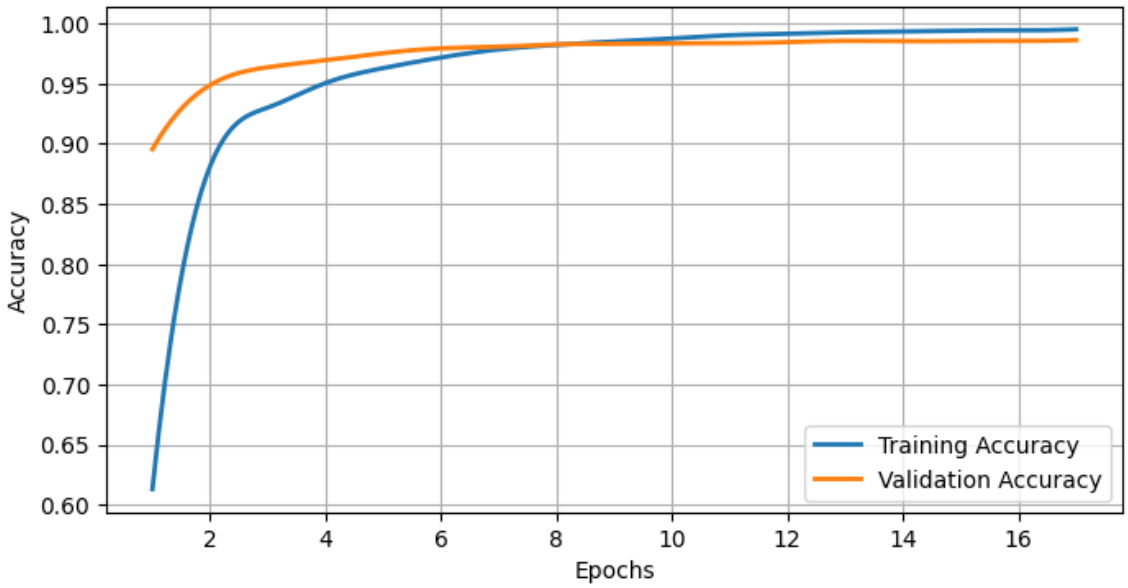
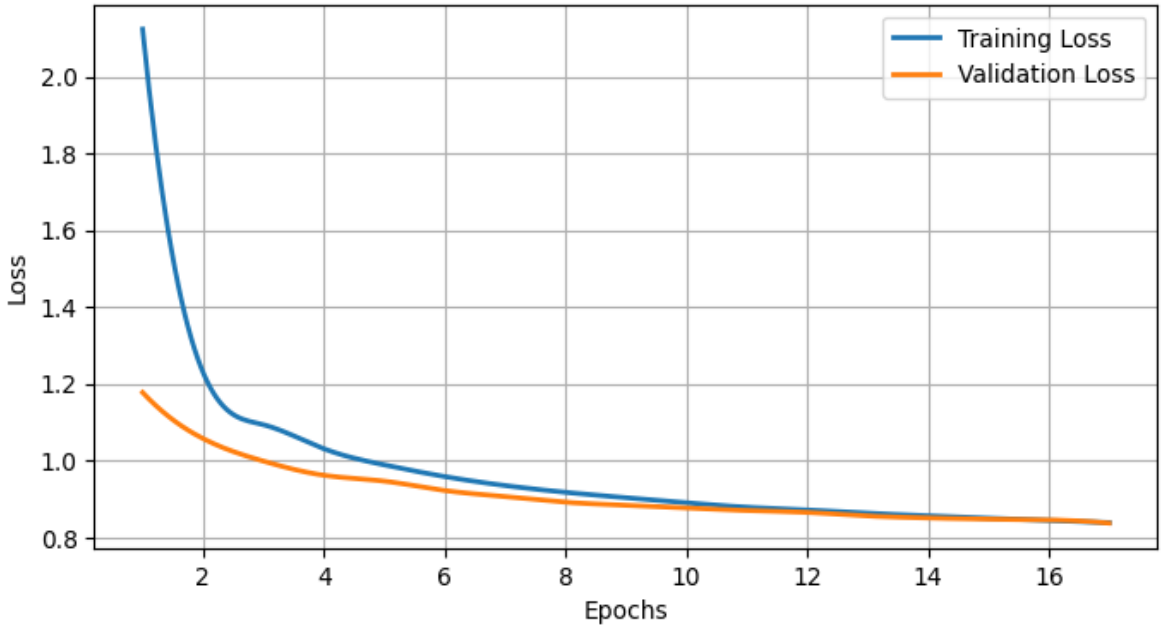


Figure 5

Training and validation loss



Tables 4 and 5 showcase the classification outcomes for all 76 crop disease categories, thoroughly assessing the Model performance across a broad spectrum of crop diseases.

Table 4

Classification performance for crop disease classes (0 To 40)

Class Name	Precision	Recall	F1-Score	Support
Apple___Apple_scab	1.00	0.99	1.00	200
Apple___Black_rot	1.00	1.00	1.00	200
Apple___Cedar_apple_rust	1.00	1.00	1.00	200
Apple___healthy	1.00	1.00	1.00	329
Banana_cordana	1.00	1.00	1.00	200
Banana_healthy	0.99	1.00	1.00	200
Banana_pestalotiopsis	0.99	0.99	0.99	200
Banana_sigatoka	1.00	0.99	1.00	200
Bean_angular_leaf_spot	1.00	0.98	0.98	201
Beans_healthy	1.00	0.99	1.00	200
Blueberry___healthy	1.00	1.00	1.00	301
Cassava_brown_spot	1.00	1.00	1.00	296
Cassava_green_mite	0.97	0.95	0.96	203
Cassava_healthy	0.98	0.99	0.98	239
Cassava_mosaic	0.97	0.98	0.97	241
Cherry___Powdery_mildew	1.00	1.00	1.00	211
Cherry___healthy	1.00	0.99	1.00	200
Corn___Cercospora_leaf_spot Gray_leaf_spot	0.97	0.96	0.96	201
Corn___Common_rust	1.00	1.00	1.00	239
Corn___Northern_Leaf_Blight	0.95	0.97	0.96	200
Corn___healthy	1.00	0.99	0.99	233
Grape___Black_rot	0.99	1.00	1.00	236
Grape___Esca_(Black_Measles)	1.00	0.99	1.00	277
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1.00	1.00	1.00	215
Grape___healthy	1.00	1.00	1.00	200
Maize _grasshoper	0.99	0.98	0.99	200
Maize-leaf_spot	0.75	0.67	0.71	248
Maize_fall_Armyworm	0.95	1.00	0.97	201
Maize_healthy	0.91	0.99	0.95	199
Maize_leaf_beetle	0.97	0.97	0.97	199
Maize_leaf_blight	0.72	0.71	0.72	200
Maize_streak_virus	0.91	0.89	0.90	199
Orange___Haunglongbing_(Citrus_greening)	1.00	1.00	1.00	1102
Peach___Bacterial_spot	1.00	1.00	1.00	460
Peach___healthy	1.00	1.00	1.00	200
Pepper, _bell___Bacterial_spot	1.00	0.99	1.00	200
Pepper,_bell___healthy	1.00	1.00	1.00	296
Potato___Early_blight	1.00	1.00	1.00	200
Potato___Late_blight	1.00	0.99	0.99	201

Table 5

Classification performance for crop disease classes (41 to 76)

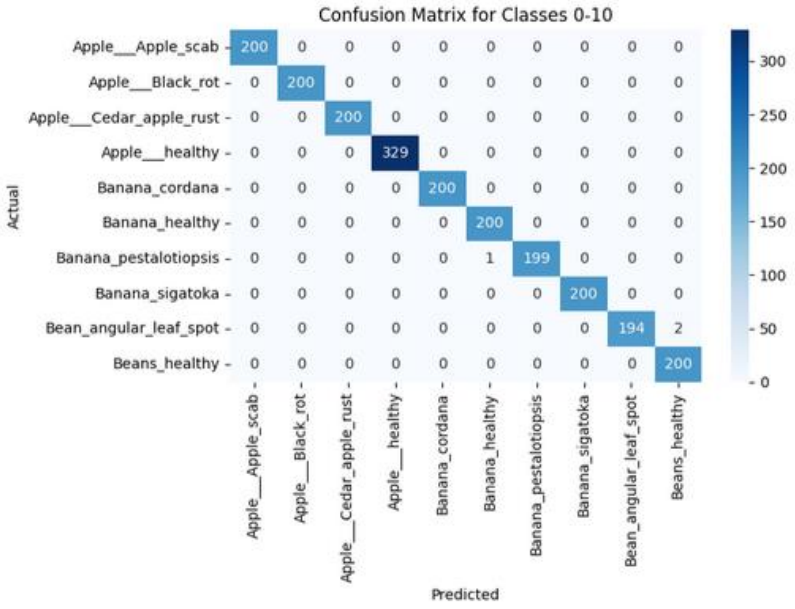
Class Name	Precision	Recall	F1-Score	Support
Potato___healthy	1.00	1.00	1.00	200
Raspberry___healthy	1.00	1.00	1.00	201
Rice_bacterial_leaf_blight	1.00	1.00	1.00	200
Rice_brown_spot	0.99	0.98	0.98	201
Rice_healthy	1.00	1.00	1.00	201
Rice_leaf_blast	0.98	0.99	0.99	200
Rice_narrow_brown_spot	1.00	1.00	1.00	200
Soybean___healthy	1.00	1.00	1.00	1018
Squash___Powdery_mildew	1.00	1.00	1.00	367
Strawberry___Leaf_scorch	1.00	1.00	1.00	222
Strawberry___healthy	1.00	1.00	1.00	200
Sugarcane_Healthy	0.97	0.99	0.98	200
Sugarcane_Mosaic	0.97	0.98	0.97	201
Sugarcane_RedRot	0.99	1.00	0.99	200
Sugarcane_Rust	1.00	0.96	0.98	200
Sugarcane_Yellow	0.99	0.99	0.99	200
Sunflower_Downy mildew	0.99	0.99	0.99	200
Sunflower_Fresh Leaf	1.00	1.00	1.00	200
Sunflower_Gray mold	1.00	1.00	1.00	201
Sunflower_Leaf scars	0.99	0.99	0.99	200
Tea_Anthracnose	1.00	1.00	1.00	201
Tea_algal leaf	1.00	1.00	1.00	200
Tea_bird eye spot	1.00	1.00	1.00	201
Tea_brown blight	1.00	1.00	1.00	200
Tea_healthy	1.00	1.00	1.00	200
Tea_red leaf spot	1.00	1.00	1.00	200
Tomato___Bacterial_spot	0.99	1.00	1.00	426
Tomato___Early_blight	0.99	0.96	0.97	201
Tomato___Late_blight	0.99	0.99	0.99	382
Tomato___Leaf_Mold	1.00	1.00	1.00	200
Tomato___Septoria_leaf_spot	1.00	0.99	1.00	354
Tomato___Spider_mites Two-spotted_spider_mite	0.98	1.00	0.99	335
Tomato___Target_Spot	0.99	0.99	0.99	281
Tomato___Tomato_Yellow_Leaf_Curl_Virus	1.00	1.00	1.00	1072
Tomato___Tomato_mosaic_virus	1.00	1.00	1.00	200
Tomato___healthy	1.00	1.00	1.00	318
bean_rust	0.97	1.00	0.98	201

The confusion matrices shown in figure 6 (a to f) illustrate the performance of the disease detection task across all classes (ranging from class 0 to class 76). These matrices display actual class labels on the X-axis and predicted labels on the Y-axis, providing insights into the Model classification accuracy for each class. Figure 12 illustrates the ROC curve,

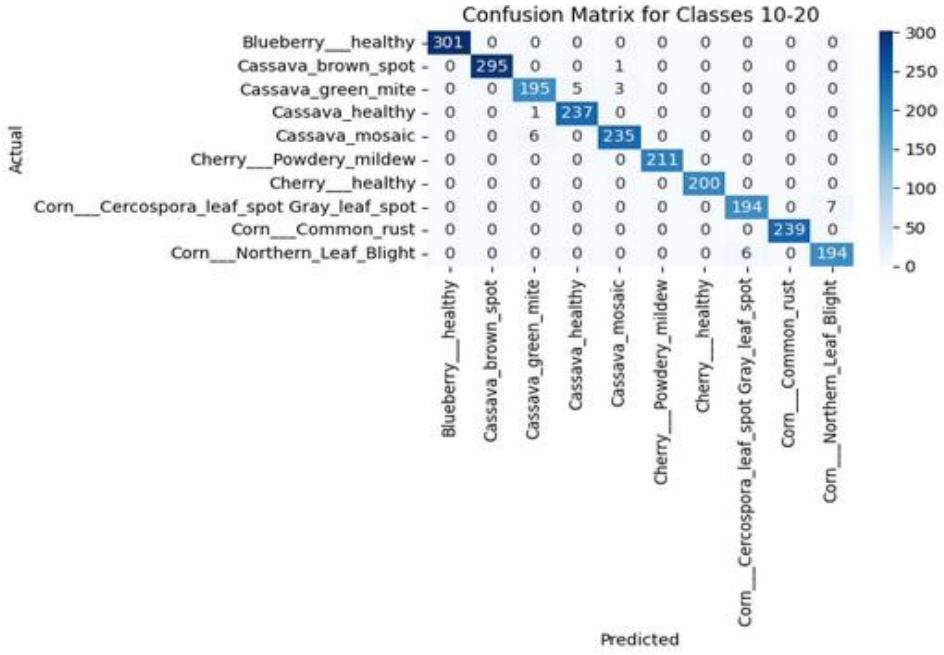
which showcases the model performance across different classification thresholds. The AUC score in the figure reflects the Model's ability to effectively differentiate between positive and negative classes, with higher AUC values indicating stronger predictive power and better overall performance.

Figure 6

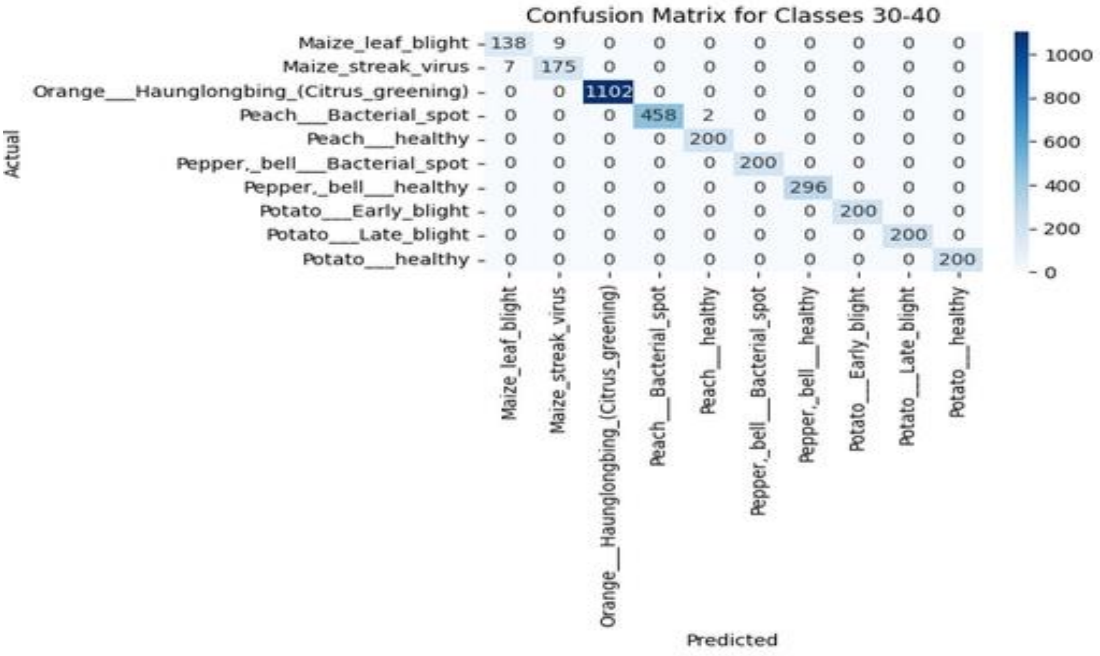
Confusion matric for classes 0-10



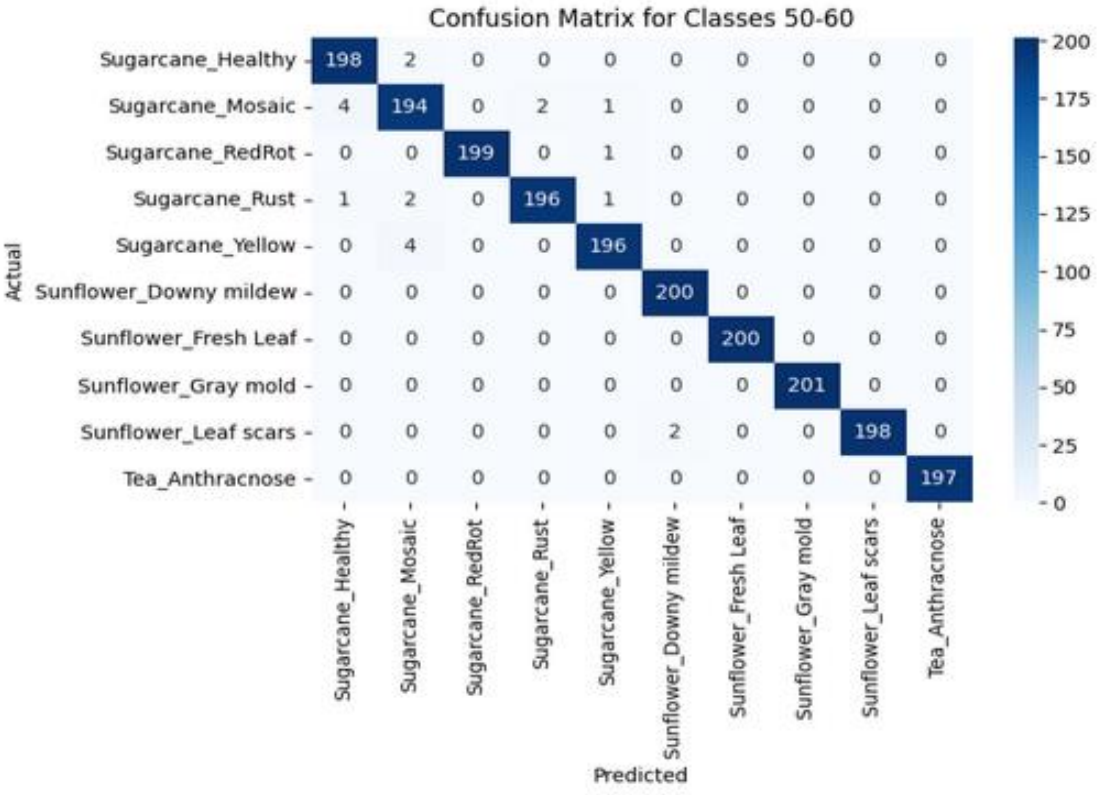
Confusion matric for classes 10-20



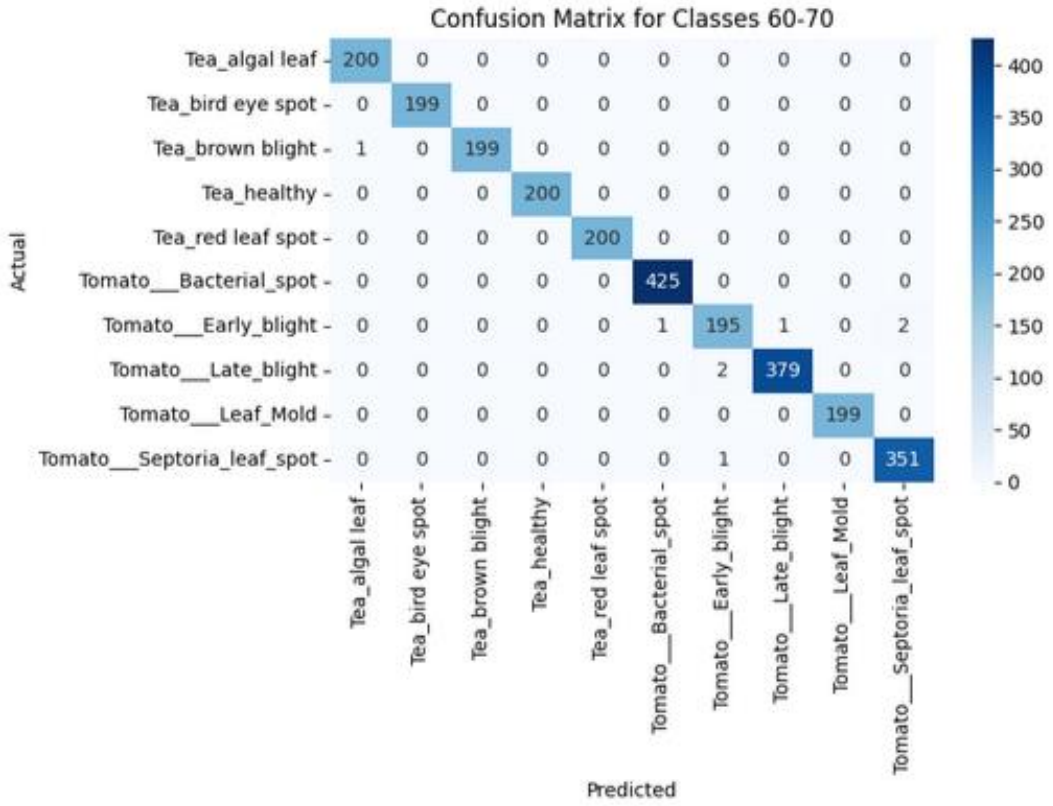
Confusion matric for classes 30-40



Confusion matric for classes 50-40



Confusion matric for classes 60-70



Confusion matric for classes 60-76

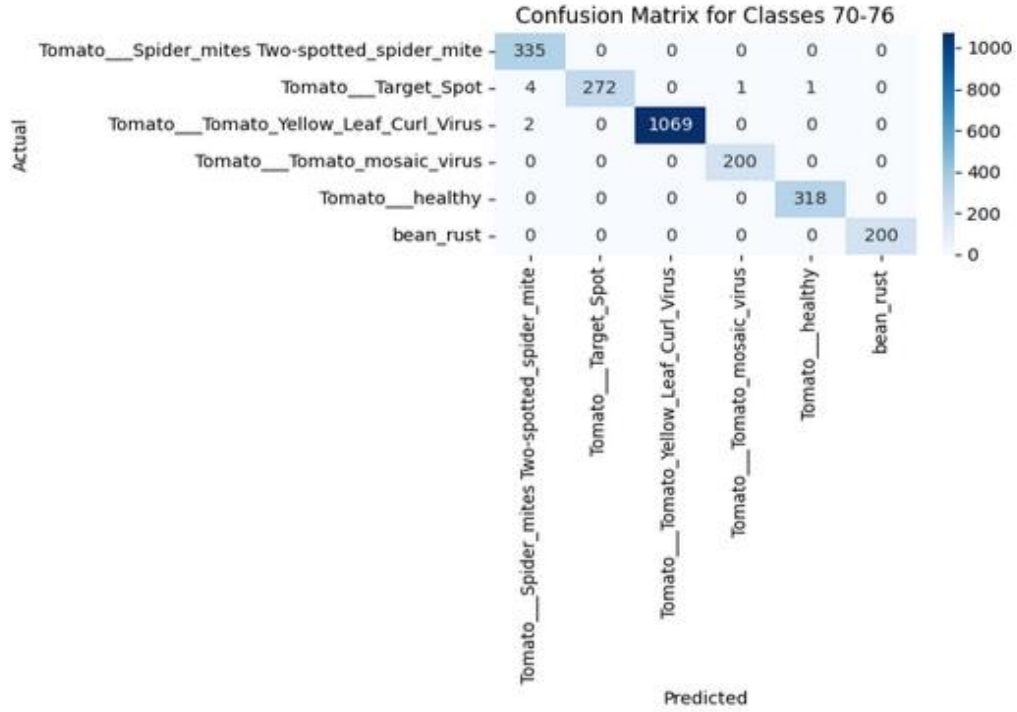
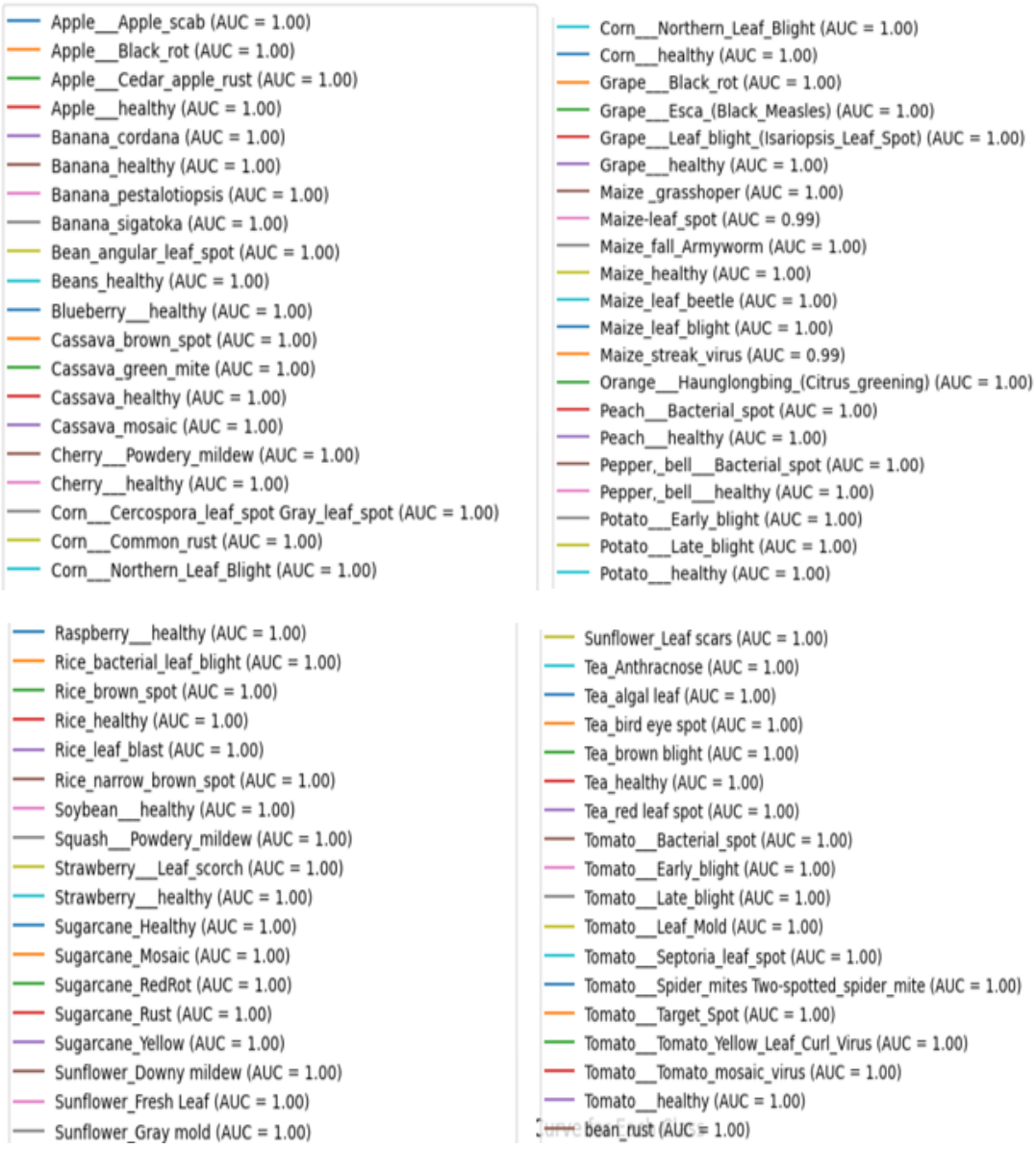


Figure 7

ROC- AUC Scores



4.5. Ablation Studies

Table 6 displays the analysis of how data augmentation techniques influence the model performance. Each transformation aimed to introduce variability, simulating real-world conditions. The rotation randomly altered the leaf's orientation, while flipping helped the model generalize across orientations. Brightness adjustment simulated lighting conditions, and zoom introduced scale and focus variation. These augmentations enhanced the model generalizability and performance in diverse environmental conditions.

Table 6

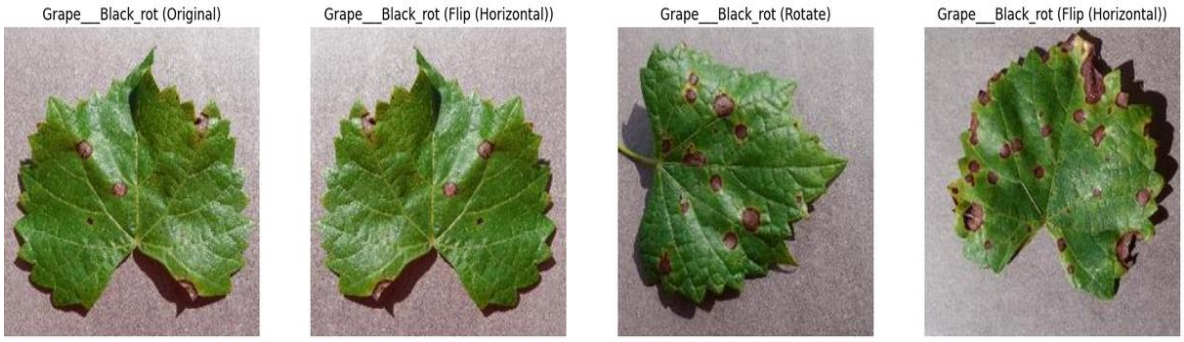
Augmentation techniques

Transformation Type	Range/Details
Rotation	0, 90, 180, or 270 degrees
Flipping	Horizontal flip and vertical flip
Brightness Adjustment	Between 0.7 (dark) and 1.3 (bright)
Zoom	Cropped a random portion and resized to 224x224 pixels

The resulting sample augmented images in figure 8 demonstrate the transformations applied during preprocessing. The augmentation techniques enhanced model training by generating a diverse dataset while preserving the distinguishing features of crop diseases.

Figure 8

Sample augmented images



On the unseen data, 249 images were processed, with 239 correctly classified, resulting in an accuracy of 95.98%. Only 10 images (4.02%) were misclassified, further supporting the model overall strong performance. The model demonstrated its ability to accurately classify plant diseases, even when the confidence scores were low for a few classes lacking dominant features. This suggested that the proposed model, as shown in figure 9, generally distinguished between healthy and diseased plants.

The proposed model capacity to deliver high-confidence predictions for diseases with strong visual cues, alongside moderate performance for others, highlights its practical utility for real-world deployment. The model proved its ability to classify plant diseases effectively in real-world scenarios, as demonstrated by the sample results in figure 10. When applied to field data, these samples illustrate the model performance, showcasing its ability to provide accurate predictions across various plant diseases with confidence scores.

Figure 9

Classification summary on unseen images

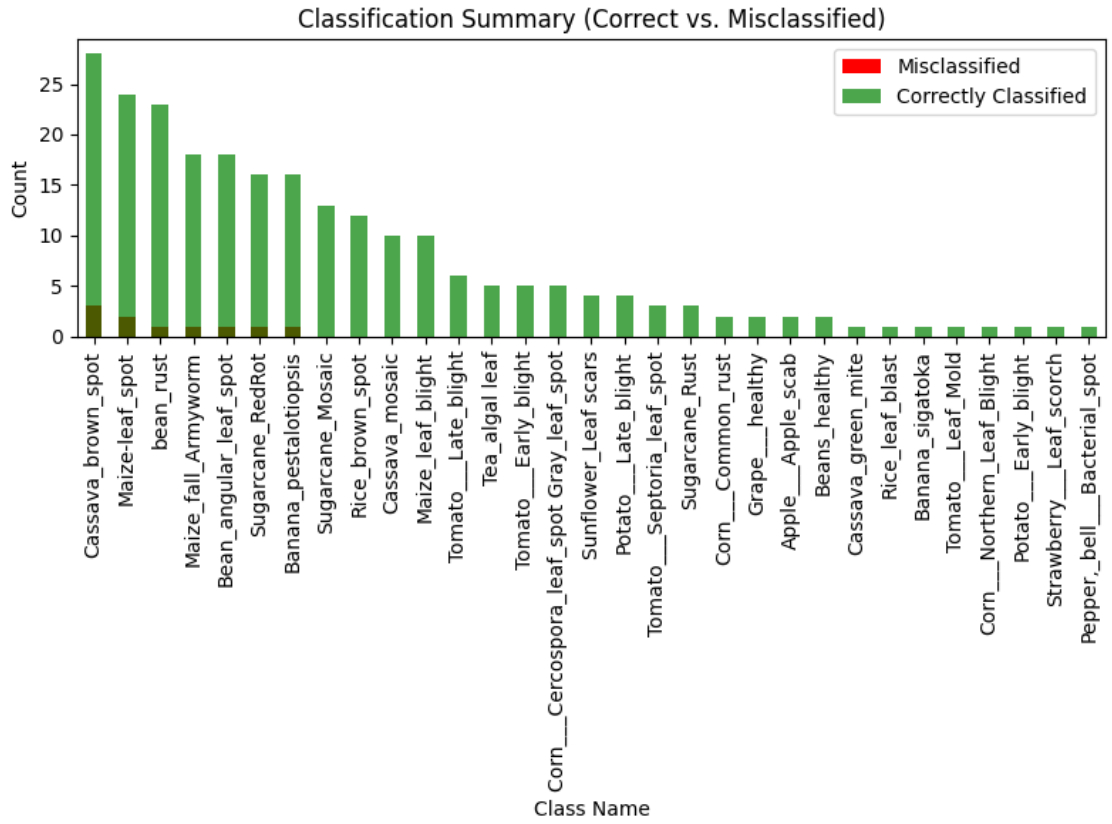


Figure 10

Random classes actual vs. predicted classification

4.6 Comparison with Other Models

The proposed hybrid model consistently outperformed existing crop disease detection models, achieving high accuracy rates that surpass traditional CNN-based and ViT-based approaches. As summarized in table 7, several models were tested, including MobileNetV2, EfficientNetB0, EfficientNetV2, DenseNet121, DenseNet50, ResNet152, AlexNet, and Custom CNN, all on the same dataset for comparison.

Table 7

Comparative performance of all models

Model	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
Proposed Model	99.61%	98.63%	0.0331	0.8458
MobileNetV2	98.92%	98.21%	0.0507	0.8663
EfficientNetB0	97.65%	91.02%	0.0811	1.2424
EfficientNetV2	99.08%	97.95%	0.0702	1.0291
DenseNet121	98.80%	97.75%	0.0675	1.0733
DenseNet50	98.75%	96.11%	0.0706	1.0975
ResNet152	98.74%	96.45%	0.0852	1.2092
AlexNet	97.88%	93.50%	0.1189	1.5391
Custom CNN	92.10%	61.84%	0.2750	2.5675

As shown in table 8, the final trained model sizes varied significantly, with AlexNet being the largest at 551,564 KB due to its complex architecture. ResNet followed with 100,500 KB, while the DEMF Model was slightly smaller at 104,625 KB, optimized for edge computing tasks. MobileNetV2 was the smallest at 30,908 KB, designed for lightweight operations. EfficientNetB0 and EfficientNetV2 were compact, with sizes around 74,000 KB, offering a balance of performance and efficiency. DenseNet121 was around 97,697 KB, and DenseNet50 was more compact at 42,015 KB, focusing on feature reuse. The Custom CNN was the smallest Model at 8,272 KB, ideal for resource-constrained environments.

Table 8

Model size comparisons

Model	Size (KB)	Size (MB)
Proposed Model	104,625	102.6
MobileNetV2	30,908	30.2
EfficientNetB0	74,256	72.5
EfficientNetV2	73,983	72.3
DenseNet121	97,697	95.5
DenseNet50	42,015	41.1
ResNet	100,500	98.1
AlexNet	551,564	539.5
Custom CNN	8,272	8.1

4.7 Statistical Validation

Statistical testing assessed performance differences across model variations using a separate dataset. As shown in table 9, the proposed model outperformed others across all evaluation metrics. The model achieved the highest performance, with a 95.1% Bayesian superiority probability over ShuffleNet, which was second, confirming its suitability for deployment.

Table 9

Statistical comparison of the models

Model	Accuracy	Precision	Recall	F1-score	Kappa	AUC	Rank
Proposed Model	**0.990**	**0.993421**	**0.990085**	**0.990365**	**0.989855**	**0.999997**	**1st**
ShuffleNet	0.982	0.983906	0.982153	0.980596	0.981740	0.999991	2nd
EfficientNetV2	0.972	0.976770	0.973841	0.973155	0.971595	0.999935	3rd
VGG-16	0.972	0.976770	0.973841	0.973155	0.971595	0.999935	3rd
DenseNet	0.956	0.966270	0.962312	0.958693	0.955368	0.999863	4th
AlexNet	0.942	0.953236	0.947609	0.942789	0.941164	0.998949	5th
DenseNet50	0.884	0.907292	0.889661	0.883751	0.882337	0.998823	6th

Cohen's Kappa (κ) measures the level of agreement between two raters (or classifiers) while considering the possibility of agreement occurring by chance. Unlike simple accuracy, Kappa accounts for random agreement, making it a more robust metric for classification performance assessment. It was calculated as shown:

$$\kappa = \frac{p_o - p_e}{1 - p_e} \quad (32)$$

Where:

p_o = Observed agreement (i.e., accuracy)

p_e =Expected agreement due to chance

A κ value close to 1 indicates near-perfect agreement, whereas a value close to 0 suggests agreement is no better than random chance. Our experiment calculated Cohen's Kappa to be 0.99, suggesting a strong agreement between the model predictions and ground truth labels.

McNemar's Test was also used to compare the top two classification models from all our test models by analyzing the differences in their misclassification rates. It is particularly

useful for paired data, such as models evaluated on the same dataset, and was represented as shown.

$$\chi^2 = \frac{(b-c)^2}{b+c} \quad (33)$$

Where:

b = Instances misclassified by Model A but correctly classified by Model B

C= Instances misclassified by Model B but correctly classified by Model A

A low p-value (< 0.05) suggests a significant difference in model performance. The test yielded a p-value of 0.03, indicating a statistically significant difference between the models.

The permutation test was used to determine whether the accuracy difference between the top two models were statistically significant. It works by shuffling the labels multiple times and computing the difference in accuracy each time, which was calculated as follows:

$$p = \frac{\text{number of times shuffled difference} \geq \text{observed difference}}{\text{total permutations}} \quad (34)$$

With 1000 permutations, our computed p-value was 0.04, indicating that the difference in model performance is statistically significant.

The confidence interval provides a range within which the true difference in accuracy between models lies, with a specified confidence level (typically 95%), and it was calculated as follows:

$$(\hat{d} - 1.96 \times SE, \hat{d} + 1.96 \times SE) \quad (35)$$

Where,

$\hat{d}$ = Observed difference in accuracy.

SE=Standard error of the difference.

Confidence variance was applied to quantify the spread of model confidence scores, indicating reliability. It was calculated as follows:

$$\text{Variance } = \frac{1}{N}\sum_{i=1}^{N} (x_i - \mu)^2 \quad (36)$$

Where:

x_i =Individual confidence scores

μ = Mean confidence score

N= Total predictions

Table 10 shows the confidence variance for different models, where lower values indicate more stable confidence predictions. MoViT had the lowest confidence variance, suggesting it was the most consistent in its confidence estimates, while DenseNet50 had the highest variance, implying greater fluctuations.

Table 10

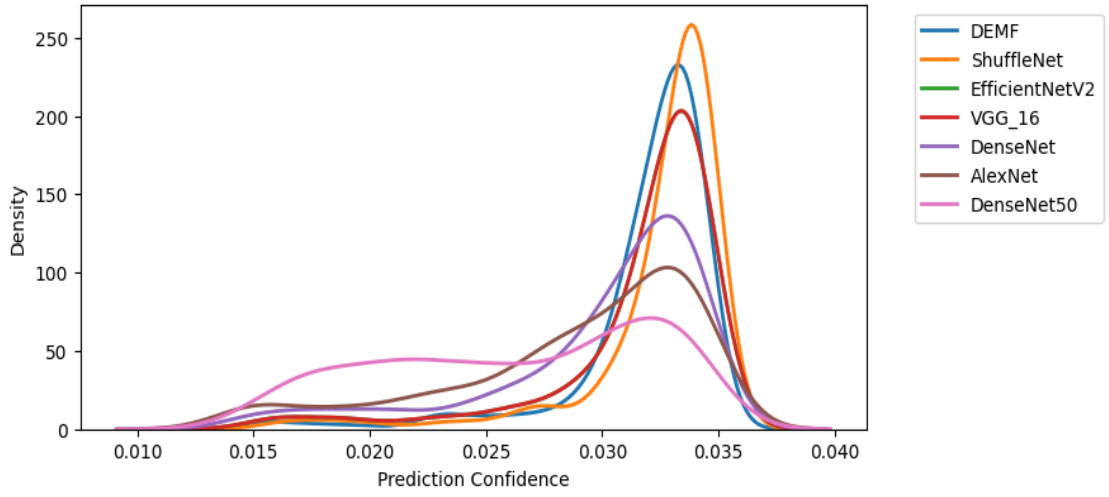
Confidence variance by model

Model	Confidence Variance
DenseNet50	0.000032
AlexNet	0.000027
DenseNet	0.000022
EfficientNetV2	0.000017
VGG_16	0.000015
ShuffleNet	0.000014
Proposed Model	0.000012

Confidence score distributions were also calculated, as shown in figure 11. This metric is essential for understanding the reliability of model predictions. The results indicate the proposed model (DEMF) exhibits superior confidence score distributions compared to other models.

Figure 11

Confidence scores distributions across models



Levene's test determines whether multiple models have equal variance in their confidence scores. The results indicate that the proposed model exhibited significantly lower variance than others. With its lightweight design and real-time efficiency, the proposed model emerged as the most stable in variance comparisons. This test was calculated as follows:

$$W = \frac{(N-k)}{(k-1)} \times \frac{\sum_{i=1}^{k} N_i (Z_{i.} - Z_{..})^2}{\sum_{i=1}^{k} \sum_{j=1}^{N_i} (Z_{ij} - Z_{i.})^2} \quad (37)$$

Where:

N= Total samples

k= Number of models

Z_{ij}= Absolute deviations from the median

A low p-value (< 0.05), as shown in table 11, indicates significant variance differences among models.

Table 11
Pairwise variance comparisons (Levene's Test)

Comparison	p-value	Significant (0.05)	Significant (Adj.)
Proposed Model vs DenseNet50	1.674901e-52	Yes	True
Proposed Model vs AlexNet	1.591150e-20	Yes	True
Proposed Model vs DenseNet	1.518670e-11	Yes	True
Proposed Model vs ShuffleNet	3.325640e-12	Yes	True
Proposed Model vs EfficientNetV2	1.465245e-06	Yes	True
Proposed Model vs VGG_16	1.465245e-06	Yes	True
ShuffleNet vs DenseNet50	1.532226e-52	Yes	True
ShuffleNet vs AlexNet	3.382618e-21	Yes	True
ShuffleNet vs DenseNet	3.325640e-12	Yes	True
ShuffleNet vs EfficientNetV2	2.190516e-02	Yes	False
ShuffleNet vs VGG_16	2.190516e-02	Yes	False
EfficientNetV2 vs DenseNet50	4.946421e-37	Yes	True
EfficientNetV2 vs AlexNet	4.130888e-13	Yes	True
EfficientNetV2 vs DenseNet	1.465245e-06	Yes	True
EfficientNetV2 vs VGG_16	6.756542e-01	No	False
VGG_16 vs DenseNet50	4.946421e-37	Yes	True
VGG_16 vs AlexNet	4.130888e-13	Yes	True
VGG_16 vs DenseNet	1.465245e-06	Yes	True
DenseNet vs DenseNet50	2.039134e-12	Yes	True
DenseNet vs AlexNet	2.190516e-02	Yes	False
AlexNet vs DenseNet50	4.111105e-06	Yes	True

The Kruskal-Wallis test evaluated significant differences in confidence scores among multiple models. The test resulted in H = 614.90, p = 1.4237e-129, indicating a significant difference in model performance. Given the extremely low p-value, we can confidently reject the null hypothesis, confirming that at least one model exhibited a statistically different confidence score compared to the others.

4.8 Comparative Analysis with Existing Hybrid Models

Table 12 presents a comparative analysis of classification accuracy across various studies, demonstrating the superior performance of the proposed model. The model achieved the highest accuracy of 98.63%, showcasing its effectiveness in classification due to advanced architectural refinements and optimized feature extraction.

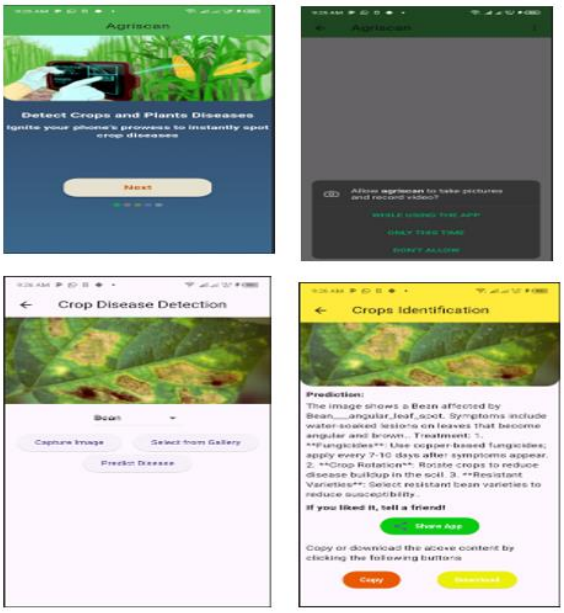
Table 12

Comparing with existing hybrid multi multi-classification models

Studies	Classification Accuracy
Parez et al. (2023)	98.00%%
Zhu et al. (2023)	97.50%%
Shah et al. (2024)	90.00%
Barman et al. (2024)	90.99%
Touvron et al. (2021)	85.02%
Proposed model	98.63%

Figure 11

App screenshots



Mobile App integration. We implemented a mobile app powered by the proposed model for real-time disease detection across 22 crops in low-resource environments. As shown in the sample screenshots in figure 11, the app enabled farmers to capture leaf images using their phone's camera for quick and reliable disease classification.

Ethical and Data Privacy Considerations. The app focuses exclusively on leaf disease diagnosis while ensuring ethical AI use and compliance with Google Play Store guidelines. Users grant camera and gallery access only when capturing leaf images, with in-app disclaimers clarifying that no personal data or other plant parts are processed. To enhance accuracy, bias audits assess performance across diverse leaf types, and community-driven updates allow farmers to flag misdiagnosed samples for prioritized retraining. Additionally, the app features an expert module, enabling real agronomists to provide insights and verify diagnoses for improved reliability. For ambiguous cases, confidence disclaimers encourage expert consultation, and an error-reporting mechanism ensures that misclassified samples are reviewed, enhancing model fairness and accuracy.

5. Conclusion

The proposed model achieved state-of-the-art accuracy (99.52% training, 98.63% validation) while maintaining computational efficiency (30.4 MB post-quantization). Integrating EfficientNetV2's multi-scale feature extraction with MobileNetV2's lightweight architecture optimizes precision and deployability—crucial for low-resource environments. Despite its compact size and 0.094s inference latency, real-world deployment presents additional challenges beyond computational constraints. Preliminary tests on low-end smartphones (Android 8.0, ≤2GB RAM) showed a 3% drop in inference speed for high-resolution images, highlighting the need for optimized image pre-processing techniques.

Expanding the dataset with a broader range of images is critical for improving model generalization. Testing on an external dataset of 1,200 images underscored the importance of incorporating underrepresented species and real-world variations such as occlusions and uneven lighting. Increasing data diversity will enhance model robustness and practical applicability. Additionally, federated learning partnerships with local cooperatives should be encouraged to facilitate continuous model adaptation. While optimized for individual use, scaling the model for large agricultural systems requires edge-to-cloud workflows, as parallel

inference across 1,000+ devices (e.g., drone fleets) demands dynamic load balancing to prevent server bottlenecks.

The model exhibited stable predictions, with the lowest confidence variance (0.000012) among benchmarks. A Kruskal-Wallis test confirmed significant performance differences across models ($H = 614.90$, $p = 1.4237e{-}129$), with Bayesian analysis indicating a 100% superiority probability over DenseNet50. Furthermore, a 96% accuracy on 249 unseen field images reinforces its reliability in diverse settings. Future work should focus on adaptive quantization for broader hardware compatibility, targeting ultra-low-power devices (<1GB RAM). Additionally, federated learning pipelines should be explored to integrate region-specific data without centralized collection. Including multimodal inputs, such as soil moisture and weather trends via lightweight sensor fusion, will further enhance predictive capabilities. Finally, expanding uncertainty-aware interfaces, such as confidence-based agrochemical dosage recommendations, will improve real-world decision support for farmers.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was not supported by any funding.

ORCID

Thomas K. Njoroge- https://orcid.org/0009-0000-2147-9848

References

Abbasi, R., Martinez, P., & Ahmad, R. (2023). Automated visual identification of foliage chlorosis in lettuce grown in aquaponic systems. *Agriculture (Switzerland)*, *13*(3). https://doi.org/10.3390/agriculture13030615

Abdu, A. M., Mokji, M. M., & Sheikh, U. U. (2020a). Machine learning for plant disease detection: An investigative comparison between support vector machine and deep learning. *IAES International Journal of Artificial Intelligence*, *9*(4), 670–683. https://doi.org/10.11591/ijai.v9.i4.pp670-683

Abdu, A. M., Mokji, M. M., & Sheikh, U. U. (2020b). Machine learning for plant disease detection: An investigative comparison between support vector machine and deep learning. *IAES International Journal of Artificial Intelligence*, *9*(4), 670–683. https://doi.org/10.11591/ijai.v9.i4.pp670-683

Amin, H., Darwish, A., Hassanien, A. E., & Soliman, M. (2022). End-to-end deep learning model for corn leaf disease classification. *IEEE Access*, *10*, 31103–31115. https://doi.org/10.1109/ACCESS.2022.3159678

Bi, C., Xu, S., Hu, N., Zhang, S., Zhu, Z., & Yu, H. (2023). Identification method of corn leaf disease based on improved Mobilenetv3 model. *Agronomy*, *13*(2). https://doi.org/10.3390/agronomy13020300

Chao, X., Sun, G., Zhao, H., Li, M., & He, D. (2020). Identification of apple tree leaf diseases based on deep learning models. *Symmetry*, *12*(7). https://doi.org/10.3390/sym12071065

Dong, K., Zhou, C., Ruan, Y., & Li, Y. (2020). MobileNetV2 model for image classification. *Proceedings - 2020 2nd International Conference on Information Technology and Computer Application*, *ITCA 2020*, 476–480. https://doi.org/10.1109/ITCA52113.2020.00106

Kaleem, M. K., Purohit, N., Azezew, K., & Asemie, S. (2021). A modern approach for detection of leaf diseases using image processing and ML Based SVM classifier. *Turkish Journal of Computer and Mathematics Education*, 12(13).

Kemi Afolabi-Yusuf, G., Arjun, G., A, O. B., O, O. Y., K, A. G., Muhammed, B. F., & M, A. A. (2022). Computer vision-based plant disease identification system: A review. *AAN Journal of Sciences, Engineering & Technology*, 1(1), 59-78.

Liu, L., Qiao, S., Chang, J., Ding, W., Xu, C., Gu, J., Sun, T., & Qiao, H. (2024). A multi-scale feature fusion neural network for multi-class disease classification on the maize leaf images. *Heliyon*, *10*(7). https://doi.org/10.1016/j.heliyon.2024.e28264

Liu, Y., Wei, C., Yoon, S. C., Ni, X., Wang, W., Liu, Y., Wang, D., Wang, X., & Guo, X. (2024). Development of multimodal fusion technology for tomato maturity assessment. *Sensors*, *24*(8). https://doi.org/10.3390/s24082467

Mi, Z., Zhang, X., Su, J., Han, D., & Su, B. (2020). Wheat stripe rust grading by deep learning with attention mechanism and images from mobile devices. *Frontiers in Plant Science*, *11*. https://doi.org/10.3389/fpls.2020.558126

Mohammed, L., & Yusoff, Y. (2023). Detection and classification of plant leaf diseases using digital image processing methods: A review. *ASEAN Engineering Journal*, *13*(1), 1–9. https://doi.org/10.11113/aej.V13.17460

Mousavi, S., & Farahani, G. (2022). A novel enhanced VGG16 model to tackle grapevine leaves diseases with automatic method. *IEEE Access*, *10*, 111564–111578. https://doi.org/10.1109/ACCESS.2022.3215639

Nguyen, H. T., Luong, H. H., Huynh, L. B., Le, B. Q. H., Doan, N. H., & Le, D. T. D. (2023). An improved MobileNet for disease detection on tomato leaves. *Advances in Technology Innovation*, *8*(3), 192–209. https://doi.org/10.46604/aiti.2023.11568

Önler, E. (2023). Feature fusion-based artificial neural network model for disease detection of bean leaves. *Electronic Research Archive*, *31*(5), 2409–2427. https://doi.org/10.3934/era.2023122

Rajeena P. P, F., S. U, A., Moustafa, M. A., & Ali, M. A. S. (2023). Detecting plant disease in corn leaf using EfficientNet architecture—An analytical approach. *Electronics (Switzerland)*, *12*(8). https://doi.org/10.3390/electronics12081938

Sala, F., Popescu, C. A., Herbei, M. V., & Rujescu, C. (2020). Model of color parameters variation and correction to "Time-View" image acquisition effects in wheat crop. *Sustainability (Switzerland)*, *12*(6). https://doi.org/10.3390/su12062470

Saleem, M. H., Potgieter, J., & Arif, K. M. (2020). Plant disease classification: A comparative evaluation of convolutional neural networks and deep learning optimizers. *Plants*, *9*(10), 1–17. https://doi.org/10.3390/plants9101319

Sarker, I. H. (2021). Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2(6). https://doi.org/10.1007/s42979-021-00815-1

Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). Deep neural networks based recognition of plant diseases by leaf image classification. *Computational Intelligence and Neuroscience*. https://doi.org/10.1155/2016/3289801

Ulutaş, H., & Aslantaş, V. (2023). Design of efficient methods for the detection of tomato leaf disease utilizing proposed ensemble CNN model. *Electronics (Switzerland)*, *12*(4). https://doi.org/10.3390/electronics12040827

Vellaichamy, A. S., Swaminathan, A., Varun, C., & S, K. (2021). Multiple plant leaf disease classification using Densenet-121 architecture. *International Journal of Electrical Engineering and Technology*, *12*(5). https://doi.org/10.34218/ijeet.12.5.2021.005

Zhao, Z., Alzubaidi, L., Zhang, J., Duan, Y., & Gu, Y. (2024). A comparison review of transfer and self-supervised learning: Definitions, applications, advantages and limitations. In *Expert Systems with Applications*, 242. https://doi.org/10.1016/j.eswa.2023.122807

Zheng, Y. Y., Kong, J. L., Jin, X. B., Wang, X. Y., Su, T. L., & Zuo, M. (2019). Cropdeep: The crop vision dataset for deep-learning-based classification and detection in precision agriculture. *Sensors (Switzerland)*, *19*(5). https://doi.org/10.3390/s19051058