Real-Time Disease Identification of Banana Plants Using Neural Network Models

Rushit R. Rivankar, Smitha N. Pai, Abhishek Rhisheekesan, Deekshitha, Lohith Prakash, Sunil V G, Abel Philip Joseph

Abstract— **The consumption of agricultural products is fundamental to human survival. Enhancing agricultural productivity and sustainability necessitates the effective monitoring and nurturing of healthy crops. In this regard, one of the most important study areas is using sophisticated neural network models to identify plant diseases. Four well-known convolutional neural network (CNN) architectures—InceptionV3, ResNet50V2, VGG16, and MobileNet—that use transfer learning for real-time disease identification in banana plants are thoroughly compared in this work. A total of 11 disease variants were classified, and performance metrics were computed for each model. Stratified random sampling was employed to ensure balanced representation of classes during training and validation. Evaluation using Cross-Validation, ROC-AUC, and confusion matrices ensured robust and interpretable classification results. Among the models, ResNet50V2 obtained the maximum accuracy of 99.52% on in-distribution test data, whereas MobileNet reached 90% accuracy on real-time datasets. The results were validated by an agricultural expert, confirming their practical reliability. This study provides useful information about the advantages and disadvantages of each architecture, along with practical suggestions for their implementation in actual agricultural environments to help farmers identify and treat diseases promptly.**

Index Terms— **Agriculture, Banana plant, Convolutional neural network, Identification of plant diseases, Deep learning models**

I. Introduction

Agriculture is essential to both economic stability and global food security, providing sustenance and livelihoods for billions of people. It has a crucial part in the economic transformation of developing nations, boosting productivity, increasing income, and reducing poverty and hunger [1]. With a projected 9 billion people on the planet by 2050, food production will need to rise by 70%. Efficient and sustainable agricultural practices are essential to meet this demand and ensure food security [2].

Plant diseases threaten crop yields and quality, posing significant risks to food availability and economic stability. Timely and precise illness detection is desperately needed to prevent widespread crop damage and financial losses [3]. Traditional plant disease detection relies on visual inspections by experienced farmers or experts like professors, researchers, and officers working in the agricultural field. This approach, which entails looking for outward signs of discoloration, spots, or malformations in plants, has drawbacks such as a high need for professional experience, protracted diagnosis delays, and low productivity. [4]. In some cases, laboratory tests are used to confirm a specific pathogen [5].

Plant disease detection has been transformed by technological developments, especially in the areas of computer vision and machine learning. Researchers derived feature information from segmented images to identify and distinguish abnormal conditions [6]. Automated systems can analyze images of plants to identify diseases accurately and quickly [7]. These technologies enable early detection and prompt intervention, reducing disease spread and minimizing economic losses. Image classification problems are a great fit for neural networks, particularly Convolutional Neural Networks (CNNs) [8]. They are able to accurately diagnose diseases by directly identifying visual patterns in pictures. CNNs can correctly distinguish between different diseases after being trained on enormous amounts of photos of both healthy and unhealthy plants. Neural networks, especially Convolutional Neural Networks (CNNs), are great for detecting plant illnesses because of their remarkable performance in image classification tasks [9]. A study of various models and their implications for banana plant disease application is essential to obtain a robust model.

II. Literature Survey

The rapid advancements in deep learning have significantly enhanced the capability of automated systems to detect plant diseases, a crucial aspect of ensuring food security and agricultural productivity. Recent studies have demonstrated the potential of neural network models in

Manuscript received Jan 23, 2025; revised May 26, 2025.

Rushit R. Rivankar is an undergraduate student at the School of Computer Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, India. (e-mail: rushit.rivankar@learner.manipal.edu).

Smitha N. Pai is a Professor at the School of Computer Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, India (Corresponding author e-mail: smitha.pai@manipal.edu).

Abhishek Rhisheekesan is the CEO of aiRender Technology Pvt Ltd., Bangalore, India, (e-mail: abhishek.airender@gmail.com).

Deekshitha is the Head of 3-D Graphics Dev Team, aiRender Technology Pvt Ltd., Bangalore, India (e-mail: deekshitha.airender@gmail.com).

Lohith Prakash is a Technical Lead of the Full-Stack Software Dev Team, aiRender Technology Pvt Ltd., Bangalore, India (e-mail: lohith.airender@gmail.com).

Sunil V. G is an Assistant Professor of the Agricultural Extension Communication Centre, Kerala Agricultural University, Mannuthy, Kerala, India (e-mail: sunil.vg@kau.in).

Abel Philip Joseph is an undergraduate student of the Department of Computer Science and Engineering, Indian Institute of Information Technology, Kottayam, Kerala, India (e-mail: abel.airender@gmail.com).

accurately identifying various plant diseases from image data, thereby offering a promising tool for early intervention and management.

The use of deep learning methods for plant disease detection has been investigated by numerous researchers. The results of the study [10] provide a thorough analysis of the use of convolutional neural networks (CNNs) to the identification of plant diseases from photos of leaves. The effectiveness of CNNs in achieving high accuracy rates, demonstrating their potential as a reliable method for disease detection in agricultural practices, is highlighted in the study. Similarly, [11] discusses the application of deep learning models, emphasizing their robustness in handling complex patterns and variations in plant disease symptoms.

Comparative studies have also been used to evaluate the efficacy of various neural network topologies in the context of plant disease detection. For instance, [12] assesses how well a number of deep learning models, including CNNs, ResNet, and Inception, can recognize diseases in images of plants. According to their findings, some architectures, such as ResNet, perform better than others in terms of accuracy and computational efficiency, even if all models demonstrate optimistic outcomes.

More complex plant disease detection systems have been developed as a result of developments in image processing and machine learning. The combination of deep learning and image-based detection methods is discussed in [13], highlighting the significance of sizable datasets and excellent images for building reliable models. The assessment sheds light on the difficulties and potential paths in this quickly developing subject.

The application of deep learning for specific crops, such as bananas, has also garnered attention. The work in [14] focuses on detecting banana diseases using deep learning models, addressing unique challenges posed by the crop's morphology and disease manifestations. Their study highlights the adaptability of neural networks in tailoring solutions for specific agricultural contexts, thereby enhancing the precision and reliability of disease detection.

The application of deep learning to the identification of plant diseases is a significant development in agricultural technology. Comparative analyses of different neural network models highlight how crucial it is to choose the right design in order to get the best outcomes. It is anticipated that further research and developments in image processing, data augmentation, and model training will improve the precision and effectiveness of these systems as the field develops, supporting sustainable farming methods.

After a thorough analysis of the work conducted by various authors, it is noticed that disease identification for a whole banana plant has not yet been carried out. The identification of 11 variants of disease in banana plants is still a challenge. The primary objective of this research is to conduct a comparative analysis of various neural network models, evaluating their performance using a dataset comprising images of banana plants affected by 11 distinct diseases. The dataset includes a diverse range of banana diseases, such as Black Sigatoka, Panama, and others. Each disease presents unique symptoms and challenges, impacting banana production differently. Important measures like accuracy, precision, recall, F1-score, and computing efficiency will be used to evaluate the models. Additionally, the study will examine the training and validation standards of each model, employing cross-validation techniques and model fitting analysis.

This study aims to leverage the strengths of four prominent CNN architectures —InceptionV3, ResNet50V2, VGG16, and MobileNet to develop a robust system for detecting diseases in banana plants, a crop of significant economic value. The insights gleaned from this comparative study will not only highlight the strengths and limitations of each model but also provide practical recommendations for their deployment in real-world agricultural settings. By focusing on outcomes obtained from the evaluation of these models as well as the impacts of parameters, this research aims to offer actionable guidance for better incorporation of automation for plant disease detection, which will help in enhancing agricultural productivity and sustainability.

A. Overview of Plant Disease Detection using CNN

Convolutional neural network (CNN) architectures and their variants have been used by researchers to categorize and identify plant diseases [15]. This approach involves configuring a certain CNN architecture to suit the requirements or utilizing any of the existing popular architecture that has been proven to be reliable for such classification problems.

The process involves training the Neural Network model with a large dataset containing sample images of diseased and non-diseased plants, wherein each image is labeled appropriately with the disease names. The model reads these images and adopts the respective features of each disease throughout multiple iterations. The model can be tuned for improvising its efficiency or performance by adjusting various hyperparameters like Learning Rate, Batch Size, etc., which control the model’s adaptability towards learning the features of the input images during training.

After the model has been trained in accordance with the specifications, it is tested using input images of both diseased and non-diseased plants. The model reads these images to determine its features and predicts the score (probability) for each disease class; the class with the highest score is chosen to be the predicted disease for the sample image. Metrics like Accuracy, Precision, Recall, and F1-score are used to evaluate the model's performance by comparing the test predictions with the actual illness diagnoses. Figure 1 shows the steps followed to train and test samples.

In this study, we are comparing the performance of four state-of-the-art convolutional neural network architectures: MobileNet, ResNet50V2, InceptionV3, and VGG16. Each model is pre-trained on the ImageNet dataset and modified for our particular classification task by substituting custom fully connected layers for the top layers. The models are trained and evaluated over multiple epochs to ensure proper performance analysis. Additionally, the impact of varying relevant hyperparameters is explored, such as the learning rate, batch size, and the number of units in the fully connected layers on the model performance.

During the training phase, the accuracy and loss of the training and validation sets were monitored. As a result, the optimal model and hyperparameter configurations could be identified. The outcomes are examined to identify each architecture's advantages and disadvantages in relation to our particular goal, offering valuable information about the best ways to use transfer learning in picture categorization.

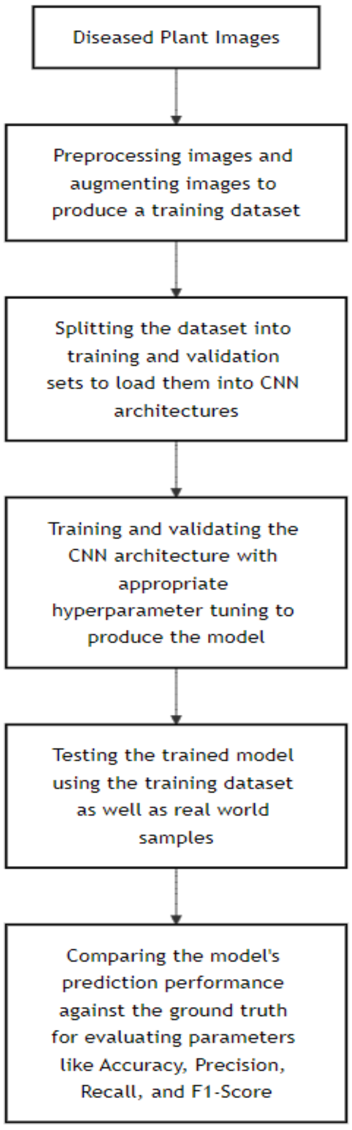

Diseased Plant Images
Preprocessing images and augmenting images to produce a training dataset
Splitting the dataset into training and validation sets to load them into CNN architectures
Training and validating the CNN architecture with appropriate hyperparameter tuning to produce the model
Testing the trained model using the training dataset as well as real world samples
Comparing the model's prediction performance against the ground truth for evaluating parameters like Accuracy, Precision, Recall, and F1-Score

Fig. 1. Flow Diagram of the general process of training and evaluation of a CNN Model for Plant Disease Detection.

B. Model Architectures

ResNet50V2

ResNet50V2 introduces modified residual connections, allowing the network to learn residual functions more efficiently by using pre-activation blocks. With this modification, vanishing gradient problems can be further mitigated by training very deep networks with better gradient flow. ResNet50V2's residual connections facilitate the training of extremely deep networks, leading to enhanced performance on complex tasks. However, the additional complexity introduced by these connections, along with the pre-activation structure, can increase the number of parameters, requiring careful consideration of computational resources. [16,17,18]

VGG16

VGG16 is characterized by its simplicity, featuring 16 layers with small 3x3 convolutional filters. It follows a straightforward design with multiple convolutional and pooling layers, culminating in three fully connected layers for classification. The uniform architecture simplifies implementation and promotes feature reuse, making it effective for various image classification tasks. VGG16 tends to be computationally intensive and requires a larger number of parameters compared to some other architectures [17,19].

InceptionV3

InceptionV3 introduces the concept of inception modules. These modules utilize multiple filter sizes concurrently, allowing the network to capture features at different scales. The inception modules help in efficient information extraction, reducing the risk of losing important details during feature extraction. Although InceptionV3 excels in feature extraction, it may have higher computational requirements due to parallel processing in the inception modules [22].

MobileNet

MobileNet is specifically designed for mobile and edge devices, emphasizing lightweight structures. It uses Depthwise separable convolution divides the convolution process into two stages: pointwise convolution, which aggregates the results, and depth-wise convolution, which applies a filter to each input channel independently. This factorization is perfect for embedded and mobile devices since it minimizes computation and model size [23]. MobileNet is ideal for contexts with limited resources because it strikes a compromise between model size and accuracy.

C. Banana Dataset Description

The images utilized for this comparative analysis are of diseased banana plants. A total of 11 different diseases are incorporated in this dataset, with a total of 1887 images, wherein about 100-200 images of each disease variety are considered. This dataset was created by compiling images from available open sources [25][27][28]. The dataset consists of pre-augmented images as per the availability in the respective source. The disease varieties considered are:

Black Sigatoka

Black Sigatoka, also known as black leaf streak, is caused by the fungus Mycosphaerella fijiensis. It produces streaks and spots on leaves that darken and spread, reducing photosynthesis. Severe infections can lead to significant yield losses and premature ripening of fruit [30]. Fig. 6 shows an example of a banana leaf infected with Black Sigatoka disease.

Yellow Sigatoka

This fungal disease, also known as Sigatoka leaf spot, caused by Pseudocercospora musicola, produces yellowish streaks and spots on banana leaves. Like black Sigatoka but generally less aggressive, it can still lead to reduced photosynthetic activity and yield if not managed properly [31]. Fig. 7 shows an example of banana leaf affected by Yellow Sigatoka Disease.

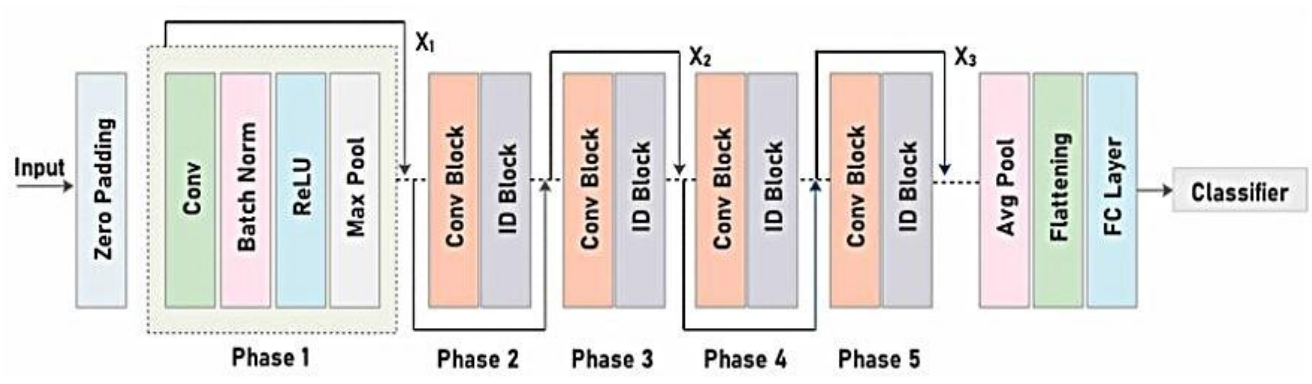


Fig. 2. Architecture Diagram of ResNet50V2 [20]

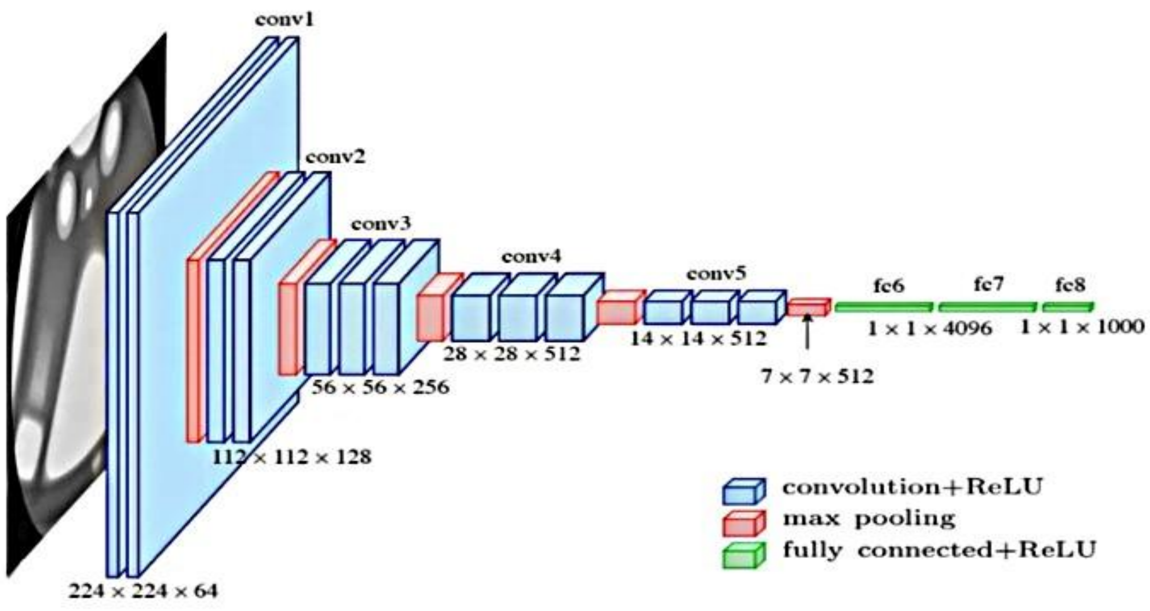


Fig. 3. Architecture Diagram of VGG16 [21]

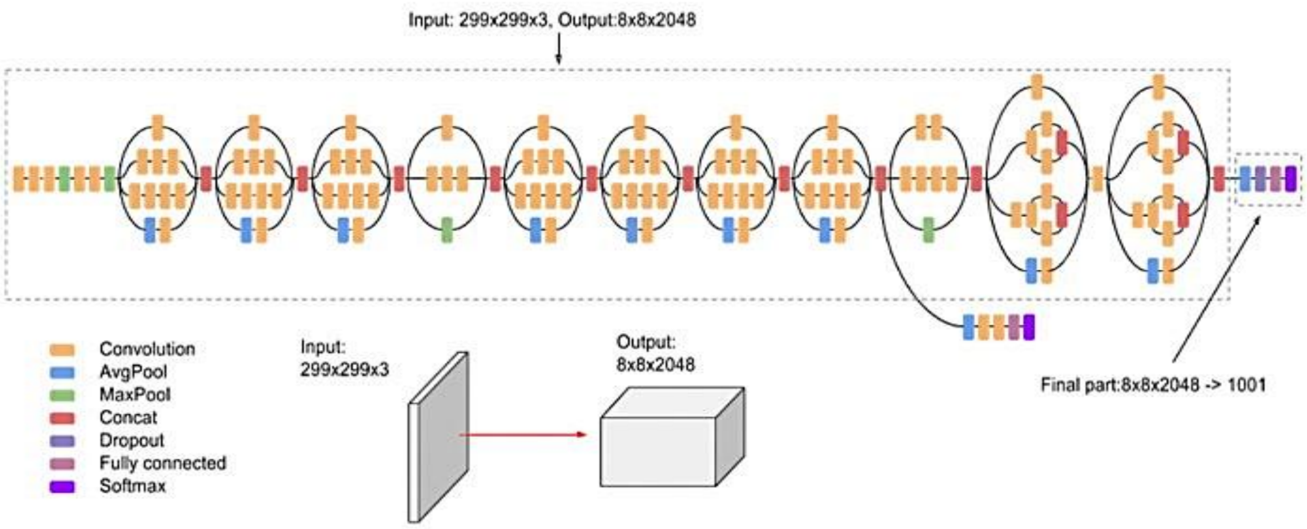


Fig. 4. Architecture Diagram of InceptionV3 [22]

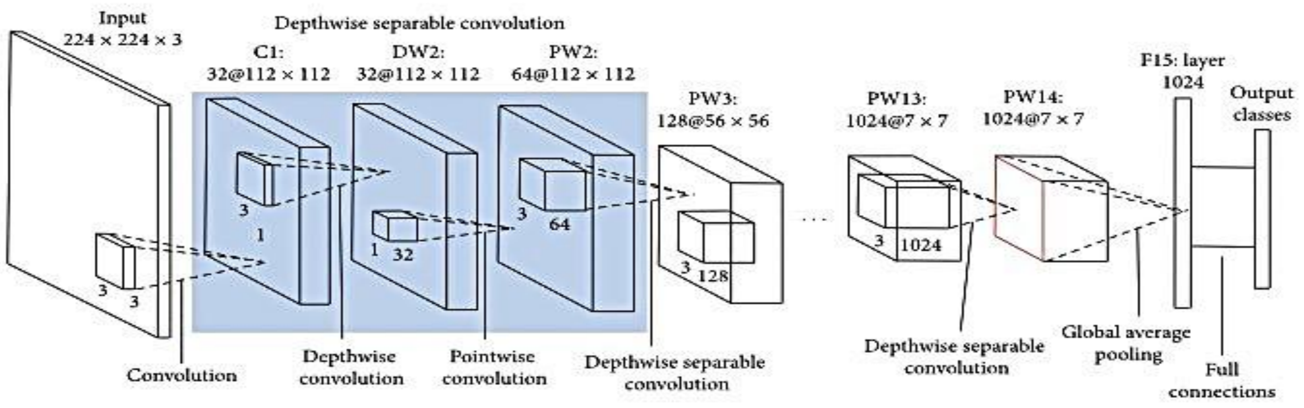


Fig. 5. Architecture Diagram of MobileNet [24]

Cordana

Cordana leaf spot, caused by the fungi Cordana musae and C. johnstonii, results in oval to diamond-shaped pale brown leaf spots, measuring up to 10 cm, appearing on the upper surface of the leaves, often bordered by a yellow edge. These spots frequently merge, damaging larger sections of the leaf, which eventually turn brown and dry out. On the underside, spores form in abundance, leading to greyish-brown, hairy lesions [32]. Fig. 8 shows the banana leaves infected with Cordana disease.

Pestalotiopsis

Pestalotiopsis is a fungal disease that causes leaf spots, fruit rot, and crown rot. The pathogen can survive in plant debris and soil, leading to recurring infections if not managed. Infected tissues exhibit dark, necrotic spots with concentric rings [33]. Fig. 9 shows banana leaves infected with Pestalotiopsis disease.

Boron Deficiency

Symptoms of deficiency include reduced leaf size, leaf curling, deformation of the leaf blade, white stripes appearing across the veins on young leaves, thickened secondary veins, and inhibited root and flower development [34]. Fig. 10 shows an image of Boron deficiency in the banana leaves.

Potassium Deficiency

The deficiency symptoms include yellowing or orange discoloration of older leaves, leaf margin scorching, a reduction in overall leaf area, and curving of the midribs. It can also cause leaves to become congested, delay flower initiation, and ultimately reduce both yield and quality [34]. Fig. 11 shows potassium deficiency in banana leaves.

Panama Disease

The soil-borne fungus Fusarium oxysporum f.sp. cubense is the cause of Panama disease, also known as Fusarium wilt. It infects the roots and vascular system, causing wilting and yellowing of leaves. It's one of the most destructive banana diseases, particularly for the Cavendish variety. [35]. Fig. 12 shows Panama disease on the stem of the banana plant.

Bacterial Soft Rot

This disease is caused by bacteria such as the Erwinia species. It leads to the softening and rotting of plant tissues, primarily affecting the rhizome and pseudostem. Infected areas emit a foul odor and turn mushy, causing the plant to collapse [36]. Fig. 13 shows a banana stem affected by Soft Rot disease.

Pseudostem Weevil

Pseudostem weevil (Odoiporus longicollis) larvae burrow into the pseudostem of banana plants, causing internal damage that weakens the plant and leads to collapse. Symptoms include boreholes and oozing sap from the pseudostem [37]. Fig. 14 shows Pseudostem Weevil disease in bananas.

Fruit Scarring Beetle

The fruit scarring beetle damages banana fruits by feeding on the peel, leaving scars and blemishes. This affects the marketability of fruit. The beetles also lay eggs in the scars, leading to further damage as larvae feed [38]. Fig. 15 shows bananas infested by the Fruit Scarring Beetle.

Aphids

Aphids are tiny insects that feed on sap and can seriously damage banana trees. By feeding on the sap of the plant, they weaken it, causing growth retardation, yellowing of the leaves, and possibly the spread of viral illnesses. They secrete honeydew, which fosters sooty mold growth on leaves [39]. Fig. 16 shows banana stems infested by Aphids.

TABLE I
DATASET DISTRIBUTION

S.No.	Banana Disease	Quantity
1.	Black Sigatoka	200 Images
2.	Yellow Sigatoka	200 Images
3.	Cordana	162 Images
4.	Pestalotiopsis	173 Images
5.	Boron Deficiency	100 Images
6.	Potassium Deficiency	200 Images
7.	Panama	102 Images
8.	Bacterial Soft Rot	200 Images
9.	Pseudostem Weevil	200 Images
10.	Fruit Scarring Beetle	150 Images
11.	Aphids	200 Images

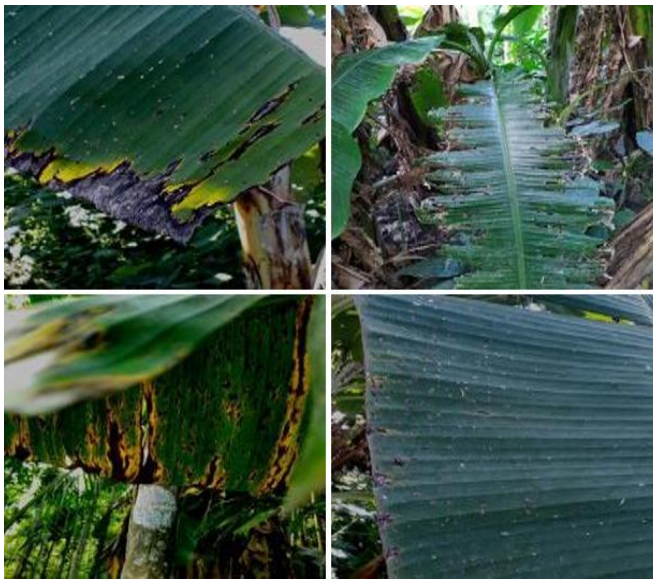


Fig. 6. Banana Black Sigatoka Disease [25]

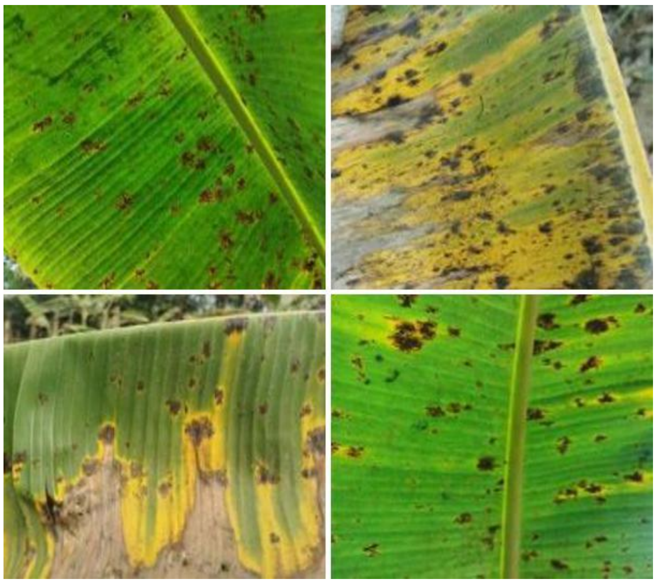


Fig. 9. Banana Pestalotiopsis Disease [27]

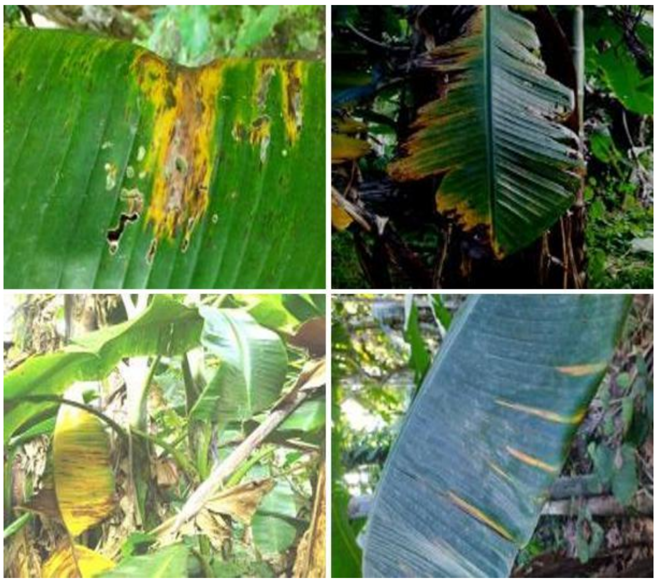


Fig. 7. Banana Yellow Sigatoka Disease [25]

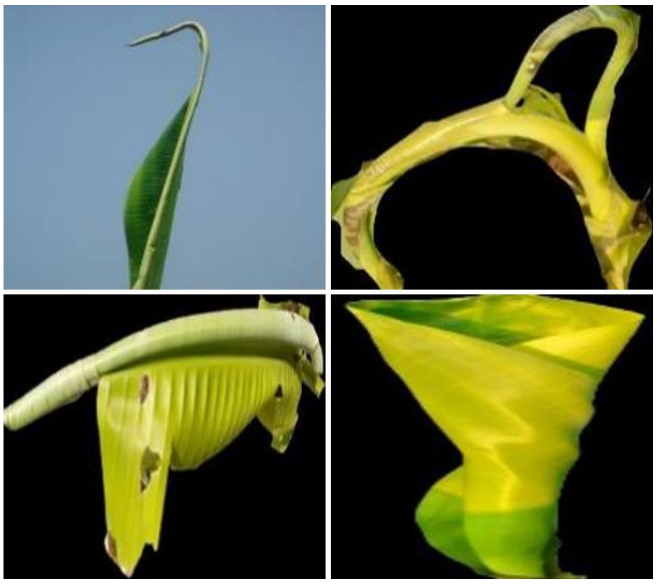


Fig. 10. Banana Boron Deficiency Disease [28][29]

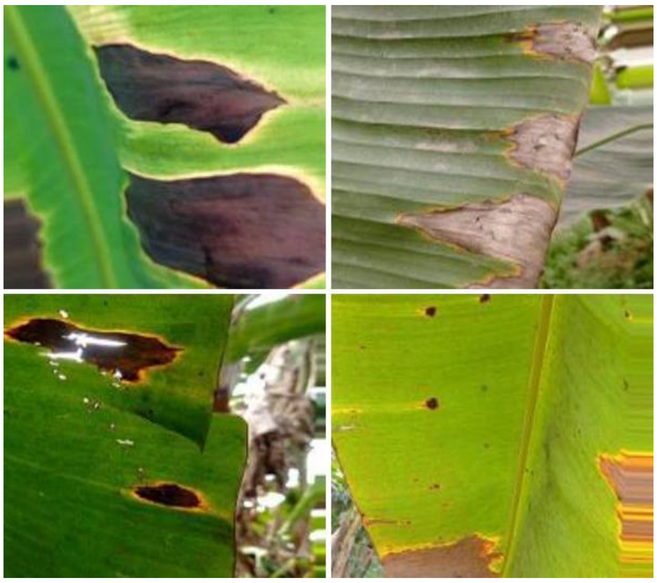


Fig. 8. Banana Cordana Disease [27]

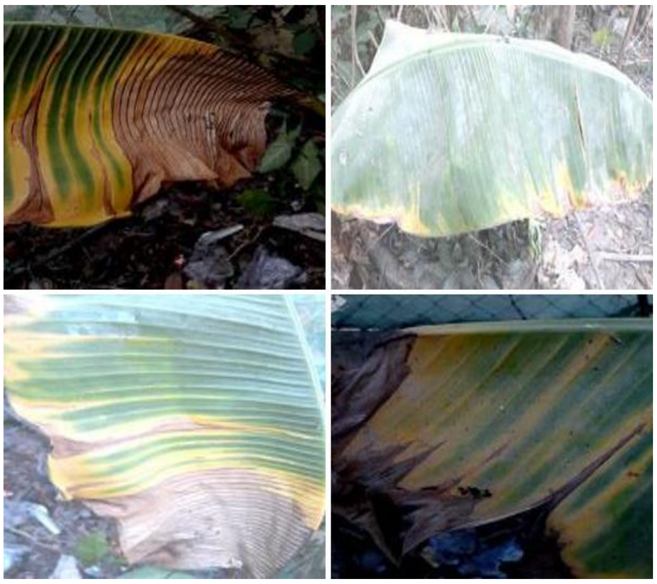


Fig. 11. Banana Potassium Deficiency Disease [24]

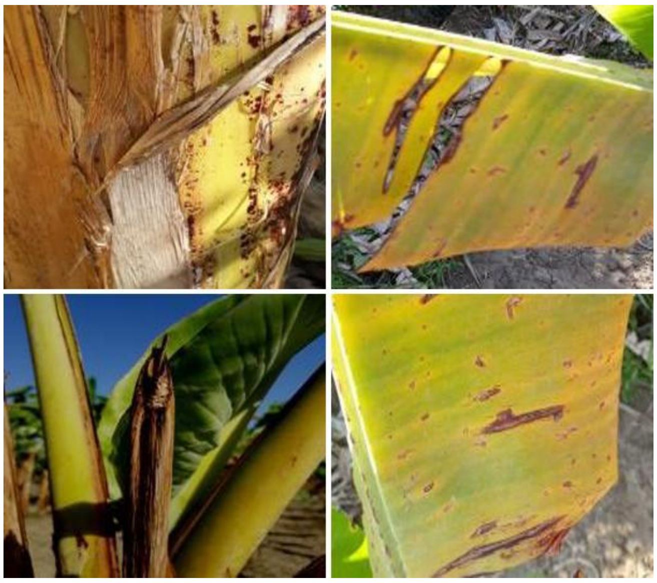


Fig. 12. Banana Panama Disease [24]

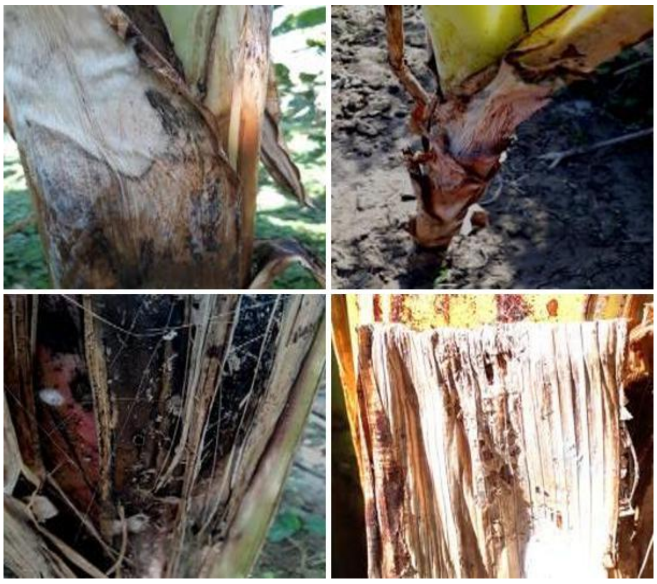


Fig. 13. Banana Bacterial Soft Rot Disease [24]

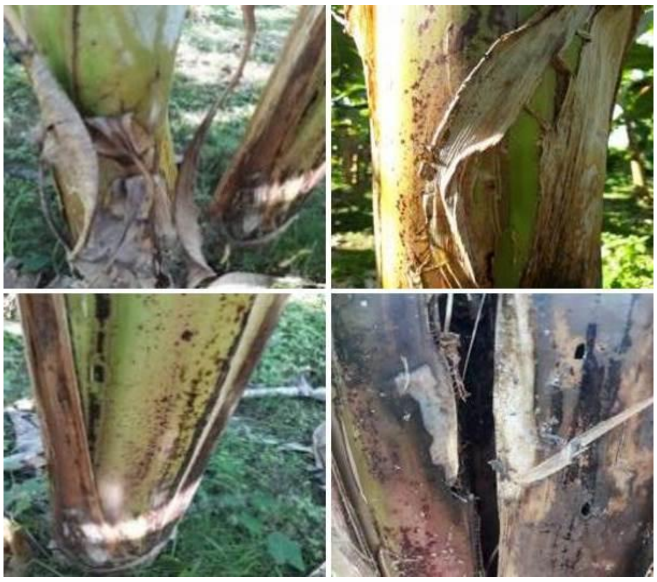


Fig. 14. Banana Pseudostem Weevil Disease [26]

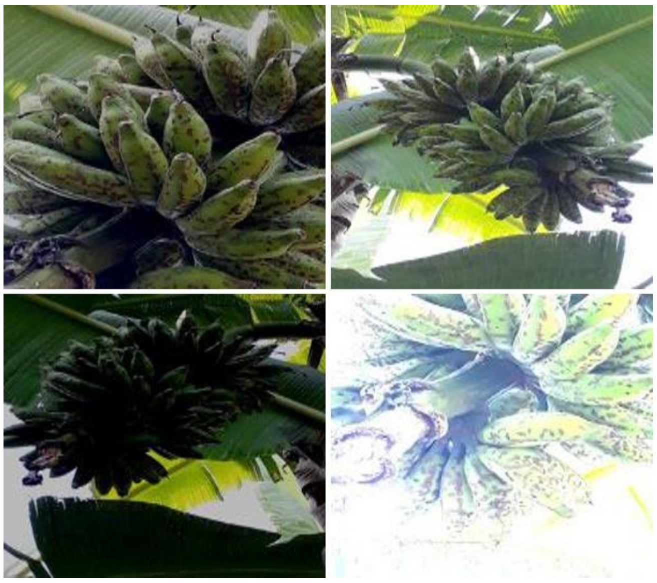


Fig. 15. Banana Fruit Scarring Beetle Disease [24]

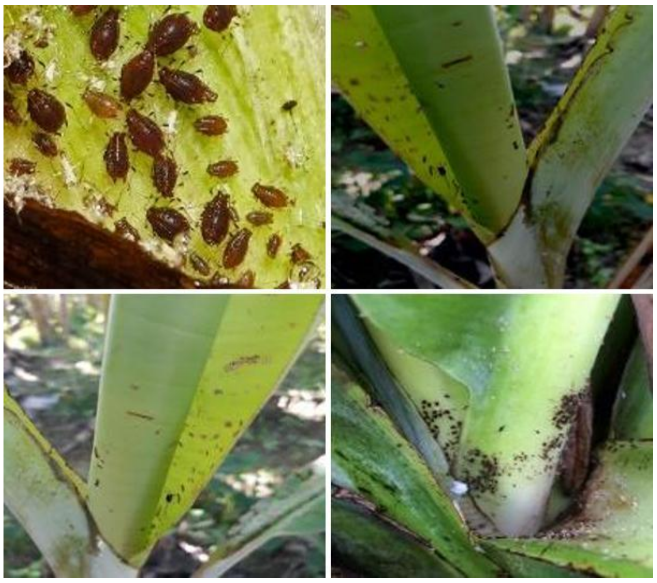


Fig. 16. Banana Aphids Disease [24][25]

III. Methodology

A. Prerequisites to Model Training

In preparing for banana disease detection, a robust and well-curated dataset is crucial for effective model training. Several key steps were taken to ensure the dataset's consistency and usability. All images were renamed using a standardized format (`banana_diseasename_index.jpg`), embedding both the disease type and a unique identifier for each image. This renaming convention not only helped in labeling the dataset but also simplified the process of generating corresponding labels. A CSV file was created with two columns—filename and label—to map each image to its associated disease, ensuring a clear structure for the model's input. Additionally, all images were resized to 256x256 pixels, standardizing their dimensions and preparing them for uniform processing in the neural network.

The training environment was established in Jupyter Notebook (Python 3.11), using TensorFlow and Keras for model development. These frameworks provide powerful resources for creating, honing, and implementing deep learning models, particularly for image classification applications. Important libraries were also included, such as `numpy` for numerical calculations and `pandas` for data manipulation. Moreover, TensorFlow's image preprocessing tools like `ImageDataGenerator` were employed to handle real-time data augmentation, further enhancing the training process by providing more variability in the data.

To streamline the training process, transfer learning techniques were employed using pre-trained models such as VGG16, ResNet50V2, InceptionV3, and MobileNet from Keras applications. These models allowed the reuse of learned features from large-scale datasets, speeding up the training process and improving accuracy for banana disease detection. The dataset, along with its labels, is loaded into the Python environment, where the Pandas library is used to ensure that each image's full path is correctly mapped in the DataFrame. This ensured seamless access to the images during the model training phase.

B. Stratified Random Sampling

CNN models, such as MobileNet, InceptionV3, ResNet50V2, and VGG16, have achieved remarkable success in a variety of applications, ranging from image classification to object detection. Despite their effectiveness, these models often struggle with class imbalance in datasets, which can lead to suboptimal performance.

Stratified random sampling is based on the concept of dividing a population into distinct subgroups, or strata, and then sampling from each subgroup proportionally. This approach ensures that each subgroup is represented in the sample according to its prevalence in the population. Stratified random sampling is a statistical technique used to ensure that different subgroups or strata within a population are adequately represented in a sample. In machine learning, stratified sampling is used to produce training and validation datasets that preserve the original dataset's class distribution.

The current project implements a proportional stratified random sampling approach wherein the size of the sample from each stratum is proportional to the size of the stratum. This can be mathematically expressed as $ni = [Ni / N] \times n$, where ni is the number of samples from stratum i, Ni be the number of instances in the ith stratum, for $i \in \{1, 2, K\}$, N is the total size of the population, n is the total sample size and K be the number of strata (or classes).

This sampling technique was incorporated into the project to address potential class imbalances and enhance the neural network model's performance. The dataset, containing image filenames and corresponding class labels, was first loaded into a Pandas DataFrame. During the data splitting phase, the `train_test_split` function from Scikit-learn was used, with the `stratify` parameter set to the label column. This ensured that the class distribution in both training and validation sets matched the proportions in the original dataset, preventing any skewed representation.

By maintaining the same class proportions in both the training and validation datasets, the neural network receives balanced input from all classes, leading to more accurate and generalized learning. The class distribution in both sets was verified post-split to confirm the effectiveness of the stratified sampling process, ensuring that each class had proportional representation like the overall dataset. The differences in performance, with and without the use of stratified random sampling, are shown in the results.

C. Implementation of Model Architectures and Additional Data Augmentation

To leverage the powerful feature extraction capabilities of all the different model architectures, a transfer learning approach is introduced. Each model loaded is pre-trained on the ImageNet dataset, excluding its top classification layer. This provides a robust convolutional base capable of extracting high-level features from input images.

The architecture was then augmented with custom fully connected layers tailored to our specific classification task. The process involved the following steps:

Global Average Pooling

The output of the convolutional base is fed into a Global Average Pooling layer, which reduces the spatial dimensions while retaining the salient features.

Fully Connected Layer:

Complex representations of the pooled features are learned by adding a Dense layer with ReLU activation. The units in the Dense Layer varied among 5 values.

Output Layer

Finally, a Dense layer with several units equal to the number of classes in our dataset and a SoftMax activation function was appended to produce a probability distribution over class labels.

The weights of previously trained layers are frozen to avoid overfitting and speed up the training process, which is achieved by disabling the trainable attribute of each layer in the base model. Consequently, only the weights of the newly added fully connected layers are updated during the training phase.

To enhance the diversity of the training dataset and improve the generalization capability of the model, an extensive data augmentation strategy using the ImageDataGenerator class from Keras is employed. The data augmentation parameters included:

1) Rescaling: Normalizing pixel values by scaling them to the range [0, 1].
2) Shear Transformation: Applying shear transformations of 0.2.
3) Zoom Transformation: Applying zoom transformations of 0.2.
4) Horizontal Flipping: Randomly flipping images horizontally.

These augmentations were applied to the training dataset while ensuring that the validation set remained unaltered to provide an unbiased evaluation of the models' performance.

D. Hyperparameter Tuning for Comparative Analysis

To analyze the performance of each model during the training and validation phases on the Banana Disease Detection Dataset, we performed hyperparameter tuning across the four models and compared their performance under various configurations. We focused on five key hyperparameters: optimizer, learning rate, number of epochs, optimizer, batch size, and the number of units in the fully connected dense layer. Each of these hyperparameters is tested with five distinct values, allowing thorough access to the impact of different settings on the models' accuracy and generalization ability. This approach helped in identifying the optimal configuration for each model.

Optimizers adjust model parameters iteratively during training to minimize a loss function, enabling neural networks to learn from data. There are numerous optimizers available for training machine learning models, each with specific benefits and drawbacks. While some optimizers are more general-purpose, others are more appropriate for specific kinds of models or data [40]. In the current work, five types of optimizers are utilized:

Stochastic Gradient Descent with Momentum

Stochastic Gradient Descent (SGD) uses a small, randomly chosen portion of the data (a "mini batch") rather than the complete dataset to update model parameters. Adding a momentum term to SGD improves this and enables the optimizer to stay on course even with a small local gradient [40]. In this project, the momentum term has been set to 0.9.

RMSProp

RMSProp is an optimization algorithm in which, instead of accumulating the sum of squared gradients, it uses an exponentially decaying average of these squares. This approach helps prevent the learning rate from decaying too quickly, leading to more stable updates and improved convergence, particularly in non-stationary and noisy settings. [40].

AdaDelta

AdaDelta is an optimization algorithm like RMSProp, but it eliminates the need for a predefined learning rate. It is more resilient and self-adjusting during training because it dynamically adjusts the learning rate by determining the update scale using an exponentially decaying average of the gradients and the squared gradients. [40]. The learning rate field was set to 1 while initializing this optimizer.

Adaptive Moment Estimation

The optimization approach known as Adaptive Moment Estimation (Adam) combines ideas from SGD with momentum and RMSProp. Similar to the RMSProp, it computes the average of the gradients and squared gradients with exponential decay. To speed up convergence and better traverse the loss landscape, it also adds a momentum factor. This makes Adam well-suited for handling sparse gradients and noisy data [40].

Nesterov-Accelerated Adam

Nesterov-accelerated Adaptive Moment Estimation (Nadam) is an optimization algorithm that merges the benefits of Adam and Nesterov momentum [41]. It incorporates the adaptive learning rate adjustments from Adam with the look-ahead gradient updates from Nesterov momentum, improving both convergence speed and stability.

The size of weight updates during each optimization process iteration is determined by the learning rate. The step size taken in the direction of the negative gradient during backpropagation is determined by this scalar. In order to modify the weights, backpropagation entails moving the error between the expected and actual outputs backward through the network. While a high learning rate may cause the model to overshoot and maybe miss the ideal solution, a low learning rate may lead to slow convergence and an increased danger of becoming caught in local minima. For effective training and improved model performance, the learning rate must be properly adjusted. [42]. In our project, we have utilized these 5 values of learning rate: 0.001, 0.0025, 0.005, 0.0075, and 0.01

A hyperparameter called the number of epochs determines how many times the learning algorithm will run through the whole training dataset. The internal parameters of the model are updated using each sample in the training set in each epoch. The model may underfit if it hasn't learned enough from the data, or overfit if it has learnt the training data too well and performs poorly on new, unknown data. The number of epochs chosen can have an impact on training. Keeping an eye on training metrics like accuracy and loss might assist figure out how many epochs are right to balance learning and generalization [43]. For training the models, we considered the following five values for the number of epochs based on standard practices and requirements for training to effectively evaluate performance and determine the optimal training duration.: 10, 20, 30, 40, and 50.

A hyperparameter called the batch size determines how many samples must be processed before the internal parameters of the model are modified. It affects training speed and stability; larger batches offer more precise gradient estimates but use more memory, while smaller batches provide more frequent updates [43]. To investigate a variety of possibilities and evaluate their effects on training dynamics and model performance, five batch sizes—4, 8, 16, 32, and 64—are employed.

A fully linked layer's output space size is determined by a hyperparameter called the number of units in that layer [44]. More units enable the model to be more expressive and maybe perform better, which impacts the model's ability to learn complicated representations. However, increasing the number of units can also raise the risk of overfitting if not properly regularized. To evaluate its impact on model performance, we tested the following five values for the number of units in the fully connected layer: 256, 512, 1024, 2048 and 4096.

IV. Results and Discussion

The training process is carried out using four models across five different hyperparameter values. This helps in identifying the most suitable hyperparameter settings and determines the optimal number of epochs for each model, balancing performance and computational efficiency. Details of key metrics during each training session, including training accuracy, loss, and time; validation accuracy, loss, and time; testing accuracy, loss, and time; and precision, recall, and F1-score, are being captured.

After evaluating the models, the results are presented in Tables I, II, III, and IV. The optimal hyperparameter configurations, along with their corresponding performance metrics and the difference in performance made by Stratified Random Sampling, are summarized in Table V. Computational efficiency is determined by averaging the time spent on training and testing for each session. To ensure the robustness of the selected configurations, perform K-Fold cross-validation with both 5 and 10 folds, further validating their generalization capabilities. The results of these validations, along with their performance metrics, are included in Table V.

The differences in training and validation accuracy over epochs for each model's top-performing configurations are shown in Figs. 17, 18, 19, and 20. The differences in training and validation loss throughout epochs are also displayed in Figs. 21, 22, 23, and 24. The confusion matrices for each model are displayed in Figs. 25, 26, 27, and 28. For every model, we additionally examine the Receiver Operating Characteristic (ROC) curve and its Area Under the Curve (AUC) score. By graphing the true positive rate (TPR) versus the false positive rate (FPR) across various thresholds, the ROC curve illustrates a classifier's capacity to discriminate between classes [47]. Figs. 29, 30, 31, and 32 present the ROC-AUC curves for each model. In addition, Figs. 33, 34, 35, 36, and 37 plot the test accuracy versus each hyperparameter used for fine-tuning the models. Figs. 38, 39, 40, 41, 42, 43, 44, and 45 depict the cross-validation performances for 5 and 10 folds for each model.

To comprehensively assess the real-world performance of the models on unseen data, a test dataset of 50 samples was curated, including images from sources used during training, additional samples sourced online, and images captured by agricultural experts from Kerala Agricultural University. None of these images had been used in training, ensuring a valid evaluation, with each image meticulously verified by experts to confirm the specific banana disease it represented. The diversity of the test set, spanning multiple sources and real-world conditions, enabled a robust assessment of the models' ability to generalize beyond training data. Along with accuracy, the models are assessed using metrics related to resource usage monitored during the testing process, which include CPU, memory usage, and inference time. The findings, which are compiled in Table VI, provide a thorough understanding of the models' functionality. in both disease classification accuracy and computational efficiency, highlighting their practicality for diverse agricultural environments.

A. Analysis of Model Performance Metrics

All the models demonstrated high values across the performance metrics considered; however, distinct differences emerged between them. Upon analyzing the training results, ResNet50V2 achieved the highest accuracy at 99.52%, closely followed by MobileNet, which reached an impressive 99.47%, showing that it's nearly on par with ResNet50V2. InceptionV3 was also competitive, achieving 99.15% accuracy, while VGG16 trailed slightly with a still strong 98.88% accuracy. The results from 5-Fold and 10-Fold Cross Validation for all models matched the training accuracy, indicating that each model performs well in generalizing to various data subsets. The Precision, Recall, and F1-score closely matched the training accuracies, indicating balanced performance across all metrics. This consistency suggests the models accurately classified diseases while maintaining low false-positive and false-negative rates, as reflected in their confusion matrices.

When tested on the inference dataset, the models showed notable differences in performance. The best results were obtained by MobileNet, which had 90.00% accuracy, 91.50% precision, 90.00% recall, and a 90.38% F1-score, indicating its strong balance and suitability for real-time disease detection with minimal errors. ResNet50V2 followed closely with an accuracy of 86.00%, precision of 86.48%, recall of 86.00%, and F1-score of 84.90%, showcasing a slightly higher chance of missed detections.

InceptionV3 and VGG16 showed moderate results, with InceptionV3 achieving 82.00% accuracy and maintaining a reasonable balance across metrics, whereas VGG16 demonstrated 76.00% accuracy and higher precision but lower recall, suggesting it may miss some true positives.

B. Analysis of Model Efficiency

Among the models tested, MobileNet stood out for its significantly lower training and testing times compared to VGG16, ResNet50V2, and InceptionV3, despite sharing similar hyperparameters. During testing, MobileNet exhibited the lowest CPU usage, averaging 45.56%, which is advantageous for devices with limited processing power, as it ensures smooth performance without overloading the system. Additionally, MobileNet demonstrated the least memory consumption, with an average of just 799.59 MB during inference. This low memory footprint is particularly beneficial for mobile devices with constrained memory, enabling efficient real-time processing without straining resources.

MobileNet's ability to complete inference on a batch of 50 samples in approximately 4.7 seconds made it the fastest model in the test. Fast inference times are crucial for real-time applications, such as disease detection in agricultural settings, where quick results are essential for timely decision-making. Overall, MobileNet's combination of speed, low resource usage, and efficiency makes it an ideal choice for real-time, resource-constrained environments.

C. Summary of Model Evaluation

MobileNet was the most effective model in terms of computational performance and resource usage. Its lightweight architecture, optimized for mobile deployment [45], delivered the highest accuracy, precision, recall, and F1 score among the models tested. The model's minimal CPU and memory usage, coupled with its fast inference time, make it particularly well-suited for real-time disease detection on mobile devices, where both speed and resource efficiency are

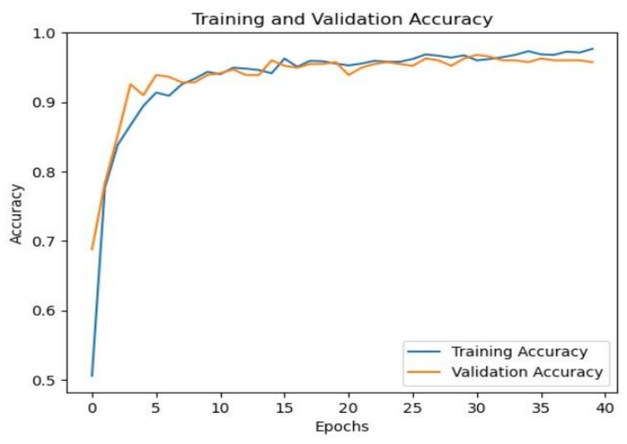


Fig. 17. VGG16 Training and Validation Accuracy Plot

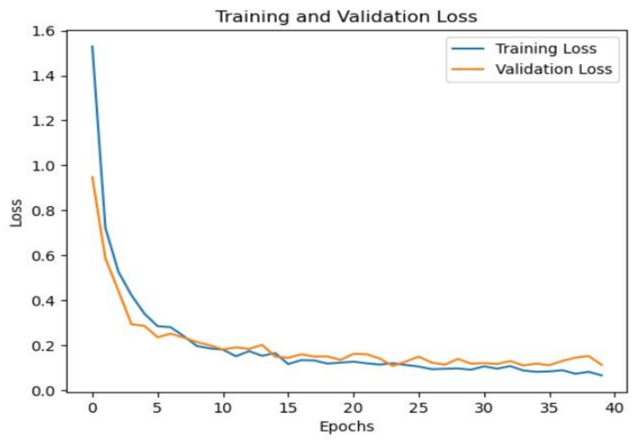


Fig. 21. VGG16 Training and Validation Loss Plot

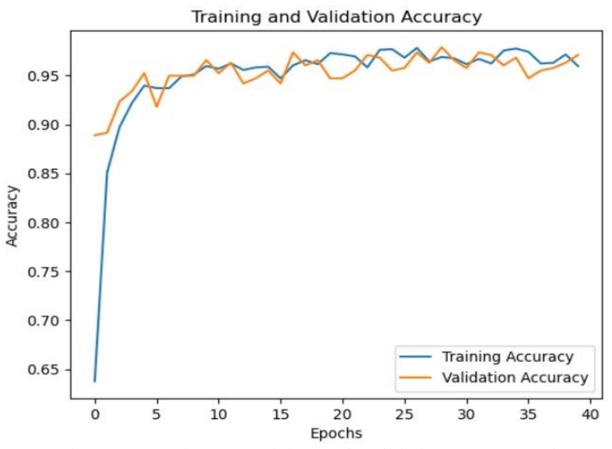


Fig. 18. InceptionV3 Training and Validation Accuracy Plot

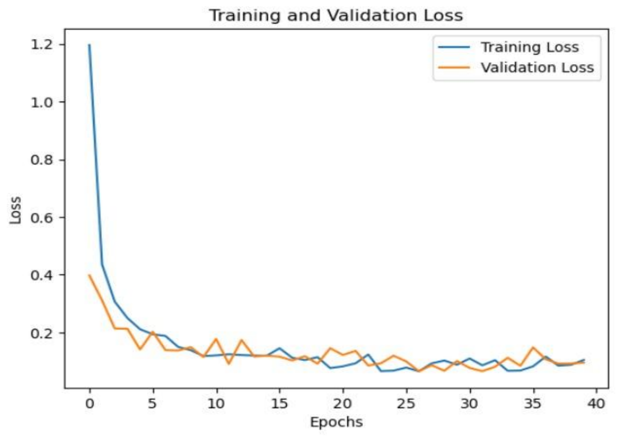


Fig. 22. InceptionV3 Training and Validation Loss Plot

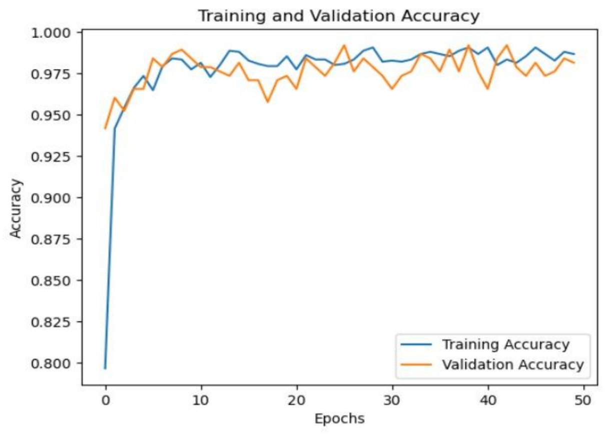


Fig. 19. ResNet50V2 Training and Validation Accuracy Plot

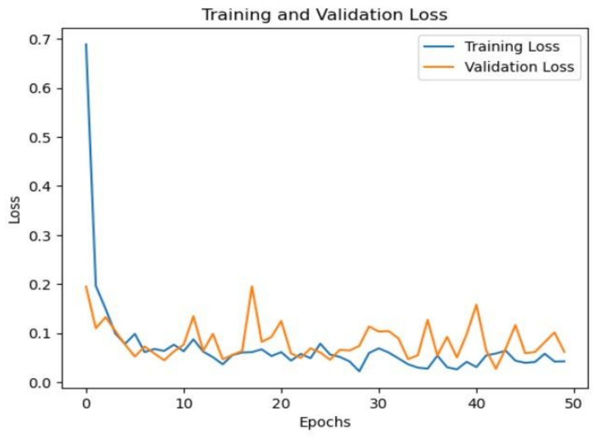


Fig. 23. ResNet50V2 Training and Validation Loss Plot

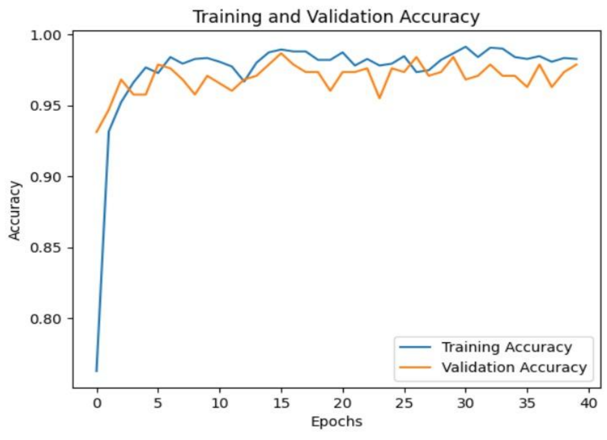


Fig. 20. MobileNet Training and Validation Accuracy Plot

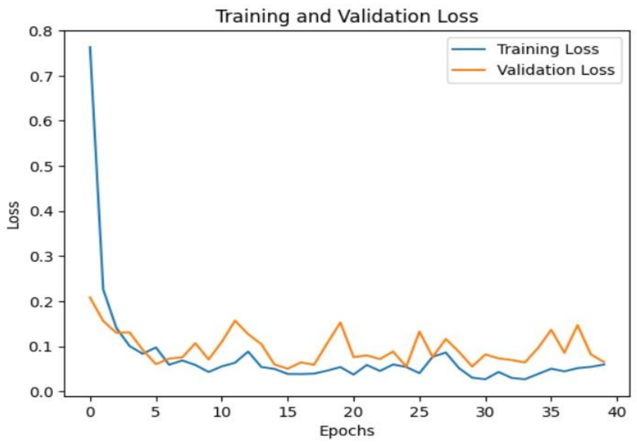


Fig. 24. MobileNet Training and Validation Loss Plot

TABLE II
PERFORMANCE OF VGG16

	Training Accuracy	Validation Accuracy	Testing Accuracy	Testing Loss	Precision	Recall	F1-Score
No. of Epochs:							
10	0.9397	0.9418	0.9629	0.1566	0.9644	0.9629	0.9630
20	0.9675	0.9577	0.9766	0.0829	0.9779	0.9766	0.9768
30	0.9728	0.9577	0.9856	0.0497	0.9864	0.9856	0.9857
40	0.9775	0.9709	0.9873	0.0432	0.9876	0.9872	0.9873
50	0.9808	0.9550	0.9856	0.0408	0.9864	0.9856	0.9857
Learning Rate:							
0.001	0.9775	0.9709	0.9873	0.0432	0.9876	0.9872	0.9873
0.0025	0.9695	0.9709	0.9809	0.0500	0.9813	0.9809	0.9809
0.005	0.9788	0.9762	0.9851	0.0455	0.9856	0.9851	0.9852
0.0075	0.9795	0.9709	0.9830	0.0496	0.9839	0.9830	0.9831
0.01	0.9682	0.9630	0.9793	0.0648	0.9801	0.9793	0.9793
No. of units in FC Layer:							
256	0.9708	0.9524	0.9772	0.0720	0.9790	0.9772	0.9773
512	0.9775	0.9709	0.9873	0.0455	0.9876	0.9872	0.9873
1024	0.9775	0.9656	0.9841	0.0550	0.9850	0.9841	0.9841
2048	0.9848	0.9524	0.9815	0.0624	0.9831	0.9814	0.9817
4096	0.9781	0.9577	0.9862	0.0477	0.9864	0.9862	0.9862
Batch Size:							
4	0.9755	0.9683	0.9888	0.0388	0.9889	0.9888	0.9888
8	0.9675	0.9497	0.9799	0.0599	0.9812	0.9798	0.9799
16	0.9775	0.9709	0.9873	0.0455	0.9876	0.9872	0.9873
32	0.9708	0.9577	0.9809	0.0693	0.9817	0.9809	0.9809
64	0.9649	0.9497	0.9777	0.0916	0.9782	0.9777	0.9777
Optimizers:							
Adam	0.9775	0.9709	0.9873	0.0455	0.9876	0.9872	0.9873
Nadam	0.9642	0.9550	0.9851	0.0449	0.9855	0.9851	0.9852
SGD	0.8270	0.8439	0.8733	0.5556	0.8881	0.8733	0.8739
RMSProp	0.9655	0.9392	0.9693	0.0884	0.9725	0.9692	0.9689
Adadelta	0.9609	0.9577	0.9740	0.0963	0.9760	0.9740	0.9743

TABLE III
PERFORMANCE OF INCEPTIONV3

	Training Accuracy	Validation Accuracy	Testing Accuracy	Testing Loss	Precision	Recall	F1-Score
No. of Epochs:							
10	0.9470	0.9418	0.9756	0.0904	0.9768	0.9756	0.9757
20	0.9556	0.9497	0.9793	0.0595	0.9804	0.9793	0.9794
30	0.9596	0.9577	0.9852	0.0426	0.9857	0.9851	0.9851
40	0.9728	0.9683	0.9915	0.0281	0.9916	0.9915	0.9915
50	0.9781	0.9444	0.9867	0.0372	0.9867	0.9862	0.9862
Learning Rate:							
0.001	0.9728	0.9683	0.9915	0.0281	0.9916	0.9915	0.9915
0.0025	0.9476	0.9550	0.9852	0.0397	0.9858	0.9851	0.9852
0.005	0.8330	0.9497	0.9751	0.0753	0.9755	0.9750	0.9751
0.0075	0.6428	0.9048	0.9300	0.2253	0.9351	0.9300	0.9299
0.01	0.4201	0.6640	0.6725	0.8429	0.6070	0.6724	0.6083
No. of units in FC Layer:							
256	0.9907	0.9683	0.9889	0.0252	0.9891	0.9888	0.9888
512	0.9728	0.9683	0.9915	0.0281	0.9916	0.9915	0.9915
1024	0.9748	0.9524	0.9714	0.0683	0.9728	0.9713	0.9714
2048	0.9622	0.9524	0.9772	0.0576	0.9781	0.9772	0.9772
4096	0.9772	0.9735	0.9803	0.0538	0.9815	0.9803	0.9801
Batch Size:							
4	0.9556	0.9683	0.9889	0.0252	0.9891	0.9888	0.9888
8	0.9728	0.9524	0.9894	0.0683	0.9895	0.9894	0.9894
16	0.9728	0.9683	0.9915	0.0281	0.9916	0.9915	0.9915
32	0.9722	0.9524	0.9772	0.0576	0.9781	0.9772	0.9772
64	0.9907	0.9735	0.9803	0.0538	0.9815	0.9803	0.9801
Optimizers:							
Adam	0.9728	0.9683	0.9915	0.0281	0.9916	0.9915	0.9915
Nadam	0.9636	0.9603	0.9868	0.0425	0.9875	0.9867	0.9868
SGD	0.9748	0.9630	0.9862	0.0417	0.9868	0.9862	0.9862
RMSProp	0.9649	0.9603	0.9883	0.0420	0.9887	0.9883	0.9883
Adadelta	0.9841	0.9709	0.9894	0.0293	0.9896	0.9894	0.9894

TABLE IV
PERFORMANCE OF RESNET50V2

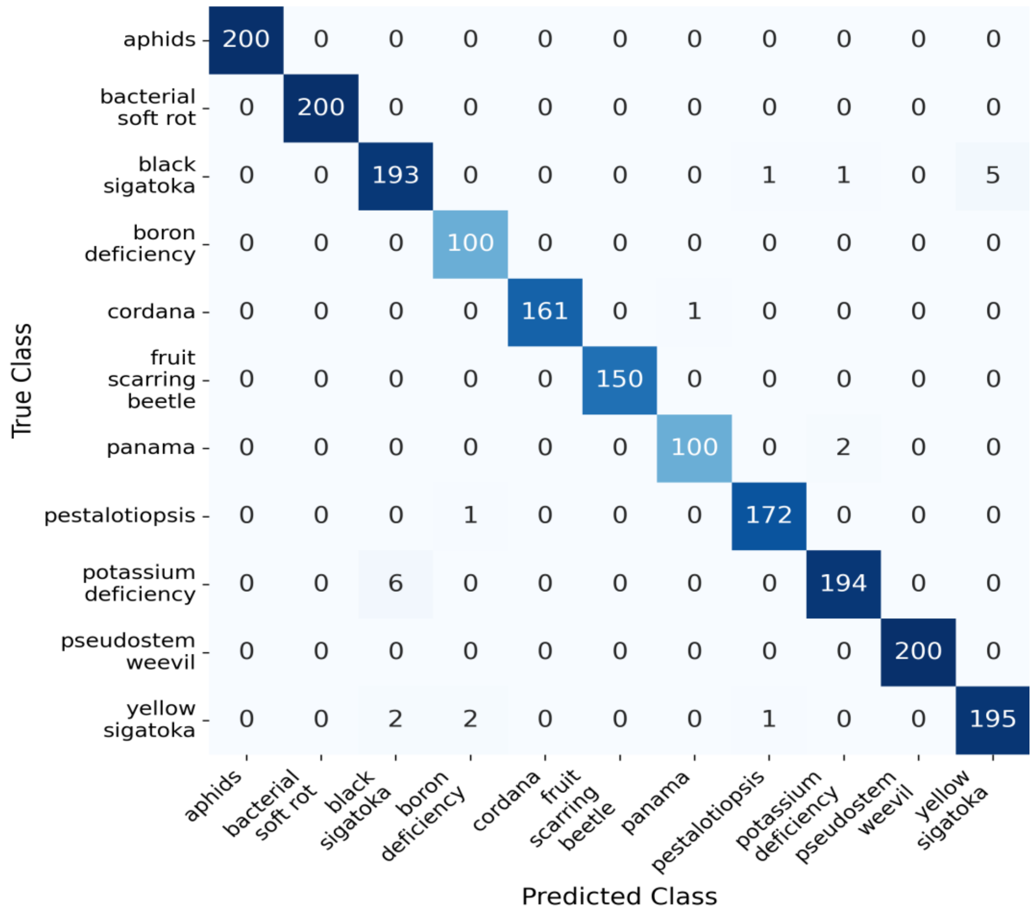
	Training Accuracy	Validation Accuracy	Testing Accuracy	Testing Loss	Precision	Recall	F1-Score
No. of Epochs:							
10	0.9775	0.9524	0.9746	0.0619	0.9768	0.9745	0.9748
20	0.9775	0.9735	0.9873	0.0334	0.9875	0.9872	0.9872
30	0.9874	0.9656	0.9841	0.0460	0.9858	0.9841	0.9843
40	0.9874	0.9815	0.9942	0.0159	0.9942	0.9941	0.9941
50	0.9887	0.9841	0.9952	0.0189	0.9952	0.9952	0.9952
Learning Rate:							
0.001	0.9887	0.9841	0.9952	0.0189	0.9952	0.9952	0.9952
0.0025	0.9828	0.9788	0.9889	0.0276	0.9897	0.9888	0.9888
0.005	0.9702	0.9656	0.9868	0.0510	0.9874	0.9867	0.9867
0.0075	0.9344	0.9524	0.9815	0.1000	0.9826	0.9814	0.9815
0.01	0.9165	0.9630	0.9815	0.0697	0.9815	0.9814	0.9814
No. of units in FC Layer:							
256	0.9887	0.9788	0.9931	0.0188	0.9932	0.9931	0.9931
512	0.9887	0.9841	0.9952	0.0189	0.9952	0.9952	0.9952
1024	0.9881	0.9841	0.9936	0.0190	0.9936	0.9936	0.9936
2048	0.9808	0.9788	0.9910	0.0800	0.9914	0.9909	0.9909
4096	0.9861	0.9815	0.9915	0.0776	0.9917	0.9915	0.9915
Batch Size:							
4	0.9874	0.9788	0.9905	0.0422	0.9906	0.9904	0.9904
8	0.9755	0.9815	0.9905	0.0307	0.9906	0.9904	0.9903
16	0.9887	0.9841	0.9952	0.0189	0.9952	0.9952	0.9952
32	0.9861	0.9735	0.9915	0.0233	0.9920	0.9915	0.9915
64	0.9947	0.9841	0.9926	0.0156	0.9927	0.9925	0.9925
Optimizers:							
Adam	0.9887	0.9841	0.9952	0.0189	0.9952	0.9952	0.9952
Nadam	0.9874	0.9788	0.9926	0.0248	0.9926	0.9925	0.9925
SGD	0.9914	0.9762	0.9931	0.0215	0.9932	0.9931	0.9931
RMSProp	0.9894	0.9868	0.9947	0.0299	0.9948	0.9947	0.9947
Adadelta	0.9934	0.9815	0.9936	0.0201	0.9937	0.9936	0.9936

TABLE V
PERFORMANCE OF MOBILENET

	Training Accuracy	Validation Accuracy	Testing Accuracy	Testing Loss	Precision	Recall	F1-Score
No. of Epochs:							
10	0.9867	0.9656	0.9889	0.0319	0.9897	0.9888	0.9888
20	0.9742	0.9762	0.9873	0.0278	0.9877	0.9872	0.9872
30	0.9894	0.9894	0.9905	0.0295	0.9909	0.9904	0.9904
40	0.9937	0.9868	0.9947	0.0101	0.9948	0.9947	0.9947
50	0.9887	0.9815	0.9926	0.0184	0.9926	0.9925	0.9925
Learning Rate:							
0.001	0.9937	0.9868	0.9947	0.0101	0.9948	0.9947	0.9947
0.0025	0.9689	0.9683	0.9883	0.0271	0.9891	0.9883	0.9884
0.005	0.9583	0.9683	0.9889	0.0273	0.9893	0.9888	0.9888
0.0075	0.9185	0.9656	0.9830	0.0477	0.9837	0.9830	0.9831
0.01	0.8615	0.9444	0.9815	0.0712	0.9820	0.9814	0.9815
No. of units in FC Layer:							
256	0.9867	0.9683	0.9915	0.0175	0.9917	0.9915	0.9915
512	0.9937	0.9868	0.9947	0.0101	0.9948	0.9947	0.9947
1024	0.9861	0.9841	0.9936	0.0209	0.9936	0.9936	0.9936
2048	0.9728	0.9735	0.9921	0.0194	0.9923	0.9920	0.9920
4096	0.9828	0.9709	0.9883	0.0598	0.9885	0.9883	0.9882
Batch Size:							
4	0.9861	0.9815	0.9905	0.0473	0.9908	0.9904	0.9904
8	0.9874	0.9735	0.9894	0.0267	0.9894	0.9894	0.9893
16	0.9937	0.9868	0.9947	0.0101	0.9948	0.9947	0.9947
32	0.9881	0.9815	0.9905	0.0203	0.9910	0.9904	0.9905
64	0.9914	0.9868	0.9931	0.0193	0.9931	0.9931	0.9931
Optimizers:							
Adam	0.9937	0.9868	0.9947	0.0101	0.9948	0.9947	0.9947
Nadam	0.9821	0.9762	0.9921	0.0284	0.9922	0.9920	0.9920
SGD	0.9887	0.9735	0.9894	0.0281	0.9899	0.9894	0.9894
RMSProp	0.9867	0.9735	0.9894	0.0503	0.9901	0.9894	0.9894
Adadelta	0.9934	0.9841	0.9878	0.0412	0.9887	0.9878	0.9877

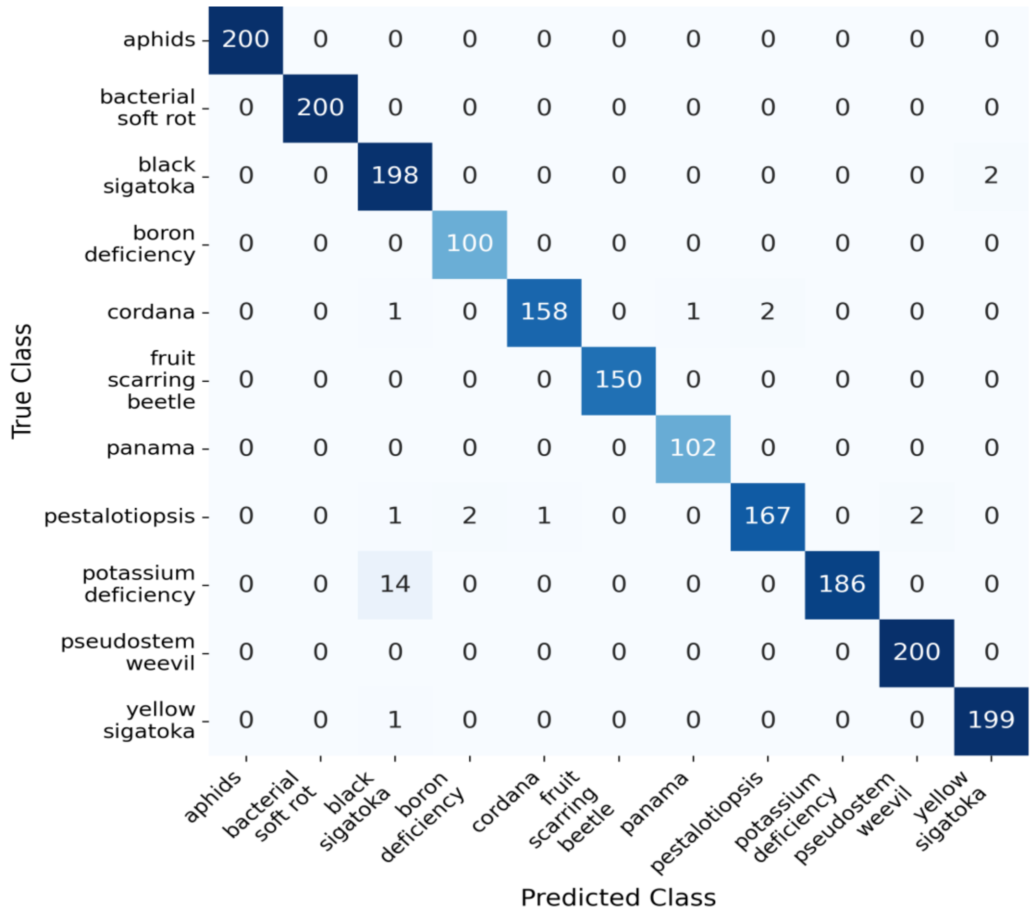
TABLE VI
COMPARISON OF THE MODELS

	VGG16	InceptionV3	Resnet50V2	MobileNet
Best Hyperparameter configuration:				
- Epochs:	40	40	50	40
- Optimizer:	Adam	Adam	Adam	Adam
- Learning Rate:	0.001	0.001	0.001	0.001
- Batch Size:	4	16	16	16
- No. of units in the FC layer:	512	512	512	512
Best performance corresponding to the best Hyperparameter configuration without Stratified Random Sampling:				
- Accuracy	0.8590	0.8484	0.8641	0.8505
- Precision	0.8591	0.8484	0.8643	0.8506
- Recall	0.8590	0.8484	0.8641	0.8505
- F1-Score	0.8590	0.8484	0.8641	0.8505
Best performance corresponding to the best Hyperparameter configuration with Stratified Random Sampling:				
- Accuracy	0.9888	0.9915	0.9952	0.9947
- Precision	0.9889	0.9916	0.9952	0.9948
- Recall	0.9888	0.9915	0.9952	0.9947
- F1-Score	0.9888	0.9915	0.9952	0.9947
Average time elapsed:				
- Training Time	5840.65 s	1935.46 s	2647.29 s	1309.79 s
- Testing Time	185.64 s	41.56 s	59.78 s	22.81 s
5-Fold Cross-Validation Performance:				
- Average Accuracy	0.9889	0.9915	0.9952	0.9947
- Average Precision	0.9891	0.9918	0.9954	0.9948
- Average Recall	0.9888	0.9915	0.9952	0.9947
- Average F1-Score	0.9888	0.9915	0.9952	0.9947
- Average Prediction Time	64.29 s	48.39 s	50.47 s	32.01 s
10-Fold Cross-Validation Performance:				
- Average Accuracy	0.9888	0.9915	0.9952	0.9947
- Average Precision	0.9895	0.9922	0.9955	0.9949
- Average Recall	0.9888	0.9915	0.9952	0.9947
- Average F1-Score	0.9888	0.9915	0.9952	0.9947
- Average Prediction Time	30.74 s	22.96 s	26.47 s	15.91 s



True Class \ Predicted Class	aphids	bacterial soft rot	black sigatoka	boron deficiency	cordana	fruit scarring beetle	panama	pestalotiopsis	potassium deficiency	pseudostem weevil	yellow sigatoka
aphids	200	0	0	0	0	0	0	0	0	0	0
bacterial soft rot	0	200	0	0	0	0	0	0	0	0	0
black sigatoka	0	0	193	0	0	0	0	1	1	0	5
boron deficiency	0	0	0	100	0	0	0	0	0	0	0
cordana	0	0	0	0	161	0	1	0	0	0	0
fruit scarring beetle	0	0	0	0	0	150	0	0	0	0	0
panama	0	0	0	0	0	0	100	0	2	0	0
pestalotiopsis	0	0	0	1	0	0	0	172	0	0	0
potassium deficiency	0	0	6	0	0	0	0	0	194	0	0
pseudostem weevil	0	0	0	0	0	0	0	0	0	200	0
yellow sigatoka	0	0	2	2	0	0	0	1	0	0	195

Fig. 25. VGG16 Confusion Matrix



True Class \ Predicted Class	aphids	bacterial soft rot	black sigatoka	boron deficiency	cordana	fruit scarring beetle	panama	pestalotiopsis	potassium deficiency	pseudostem weevil	yellow sigatoka
aphids	200	0	0	0	0	0	0	0	0	0	0
bacterial soft rot	0	200	0	0	0	0	0	0	0	0	0
black sigatoka	0	0	198	0	0	0	0	0	0	0	2
boron deficiency	0	0	0	100	0	0	0	0	0	0	0
cordana	0	0	1	0	158	0	1	2	0	0	0
fruit scarring beetle	0	0	0	0	0	150	0	0	0	0	0
panama	0	0	0	0	0	0	102	0	0	0	0
pestalotiopsis	0	0	1	2	1	0	0	167	0	2	0
potassium deficiency	0	0	14	0	0	0	0	0	186	0	0
pseudostem weevil	0	0	0	0	0	0	0	0	0	200	0
yellow sigatoka	0	0	1	0	0	0	0	0	0	0	199

Fig. 26. InceptionV3 Confusion Matrix

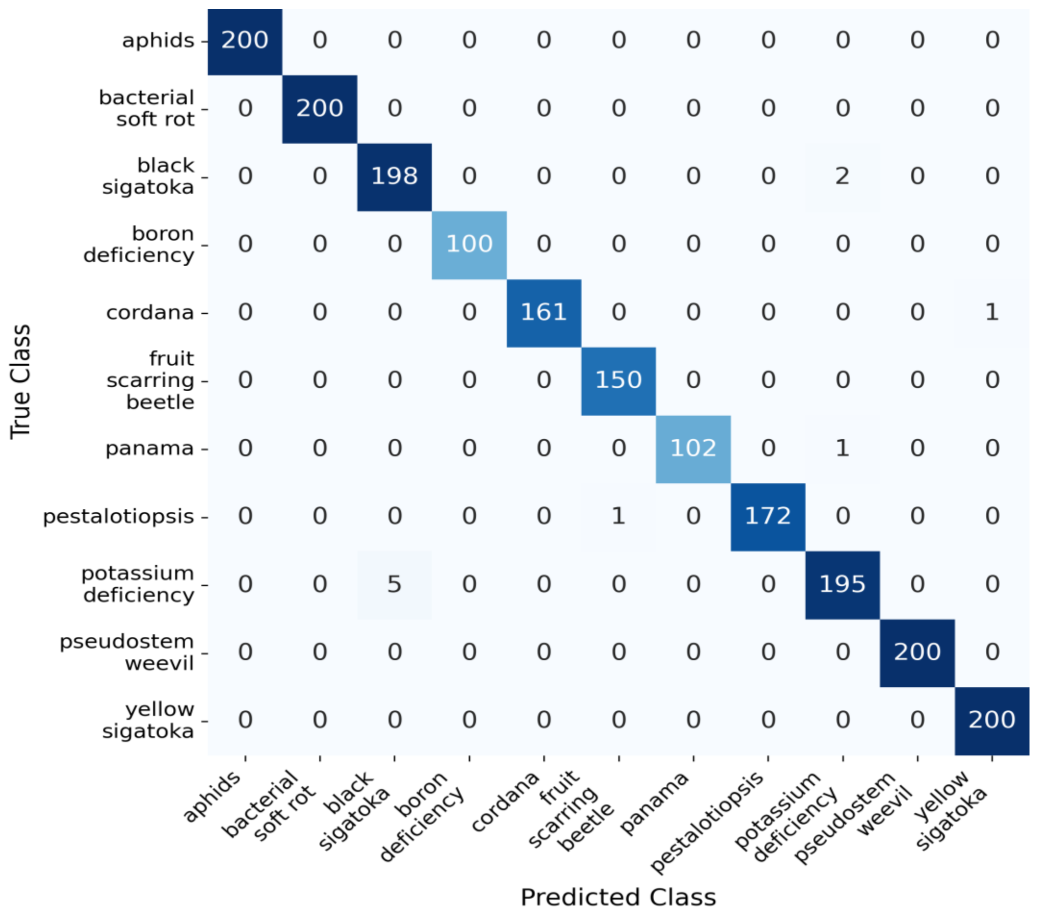


Fig. 27. ResNet50V2 Confusion Matrix

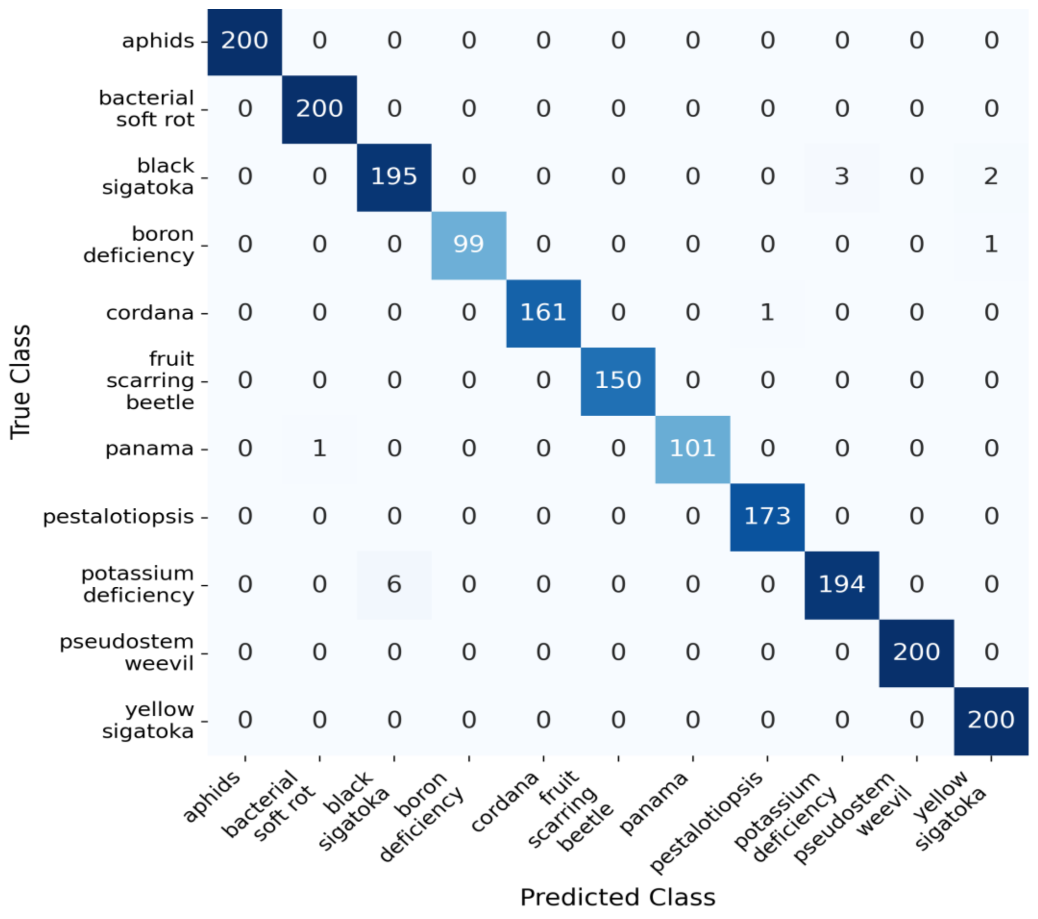


Fig. 28. MobileNet Confusion Matrix

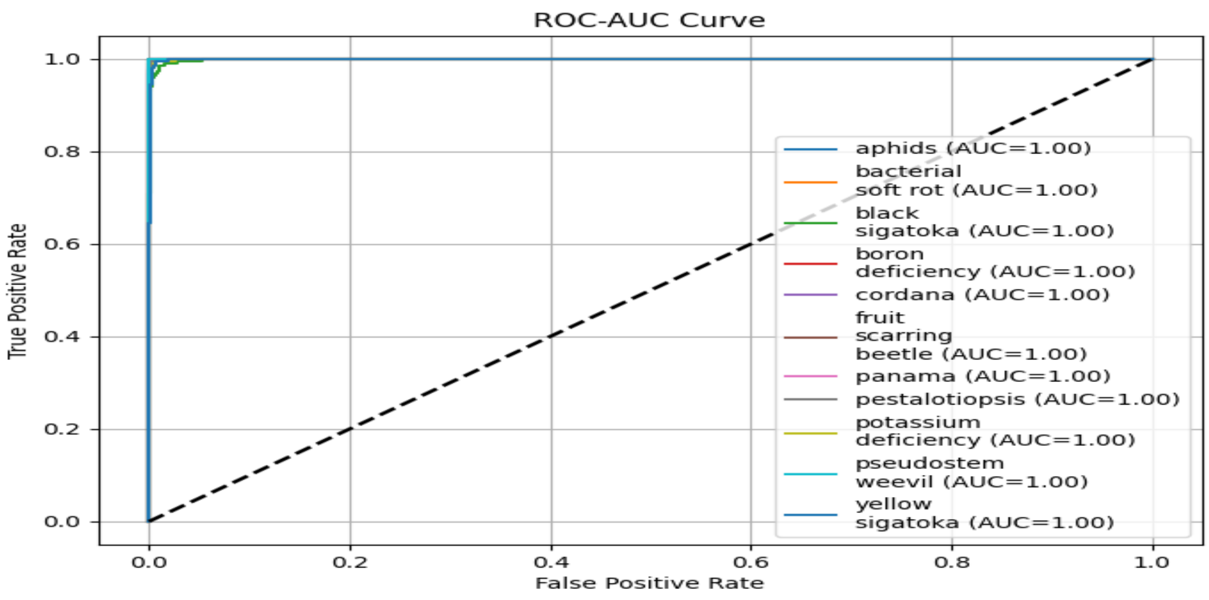


Fig. 29. VGG16 ROC-AUC Plot

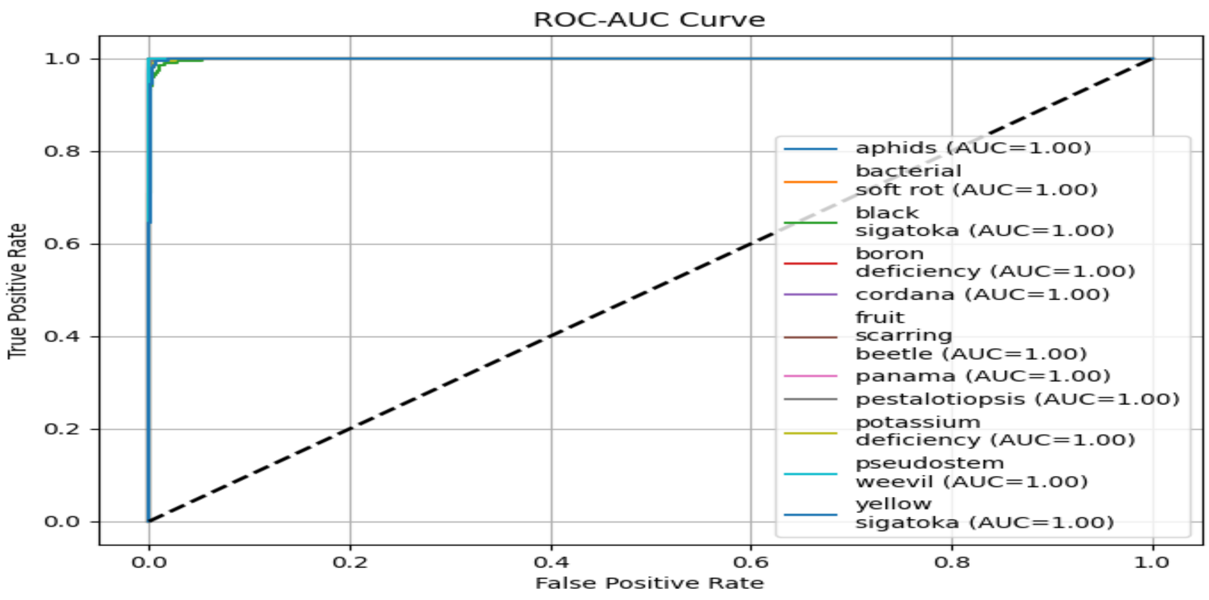


Fig. 30. InceptionV3 ROC-AUC Plot

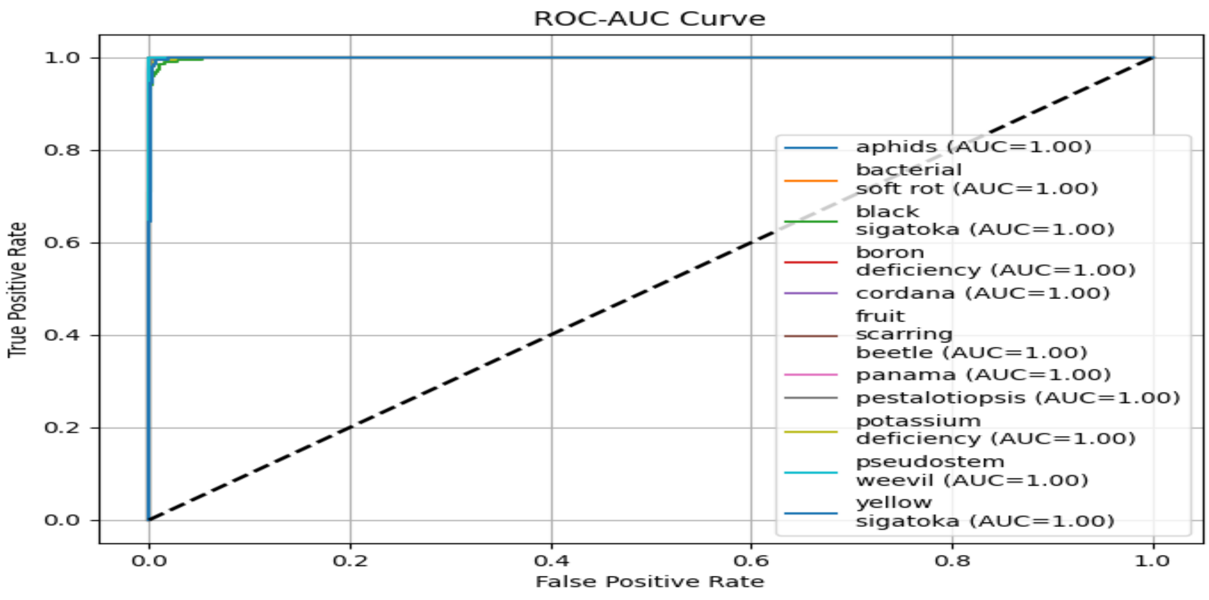


Fig. 31. ResNet50V2 ROC-AUC Plot

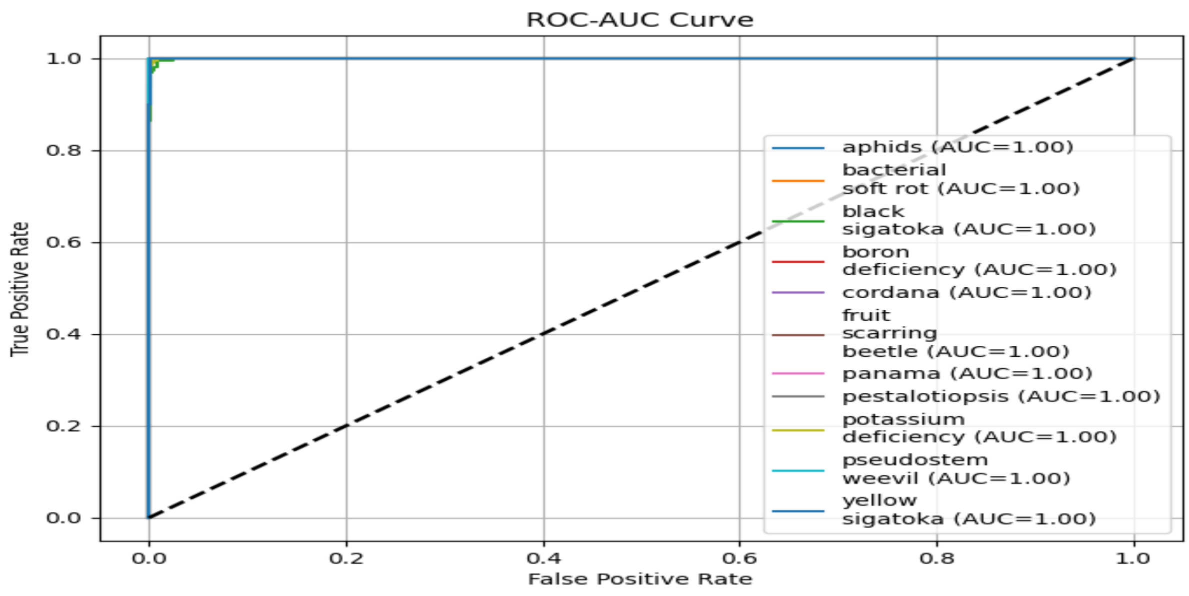


Fig. 32. MobileNet ROC-AUC Plot

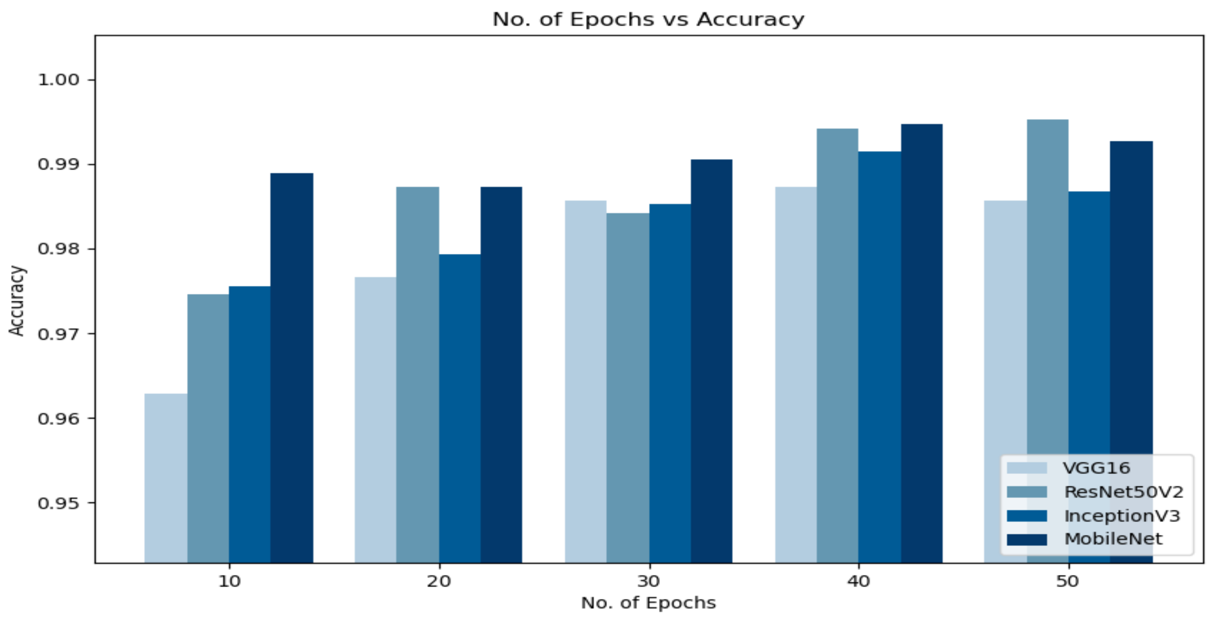


Fig. 33. Comparison of Model Accuracy for different No. of Epochs

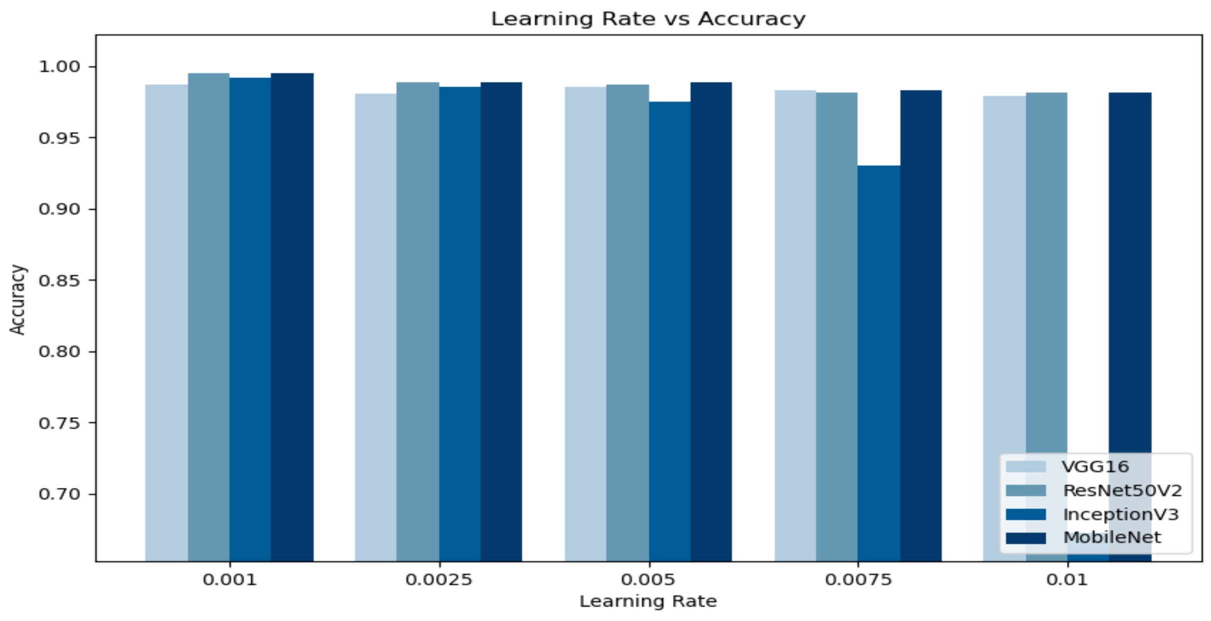


Fig. 34. Comparison of Model Accuracy for different Learning Rates

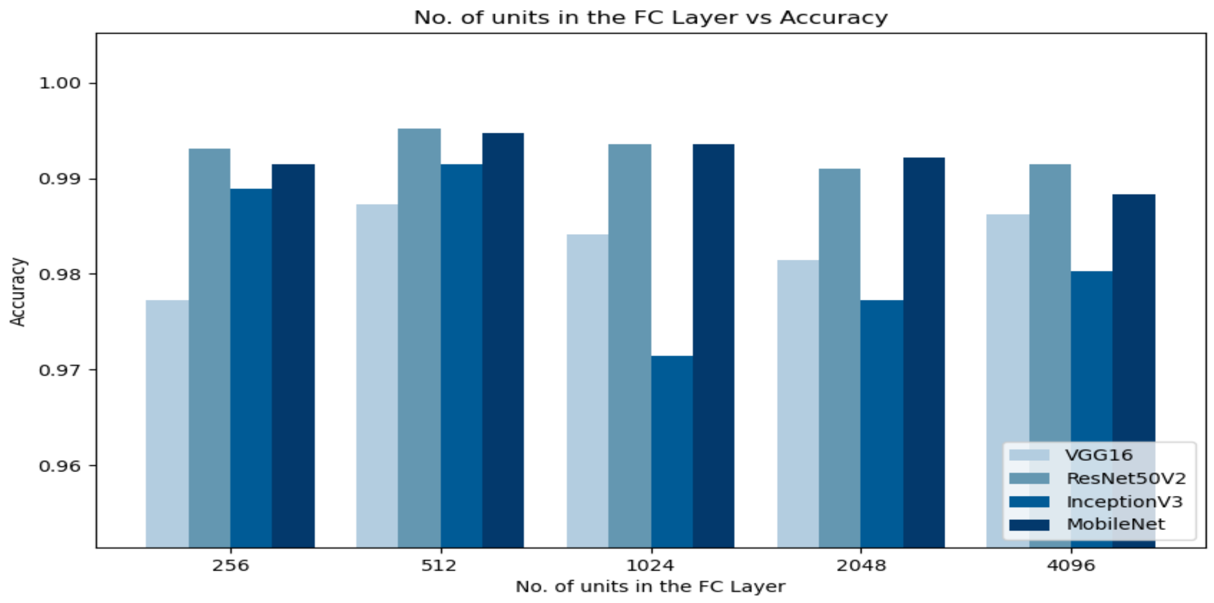


Fig. 35. Comparison of Model Accuracy for different units in FC Layer

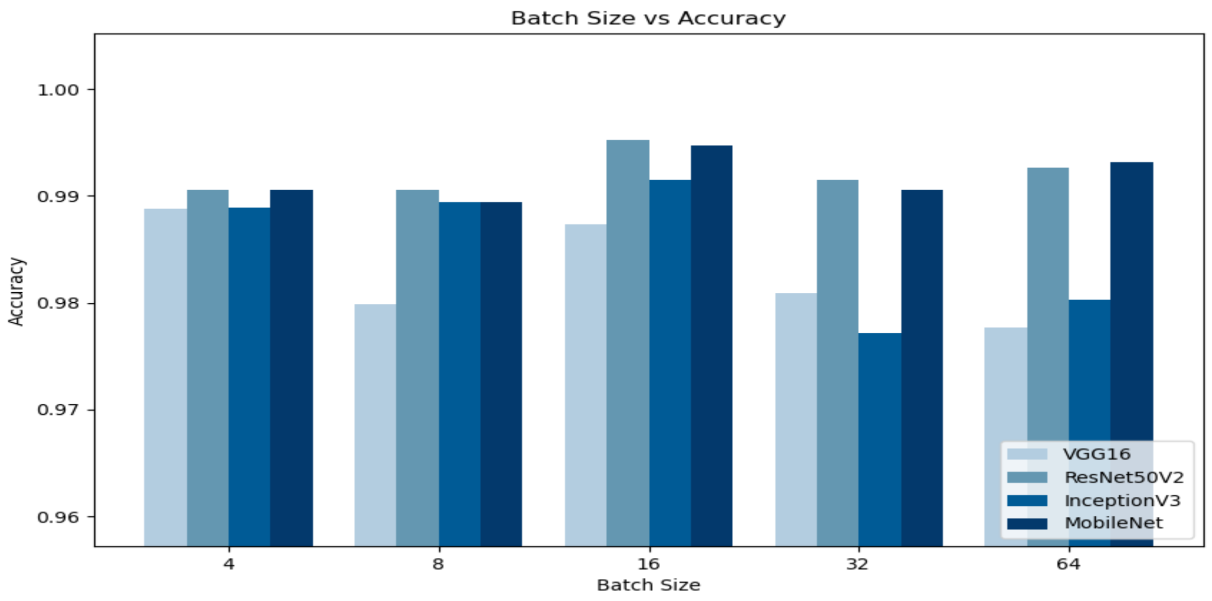


Fig. 36. Comparison of Model Accuracy for different Batch Sizes

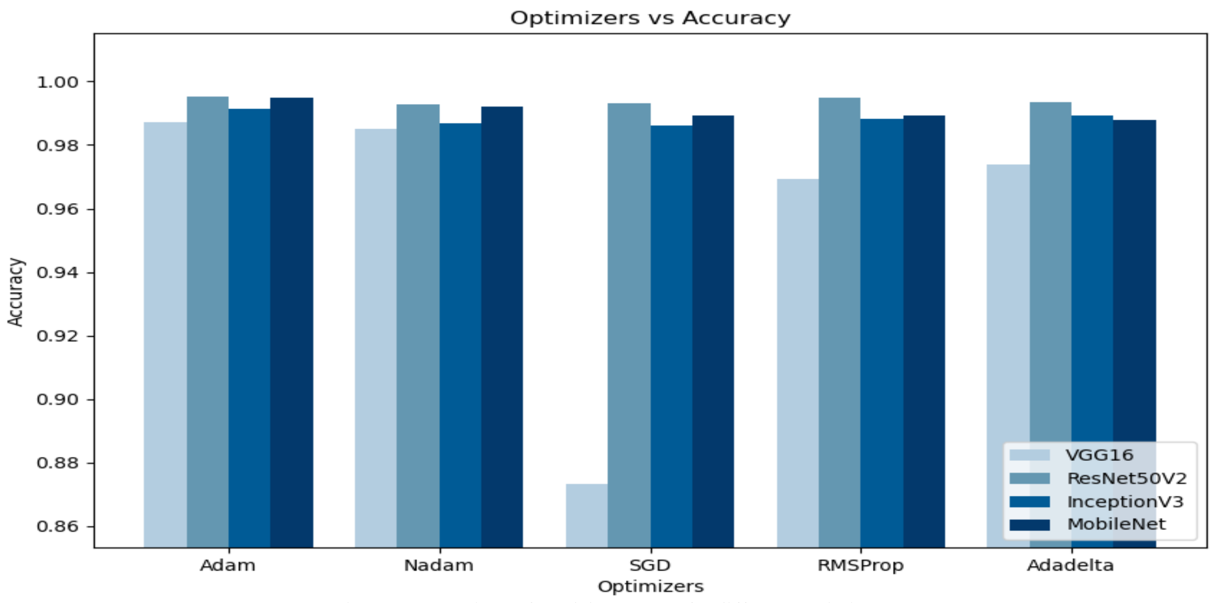


Fig. 37. Comparison of Model Accuracy for different Optimizers

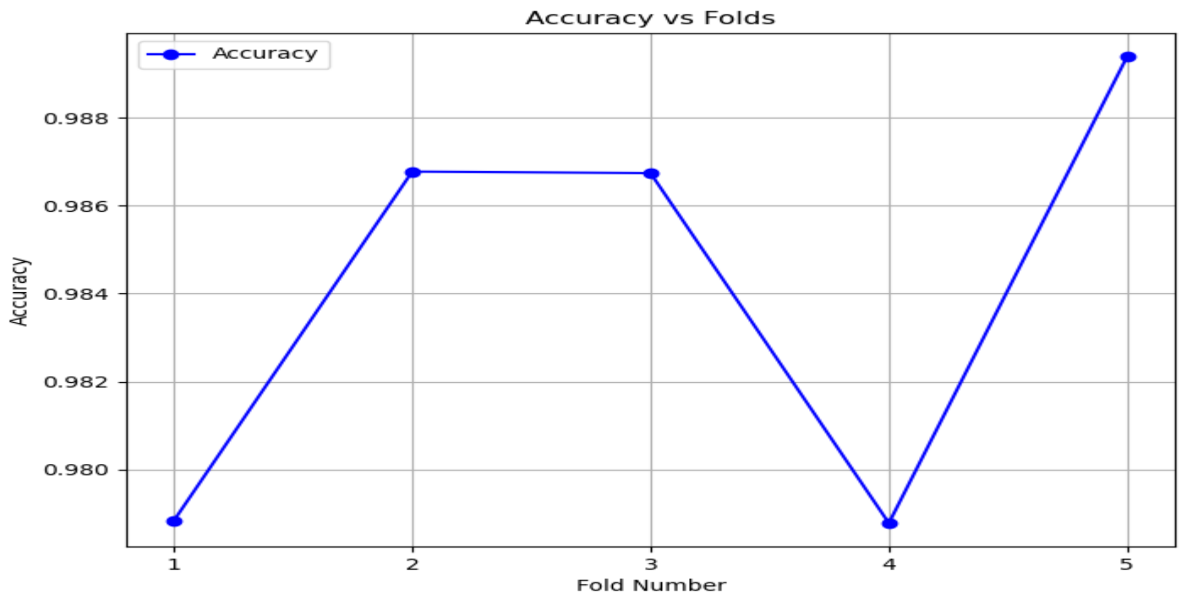


Fig. 38. VGG16 5-Fold Cross Validation Plot

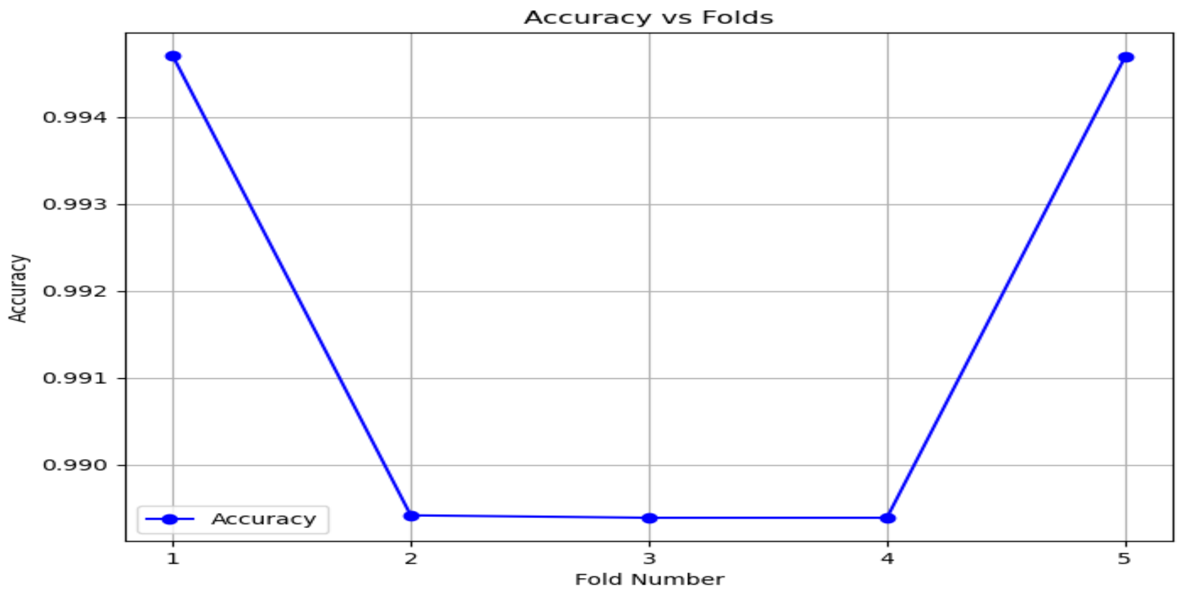


Fig. 39. InceptionV3 5-Fold Cross Validation Plot

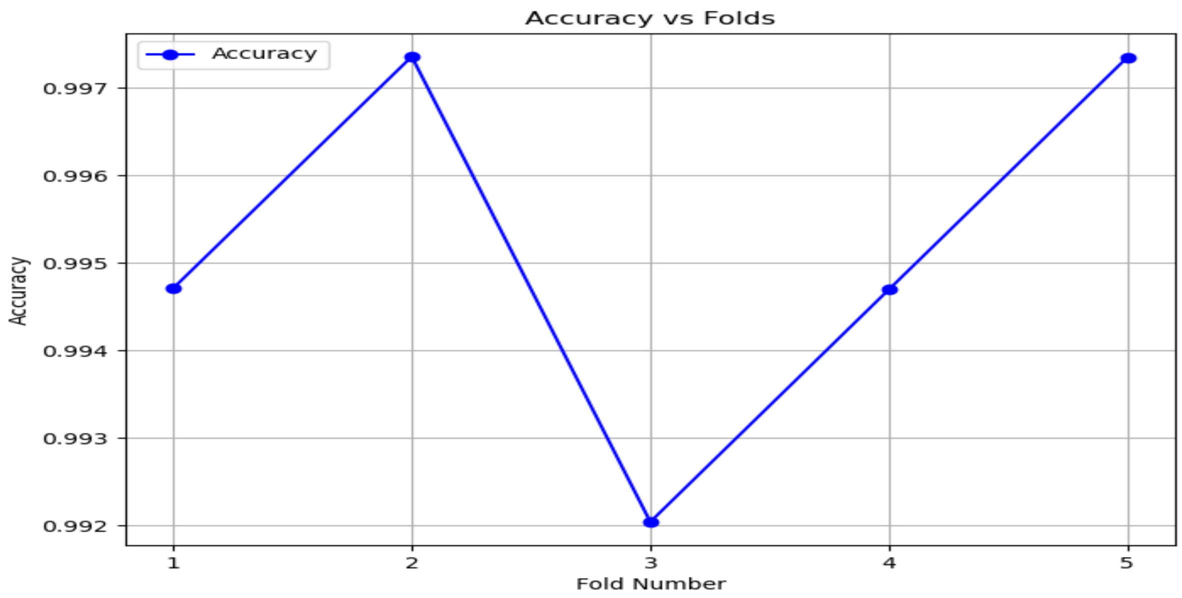


Fig. 40. ResNet50V2 5-Fold Cross Validation Plot

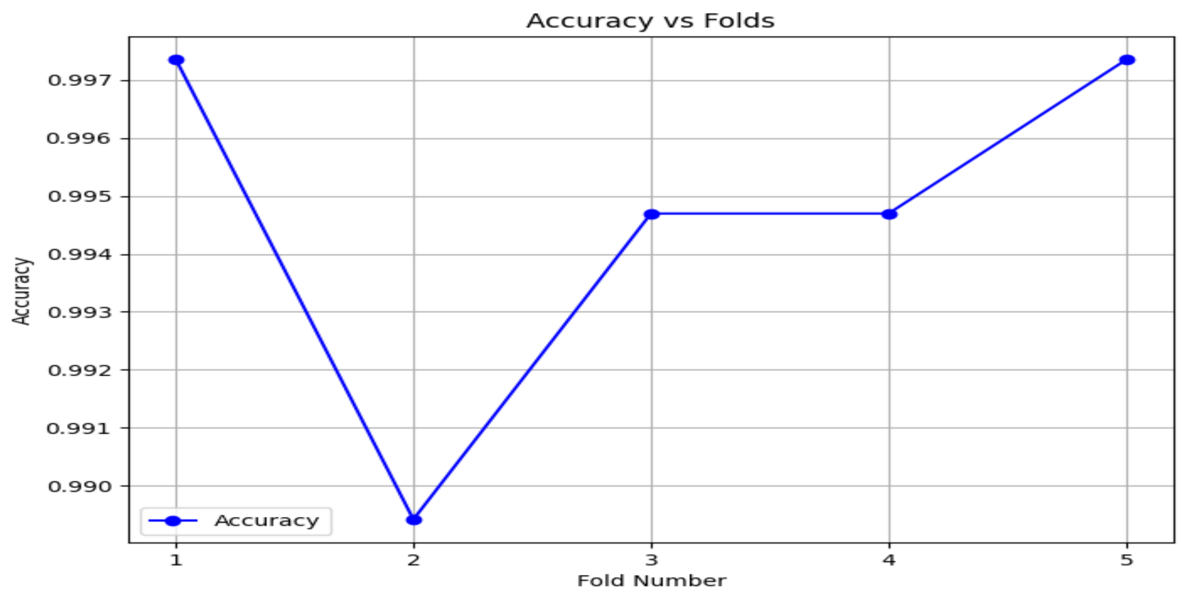


Fig. 41. MobileNet 5-Fold Cross Validation Plot

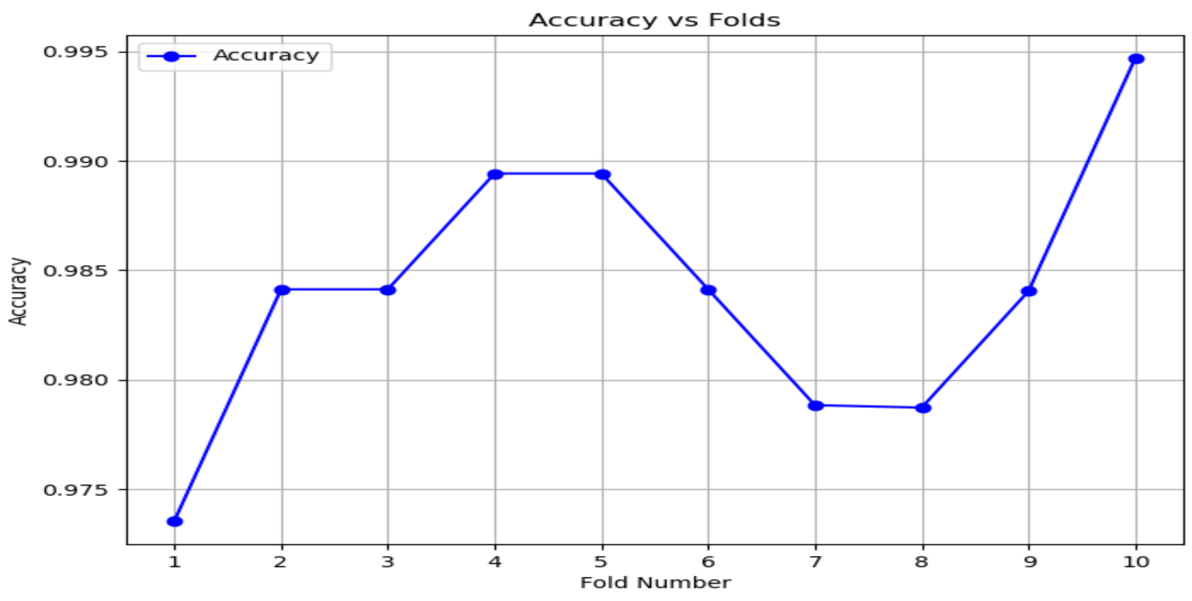


Fig. 42. VGG16 10-Fold Cross Validation Plot

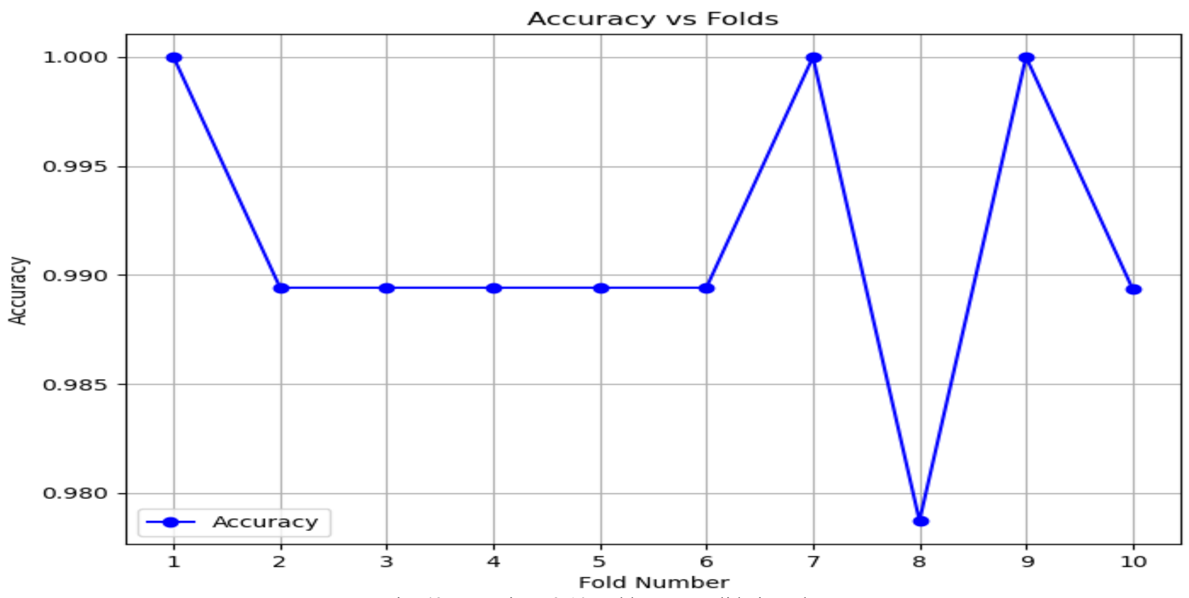


Fig. 43. InceptionV3 10-Fold Cross Validation Plot

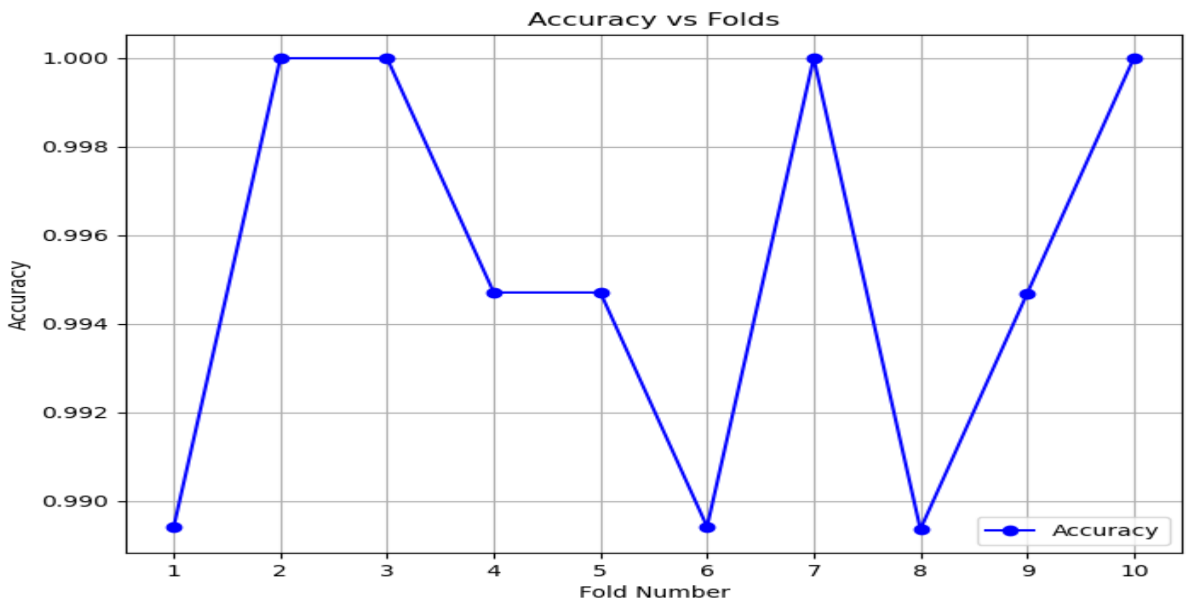


Fig. 44. ResNet50V2 10-Fold Cross Validation Plot

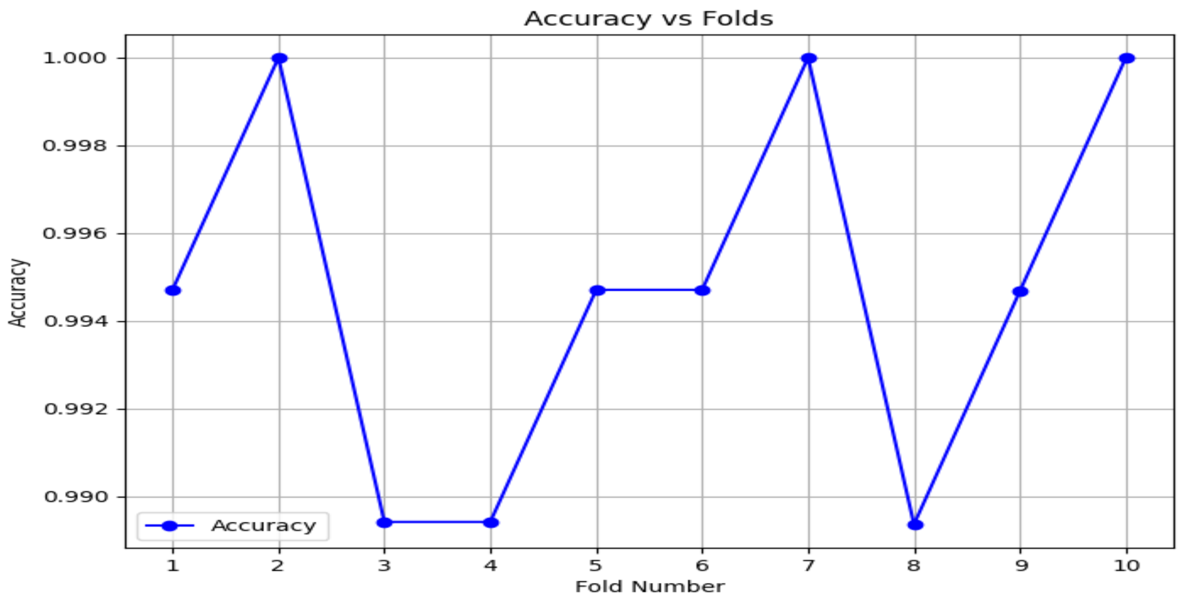


Fig. 45. MobileNet 10-Fold Cross Validation

critical. ResNet50V2 offered a solid balance of accuracy and efficiency, though it lagged behind MobileNet in terms of inference speed and memory consumption. While it was more resource-efficient than VGG16 and InceptionV3, it still proved heavier compared to the compact MobileNet.

InceptionV3 demonstrated improved efficiency over VGG16, especially in accuracy and recall, but remained resource-intensive, particularly in memory usage. Despite its reasonable performance, it was less lightweight and efficient than MobileNet, making it less suitable for environments with strict resource constraints. VGG16 [46], while delivering moderate accuracy, showed high resource consumption with significant CPU and memory utilization during inference. Its slower inference time further highlighted its limitations, particularly for real-time, mobile-based applications where fast results and low resource demands are essential.

V. Conclusion

The performance and efficiency evaluation of the four deep learning models—MobileNet, ResNet50V2, InceptionV3, and VGG16—has provided valuable insights into their suitability for real-time disease detection in resource-constrained environments. Among the models tested, MobileNet emerged as the top performer, achieving the highest balance of accuracy, precision, recall, and F1-score, combined with exceptional efficiency in terms of memory consumption, CPU usage, and inference speed. With an accuracy of 90.00% on an unknown sample dataset and a rapid inference time of approximately 4.7 seconds for 50 samples, MobileNet is particularly well-suited for deployment on mobile devices and in agricultural settings, where timely disease detection is critical.

In contrast, while ResNet50V2 offered competitive accuracy, it lagged behind MobileNet in terms of resource

TABLE VII
PERFORMANCE OF MODELS FOR UNKNOWN DATA

Metric	VGG16	InceptionV3	ResNet50V2	MobileNet
Accuracy	0.7600	0.8200	0.8600	0.9000
Precision	0.8146	0.8550	0.8648	0.9150
Recall	0.7600	0.8200	0.8600	0.9000
F1-Score	0.7679	0.8256	0.8490	0.9038
Time taken to predict all 50 samples	8.8695 s	8.3103 s	8.3274 s	4.7034 s
Average CPU Utilization	62.54 %	52.28%	54.71%	45.56 %
Average Memory Utilization	1013.57 MB	877.15 MB	882.13 MB	799.59 MB

efficiency and inference speed. Despite having respectable accuracy and recall, InceptionV3 was less suitable for real-time applications due to its increased resource usage and lengthier inference durations. VGG16, despite its higher precision, demonstrated slower inference times and significant resource usage, highlighting its limitations in mobile and resource-limited environments.

All things considered, MobileNet is the finest model for real-time disease detection because of its high accuracy and low resource usage, which makes it the ideal choice for mobile deployment where speed and efficiency are crucial. This capability enables the diagnosis of diseases in agricultural areas in real time, particularly in rural locations with low computational resources.

The ROC-AUC analysis further validates the classification strength of all four models, each achieving an AUC of 1.0, indicating perfect discrimination between banana diseases across all thresholds on the current dataset. However, such high performance may point to potential overfitting or dataset biases, highlighting the importance of testing under real-world conditions. Variations in lighting, leaf texture, and unseen disease patterns could affect model generalization. Tools like partial AUC (pAUC) are therefore essential, focusing evaluation on meaningful False Positive Rate (FPR) or True Positive Rate (TPR) ranges aligned with real-life decision-making. While a high true positive rate enables early and effective disease management, excessive false positives can lead to unnecessary treatments and increased costs.

Future development of disease detection systems can focus on optimizing accessibility through a web-based platform, enabling users to access the model on any internet-enabled device without the need for installation or high-end hardware. This would reduce hardware dependency and significantly broaden the user base, particularly in rural and resource-limited areas. Furthermore, more research into lightweight models and sophisticated optimization methods, like knowledge distillation, pruning, and model quantization, can further cut down on resource usage without sacrificing accuracy. These methods would enhance the model's suitability for deployment on mobile devices with limited computational capabilities.

In parallel, incorporating adaptive learning techniques and periodic model updates based on new data can help maintain high detection accuracy over time. This approach could address potential shifts in disease patterns or new environmental conditions, ensuring that the model remains relevant and effective. By focusing on these improvements, the disease detection system can be made more efficient, scalable, and accessible, offering a sustainable solution for timely and accurate disease detection in agricultural settings worldwide, empowering farmers and agriculturists to improve crop health and yield.

REFERENCES

[1] International Food Policy Research Institute (IFPRI). "Agriculture: Key to Economic Transformation, Food Security, and Nutrition." IFPRI Blog, February 8, 2018, https://www.ifpri.org/blog/agriculture-key-economic-transformation-food-security-and-nutrition/.

[2] "Issues Paper: How to Feed the World in 2050." Food and Agriculture Organization of the United Nations (FAO), October 12, 2009, https://www.fao.org/fileadmin/templates/wsfs/docs/Issues_papers/HLEF2050_Global_Agriculture.pdf.

[3] Ristaino, Jean B., Pamela K. Anderson, Daniel P. Bebber, Kate A. Brauman, Nik J. Cunniffe, Nina V. Fedoroff, Cambria Finegold et al. "The persistent threat of emerging plant disease pandemics to global food security." Proceedings of the National Academy of Sciences, vol. 118, no. 23, p. e2022239118, 2021.

[4] Wang, Haiqing, Shuqi Shang, Dongwei Wang, Xiaoning He, Kai Feng, and Hao Zhu. "Plant disease detection and classification method based on the optimized lightweight YOLOv5 model." Agriculture, vol. 12, no. 7, p. 931, 2022.

[5] American Phytopathological Society. "Plant Disease Diagnosis." APSnet Education Center, Accessed June 16, 2024.

[6] Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587. 2014.

[7] Bhatia, Gresha S., Pankaj Ahuja, Devendra Chaudhari, Sanket Paratkar, and Akshaya Patil. "Plant disease detection using deep learning." In Second International Conference on Computer Networks and Communication Technologies: ICCNCT 2019, pp. 408-415. Springer International Publishing, 2020.

[8] ClanX AI. "Convolutional Neural Networks (CNNs)." ClanX Glossary, January 26, 2024, https://clanx.ai/glossary/convolutional-neural-networks-cnns.

[9] Chowdhury, Muhammad EH, Tawsifur Rahman, Amith Khandakar, Mohamed Arselene Ayari, Aftab Ullah Khan, Muhammad Salman Khan, Nasser Al-Emadi, Mamun Bin Ibne Reaz, Mohammad Tariqul Islam, and Sawal Hamid Md Ali. "Automatic and reliable leaf disease detection using deep learning techniques." AgriEngineering, vol. 3, no. 2, pp. 294-312, 2021.

[10] Ferentinos, Konstantinos P. "Deep learning models for plant disease detection and diagnosis." Computers and electronics in agriculture, vol. 145, pp. 311-318, 2018.

[11] Brahimi, Mohammed, Marko Arsenovic, Sohaib Laraba, Srdjan Sladojevic, Kamel Boukhalfa, and Abdelouhab Moussaoui. "Deep learning for plant diseases: detection and saliency map visualisation." Human and machine learning: Visible, explainable, trustworthy and transparent, pp. 93-117, 2018.

[12] A. Picon, P. Alvarez-Gila, A. Seitz, J. Ortiz-Barredo, and A. Echazarra, "Deep Learning for Plant Disease Detection Using Convolutional Neural Networks." AI, vol. 3, no. 2, pp. 231-248, 2020.

[13] Athiraja, A., and P. Vijayakumar. "Retracted article: Banana disease diagnosis using computer vision and machine learning methods." Journal of Ambient Intelligence and Humanized Computing, vol. 12, no. 6, pp. 6537-6556, 2021.

[14] Sanga, Sophia, Victor Mero, Dina Machuve, and Davis Mwanganda. "Mobile-based deep learning models for banana diseases detection." arXiv preprint arXiv:2004.03718, 2020.

[15] Chowdhury, Muhammad EH, Tawsifur Rahman, Amith Khandakar, Mohamed Arselene Ayari, Aftab Ullah Khan, Muhammad Salman Khan, Nasser Al-Emadi, Mamun Bin Ibne Reaz, Mohammad Tariqul Islam, and Sawal Hamid Md Ali. "Automatic and reliable leaf disease detection using deep learning techniques." AgriEngineering, vol. 3, no. 2, pp. 294-312, 2021.

[16] Ahamed, Md Khabir Uddin, Md Manowarul Islam, Md Ashraf Uddin, Arnisha Akhter, Uzzal Kumar Acharjee, Bikash Kumar Paul, and Mohammad Ali Moni. "DTLCx: an improved ResNet architecture to classify normal and conventional pneumonia cases from COVID-19 instances with Grad-CAM-based superimposed visualization utilizing chest X-ray images." Diagnostics, vol. 13, no. 3, p. 551, 2023.

[17] Mascarenhas, Sheldon, and Mukul Agarwal. "A comparison between VGG16, VGG19 and ResNet50 architecture frameworks for Image Classification." In 2021 International conference on disruptive technologies for multi-disciplinary research and applications (CENTCON), vol. 1, pp. 96-99. IEEE, 2021.

[18] Ali, Luqman, Fady Alnajjar, Hamad Al Jassmi, Munkhjargal Gocho, Wasif Khan, and M. Adel Serhani. "Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures." Sensors, vol. 21, no. 5, p. 1688, 2021.

[19] Vaibhav Khandelwal. "The Architecture and Implementation of VGG-16." Towards AI, August 17, 2022, https://pub.towardsai.net/the-architecture-and-implementation-of-vgg-16-b050e5a5920b.

[20] Patel, Sitaram, and Nikhat Raza Khan. "A weighted-average-ensembling based hybrid CNN model for improved COVID-19 detection.", 2023.

[21] Lekhuy Nguyen. "An Overview of VGG16 and NiN Models." Medium, March 26, 2021, https://lekhuyen.medium.com/an-overview-of-vgg16-and-nin-models-96e4bf398484.

[22] Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818-2826. 2016.

[23] Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861, 2017.

[24] Deep Learning Bible. "2. Classification - Eng," Wikidocs.net, May 18, 2023, https://wikidocs.net/165429.

[25] Medhi, Epsita, and Nabamita Deb. "PSFD-Musa: A dataset of banana plant, stem, fruit, leaf, and disease." Data in brief 43, p. 108427, 2022.

[26] Foottit, R. G., H. E. L. Maw, K. S. Pike, and R. H. Miller. "The identity of Pentalonia nigronervosa Coquerel and P. caladii van der Goot (Hemiptera: Aphididae) based on molecular and morphometric analysis." Zootaxa 2358, no. 1, pp. 25-38, 2010.

[27] Arman, Shifat E., Md Abdullahil Baki Bhuiyan, Hasan Muhammad Abdullah, Shariful Islam, Tahsin Tanha Chowdhury, and Md Arban Hossain. "BananaLSD: A banana leaf images dataset for classification of banana leaf diseases using machine learning." Data in Brief, vol. 50, p. 109608, 2023.

[28] Sunitha, P. "Images of nutrient deficient banana plant leaves." Mendeley Data 1, 2022.

[29] "Deficiencies and Disorders-Banana-Boron," TNAU Agritech Portal, January, 2022, https://agritech.tnau.ac.in/horticulture/plant_nutri/banana_bor.html.

[30] Marin, Douglas H., Ronald A. Romero, Mauricio Guzmán, and Turner B. Sutton. "Black Sigatoka: an increasing threat to banana cultivation." Plant disease, vol. 87, no. 3, pp. 208-222, 2003.

[31] Anne V. "Sigatoka leaf spot." ProMusa, July 15, 2020, https://www.promusa.org/Sigatoka+leaf+spot.

[32] Queensland Government. "Tropical banana information kit.", 1998, http://era.daf.qld.gov.au/id/eprint/1656/6/5protrbn_part1.pdf.

[33] Maharachchikumbura, Sajeewa SN, Kevin D. Hyde, Johannes Z. Groenewald, J. Xu, and Pedro W. Crous. "Pestalotiopsis revisited." Studies in Mycology, vol. 79, no. 1, pp. 121-186, 2014.

[34] Satyagopal, K., S. N. Sushil, P. Jeyakumar, G. Shankar, O. P. Sharma, S. K. Sain, D. R. Boina et al. "AESA based IPM package for banana." Dept. Agricult. Cooperation, Ministry Agricult., Government India, Nat. Inst. Plant Health Manage., Hyderabad, India, Tech. Rep, p. 46, 2014.

[35] Anne V. "Fusarium wilt of banana." ProMusa, February 22, 2025, https://www.promusa.org/Fusarium+wilt.

[36] Kenganal, M. A. L. L. I. K. A. R. J. U. N., Yusuf Ali Nimbaragi, and G. S. Guruprasd. "Management of soft rot of banana caused by Erwinia carotovora sub sp. Carotovora.", 2017.

[37] Padmanaban, B., and S. Sathiamoorthy. "The banana stem weevil Odoiporus longicollis.", 1996.

[38] Sah, Shyam Babu, R. N. Gupta, Santosh Kumar, Tamoghna Saha, and B. B. Singh. "Seasonal incidence of banana scarring beetle in organic banana in Bihar." Annals of Plant Protection Sciences, vol. 30, no. 1, pp. 54-60, 2022.

[39] Waterhouse, D. F., and K. R. Norris. "Pentalonia nigronervosa Coquerel." Biological Control: Pacific Prospects, pp. 42-49, 1987.

[40] C. Jeeva. "Optimizers in Deep Learning." Scaler Topics, January 11, 2024, https://www.scaler.com/topics/deep-learning/optimizers-in-deep-learning/.

[41] Dozat, Timothy. "Incorporating nesterov momentum into adam.", 2016.

[42] Gonsalves, Tad, and Jaychand Upadhyay. "Integrated deep learning for self-driving robotic cars." In Artificial Intelligence for Future Generation Robotics, pp. 93-118. Elsevier, 2021.

[43] J. Brownlee. "Difference Between a Batch and an Epoch in a Neural Network." Machine Learning Mastery, August 15, 2022. https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/.

[44] "Dense Neural Networks: Understanding Their Structure and Function." Data Scientist, March 5, 2024 https://datascientest.com/en/dense-neural-networks-understanding-their-structure-and-function.

[45] Adedoja, Adedamola O., Pius A. Owolawi, Temitope Mapayi, and Chunling Tu. "Intelligent mobile plant disease diagnostic system using NASNet-mobile deep learning." IAENG International Journal of Computer Science, vol. 49, no. 1, pp. 216-231, 2022.

[46] Yu, X. Q., X. R. Yao, and J. Gao. "Study on Plant Diseases and Insect Pests Recognition Based on Deep Learning." IAENG International Journal of Computer Science, vol. 52, no. 1, pp. 111-120, 2025.

[47] T. P. Alvin. "Demystifying the Receiver Operating Characteristic (ROC) Curve." Towards AI, April 22, 2024, https://pub.towardsai.net/demystifying-the-receiver-operating-characteristic-roc-curve-0a509bb6f212.

Rushit R. Rivankar is a final student pursuing B.Tech Honors in Computer and Communication Engineering from the Manipal Institute of Technology, Manipal, India, and will be receiving his degree in 2025. His current research interests include AI/ML, Deep Learning, Computer Vision, and Data Science.

Smitha N. Pai (Senior Member, IEEE) received her bachelor's degree from MCE, Hassan, and the master's and Ph.D. degrees from the Manipal Institute of Technology (MIT), Manipal. She is currently a Professor and the Associate Dean of the School of Computer Engineering, MIT, Manipal Academy of Higher Education. Her current research interests include wireless sensor networks, machine learning, and computer vision. She has a good number of conferences and journals to her credit in this area.

Abhishek Rhisheekesan received his BE degree in Electronics from the Sardar Vallabhbhai National Institute of Technology (SVNIT) and an MS degree in Computer Science from Ira A. Fulton Schools of Engineering, Arizona State University. He was the Engineering Manager in Graphics Architecture Labs at Intel, Bangalore. Currently, he is the CEO and founder of aiRender Technology and Healthstream. His research interests include 3D graphics architecture, neural networks, plant disease detection, video conferencing, holographics, and 3D compression.

Deekshitha received her bachelor's degree in computer science from Poornaprajna College, Udupi, and received an M.C.A. degree in Data Science and Computer Application from Manipal Institute of Technology, Manipal. She is currently Head of the 3D Graphics Development Team at aiRender Technology Pvt Ltd. Her current research area includes game applications for 3D rendering (by using game engines like Unity, Blender, Unreal Engine 4.25), app development using React and Redux, machine learning algorithms, artificial Intelligence, and computer vision.

Lohith Prakash received his bachelor's degree in computer applications from National Degree College, Basavanagudi, and an M.C.A. degree in web technology and computer applications from PES University, Bangalore. He is currently the Head of the Full Stack App Development Team at aiRender Technology Pvt Ltd., and he is dedicated to advancing real-time 3D video conferencing solutions using Janus WebRTC. His current areas of interest are machine learning algorithms, artificial intelligence, and computer vision. His work combines technical leadership with hands-on development, focusing on optimizing connection stability, video quality, and spatial audio integration for an immersive conferencing experience.

Sunil V. G completed his PhD in Agricultural Extension from the Indian Agricultural Research Institute, New Delhi. He now works at Kerala Agricultural University as an Assistant Professor. He is the creator of the mobile application named "Farm Extension Manager" widely used in agriculture. His area of interest includes knowledge management and farmer innovation development.

Abel Philip Joseph is a pre-final-year student pursuing a B.Tech in Computer Science and Engineering from the Indian Institute of Information Technology, Kottayam, India. He is passionate about Deep Learning and Generative AI, Large Language Models, and their easy deployment over Cloud Services.