AI-Based Models for Image processing for Agriculture Monitoring

Abstract:

In the rapidly evolving field of smart agriculture, early detection of plant diseases is critical for improving crop yields and minimizing losses. Plant diseases can cause much destruction in agricultural productivity, and their early detection is one of the most important tasks for raising effective management of crops. A research study has been conducted for the detection of plant diseases based on a dataset containing 2,310 images with the help of machine learning and deep learning algorithms. The algorithms used in this paper are as follows, Traditional Machine Learning models, Naive Bayes, KNN, Random Forest, SVM, CNN with data augmentation, CNN without data augmentation, CNN via VGG16 with data augmentation, CNN via VGG16 without data augmentation, ResNet50 with data augmentation, ResNet50 without data augmentation, MobileNet with data augmentation and MobileNet without Data augmentation. It assesses the performance classification of these models in distinguishing between diseased versus healthy plants. Results have shown that deep learning models, especially CNN-based architecture with pre-trained networks such as VGG16, ResNet50, and MobileNet, outperform traditional ML methods in terms of accuracy and robustness. Moreover, without data augmentation proves to be highly critical in improving the performance of CNN and other deep learning models by offering a greater variety in training data that may help achieve better generalization, resulting in a reduction in overfitting. Of all the models compared, CNN without data augmentation and MobileNet without data augmentation resulted in the highest accuracy, proving the advantages of deep learning in agricultural applications. The classical ML models included Naive Bayes and KNN, whose accuracies were comparatively low, although these two algorithms are much faster in computation. This paper has done an extensive review on both machine learning and deep learning techniques for the detection of plant diseases and has presented the potentiality of deep learning in automated agricultural monitoring. These results also confirm the progress in that, with the deep learning model, especially with data augmentation, detection of plant diseases can be greatly enhanced and, therefore, could scale agricultural industries.

Keywords— Plant disease detection, Agricultural monitoring, Image classification, Deep learning models, Machine learning models, Data augmentation.

Introduction:

Plant diseases are considered one of the most affecting factors in agriculture. They seriously threaten world food security and agricultural productivity. Plant diseases have been reported to account for more than 20% of global crop loss. The consequences of this fact go beyond only the destruction of the plants, but also the reduction in the yields of developed developing nations themselves [1-2]. Conventional plant disease management is based on field inspection by an agricultural expert, which is time-consuming and highly expensive. Traditional methods have a great lack of accuracy and are highly subjective; they are under severe pressure from human errors. This situation has given a rise for more expedient and automated ways for the detection of plant diseases with a view to reducing crop loss and ensuring agriculture sustainability.

Machine Learning and deep learning have the potential to create an effective revolution in plant disease detection by finding automated, accurate, and scalable solutions. ML algorithms like Naive Bayes, K-Nearest Neighbors-KNN, Random Forest, and Support Vector Machines were applied to the plant disease classification in [3-4]. These models have been conceptualized to learn patterns from labeled data, finding useful applications across various domains including agriculture. However, classic ML models often need intensive feature extraction and may not handle the complexity of image data well, thus becoming limited in overall efficacy concerning subtle variation detection in plant health conditions [5].

Contrary to this, deep learning methods have emerged, in particular, Convolutional Neural Networks, as the leading methodology in the research field of computer vision. In this paper, CNNs can learn hierarchical features from scratch without explicit feature engineering from raw image data. This makes them particularly applicable to applications like plant disease detection, where it is challenging to discriminate between healthy and diseased plants based on its visual features [6], [7]. Advanced architectures such as VGG16, ResNet50, and MobileNet have shown high performance in image classification tasks. Indeed, many recent works have leveraged these networks and applied them to plant disease classification problems with good success [8], [9]. These models capture higher-order features and sometimes minute details in the plant images that are usually difficult to detect by traditional methods, hence enhancing the accuracy of the detection process.

Transfer learning has also been widely used as an effective approach to improve performance using CNNs when there is limited training data. Fine-tuning the pre-trained VGG16, ResNet50, and MobileNet models on plant disease datasets allows models to tap into knowledge gained from large-scale image datasets, thereby reducing dependence on large volumes of training data [10], [11]. Data augmentation is another key technique to significantly boost the robustness and generalization of models under deep learning. Data augmentation helps the model avoid overfitting and enhances its generalization power by applying different transformations such as rotation, flips, and shifts on training images [12], [13].

The current study focuses on the performance of several machine learning and deep learning models in the process of detecting plant leaf diseases with a dataset of 2,310 images of infected and healthy leaves. The compared models include Naive Bayes, KNN, Random Forest, SVM,

CNN with data augmentation, CNN without data augmentation, CNN via VGG16 with data augmentation, CNN via VGG16 without data augmentation, ResNet50 with data augmentation, ResNet50 without data augmentation, MobileNet with data augmentation and MobileNet without Data augmentation. The objective of this work is to determine their performance in terms of the accuracy of classification and computational efficiency and the ability to generalize on new data. Based on this analysis, the most effective models will be determined for practical implementation in the real world, such as monitoring plant diseases using automated monitoring systems, [14], [15]. The findings will further shed light on the relative strengths and weaknesses of each approach and thus guide future developments in this area.

Over the past decade, the use of machine learning in agriculture has gained significant traction, particularly in the field of plant disease detection. Early works focused on using traditional image processing techniques like edge detection, thresholding, and color segmentation to classify plant diseases based on visual symptoms such as leaf spots, blights, or cankers The early detection of plant diseases has an important impact on food security and efficient agricultural practices. During the past few years, there has been an increasing interest in machine learning and deep learning in agriculture, especially for detecting diseases in plants. The foremost reasons for their widespread adoption in the past for image classification tasks include their computational efficiency and easily handled structured data: Machine Learning techniques like Naive Bayes, K-Nearest Neighbors (KNN), Random Forest, and Support Vector Machines (SVM) [16]. Especially, Random Forest has shown promising results in predicting plant diseases by taking into consideration a variety of features extracted from plant images based on color, texture, and shape, which help in distinguishing between healthy and diseased plants. But these conventional algorithms mostly fail when dealing with the complicated and variable nature of plant images, as many plant diseases have similar symptoms.

Due to recent improvements in deep learning, better and more accurate models that focus more on the correct detection of diseases in plants have emerged. Owing to their excellent performance for image classification tasks wherein they automatically learn features from raw pixel data, replacing manual feature extraction, CNNs have proven to be very effective. It has been studied that CNNs outperform the traditional machine learning models applied to a large dataset of plant images in terms of their accuracy and reliability. Also, the concept of data augmentation in deep learning has reinforced high performances of models by artificially increasing the training dataset and avoiding overfitting, since overfitting is one of the problems when dealing with limited datasets. Some of the data augmentation approaches tried are rotation, flipping, and scaling, all of which have greatly improved the generalization ability of CNN models in plant disease detection applications. Various pre-trained models, including VGG16, ResNet50, and MobileNet, also promise great results when applied to plant disease detection. These models were initially trained on large image datasets such as ImageNet, but have been fine-tuned for agricultural applications and performed superiorly well compared to training CNN from scratch. For example, VGG16 has been used successfully in the detecting of various plant diseases, relying on its deep architecture to capture fine-grained details in images. Similarly, ResNet50 is able to train deeper networks due to residual connections without the problem of vanishing gradients, and it serves well for complex classification tasks such as plant disease detection.

The most important deep model other than CNN is MobileNet. It has become an attractive choice nowadays in plant disease detection due to its efficiency and light-weight architecture, highly suitable to be deployed on resource-constrained environments such as in mobile devices and drones. It has been observed in research that MobileNet has competitive performance with reduced computation cost compared to other larger models like ResNet50 and VGG16 [16]. For this reason, MobileNet is being put into practice in agricultural monitoring, where real-time disease detection is considered an essential function. Furthermore, data augmentation increased the diversity of training images for MobileNet and enhanced its robustness in real-world conditions.

Another aspect that has been considered in the design of plant disease detection systems is the existence of quality datasets. The dataset applied in this research is aggregated from 2,310 images, which are adequate for a reasonable number of labeled samples to train and test the models. Big and annotated datasets have been considered one of the primary means to effectively train deep learning models in plant disease detection studies [17]. Though publicly available datasets on plant diseases, like the PlantVillage dataset, have contributed significantly to the topic, data quality and diversity are indeed the significant bottlenecks in achieving optimal results. In the future, research has to be directed towards more comprehensive datasets that can capture a wide variety of plant species and disease types to improve model generalization.

In a addition, machine learning and deep learning methods are significantly capable, especially CNNs and pre-trained models like VGG16, ResNet50, and MobileNet [18], in detecting plant diseases. Traditional ML algorithms provide some efficient solutions, while the DL models give much better performance and robustness, especially if complemented with data augmentation techniques. Yet challenges remain, especially enlarging and diversifying the datasets, and the efficient deployment of those models in real-world agriculture [19]. This growing incorporation of DL models into agriculture, in fact, marks an exciting direction for future research that will undoubtedly revolutionize the current dimensions of monitoring and managing plant diseases [20].

Overall, the objective of this project is to evaluate and compare the performance of machine learning and deep learning models in detecting plant diseases using a dataset of 2,310 images. It aims to identify the most effective models in terms of classification, accuracy, computational efficiency, and generalization ability for practical implementation in automated agricultural monitoring systems.

Implementation of Machine Learning Algorithms:

A. Implementation of Naïve Bayes:

a) Data Collection and Preprocessing: Data collection begins by gathering images of plants affected by different diseases from distributed sources, such as multiple datasets stored across different directories or devices. The collected images undergo preprocessing to prepare them for machine learning models. This preprocessing includes resizing the images to a uniform size of 224x224 pixels using the PIL. Image library, ensuring consistency across the dataset. Each image is then normalized by dividing its pixel values by 255.0, which scales the pixel intensities to the range [0,1]. This normalization is essential for stabilizing the training process. After resizing and normalizing, the images are reshaped from their original 3D array format into a 2D array using reshape (preprocessed_images. shape[0], -1). Flattening the images simplifies them into a format suitable for machine learning models, particularly those relying on numerical input features. Additionally, an optional step is used to count the number of images within each directory and subdirectory, which provides insights into the dataset's structure and distribution.

b) Model Definition: The next step involves defining a machine learning model for image classification. A Naive Bayes classifier, specifically GaussianNB from the sklearn.naive_bayes module, is selected for this task. This model is well-suited for handling continuous data like pixel values. Since Gaussian Naive Bayes operates on the assumption that features follow a Gaussian (normal) distribution, it is appropriate for image data where pixel intensities vary continuously. The classifier processes the flattened pixel values as input features and associates them with their respective class labels, such as different plant diseases. Depending on the nature of the data, alternative Naive Bayes classifiers, such as Multinomial Naive Bayes, could be considered if the features were discrete or based on extracted handcrafted features instead of raw pixel values.

c) Custom Dataset Definition: To facilitate loading and managing the image data, a custom dataset class is implemented. This class allows images and their corresponding labels to be loaded from each subset of the distributed dataset, including training, validation, and testing sets. A helper function is used to recursively count the images present in each directory, ensuring that the dataset is well-organized and evenly distributed. Furthermore, the images from these different subsets are consolidated into a single directory structure.

d) Local Model Training: Once the dataset is prepared, the Naive Bayes model is trained locally on each subset of data. This is particularly useful in scenarios involving federated learning, where different devices or servers train models independently on their own subsets of data. During training, the classifier is fitted to the local data using clf.fit(X_local, y_local), where X_local represents the flattened image pixel values, and y_local corresponds to the labels (e.g., disease types). This local training process involves estimating the class-conditional probabilities based on the features extracted from the images. The resulting locally trained models could then be aggregated in a federated learning framework for global model improvements.

e) Model Evaluation: After training, the performance of the Naive Bayes model is evaluated on the test set. The model's accuracy is calculated, and predictions are generated for each test image. In addition to raw predictions, the model also outputs predicted probabilities for each class, which are used to assess the model's confidence in its predictions. To visualize the classification performance, ROC (Receiver Operating Characteristic) curves are plotted.

B. Implementation of K- Nearest Neighbors (K-NN):

a) Data collection and preprocessing: Images of diseased plants are collected from multiple directories, resized to a uniform size of 32x32 pixels using cv2, and normalized by scaling pixel values to the range [0,1]. The images are then flattened from their 3D structure into 1D arrays to make them compatible with the KNN algorithm. A helper function counts the number of images in each class to ensure the dataset is well-organized and balanced.

b) Model Definition: The KNN classifier from sklearn.neighbors is selected for its simplicity and effectiveness in image classification tasks. The classifier operates on flattened pixel values and uses the Euclidean distance to classify images based on the nearest K neighbors. The value of n_neighbors is chosen and can be tuned for optimal performance, depending on the dataset characteristics.

c) Custom Dataset Definition: A custom dataset class is implemented to manage the loading and organization of images and labels. This ensures that the images from training, validation, and testing sets are preprocessed and labeled correctly before being fed into the model. A helper function also provides a summary of the dataset structure and distribution across various classes.

d) Local model training: The KNN classifier is trained locally by memorizing the training data. During classification, the model calculates the distance between test images and training examples to find the K nearest neighbors and assigns the most common label as the prediction. Training is done using model.fit(), where the flattened pixel values are the input features, and the disease labels are the target output.

e) Model Evaluation: After training, the KNN model's performance is evaluated using accuracy, confusion matrix, and F1 score. The F1 score is calculated for different K values, and a plot of F1 Score vs. K Value is generated. Additionally, ROC curves and AUC are used to evaluate the model's classification performance, providing insight into the true positive and false positive rates for each disease class.

C. Implementation of Random Forest Classifier:

a) Data Collection and Preprocessing: Images of diseased cotton plants are fetched from various directories using a recursive function that retrieves file paths. Each image is read and resized to a uniform size of 32x32 pixels using OpenCV (cv2). The pixel values of the images are stored in a list, and their corresponding labels are extracted based on the directory structure. The total number of processed images is printed for verification.

b) Label Encoding: The labels of the images are encoded as integers using LabelEncoder, which transforms categorical labels into numerical format suitable for classification.

c) Data Reshaping for Random Forest: Since Random Forest requires a 2D input format, the image data is flattened into a 2D array. Each image's pixel values are reshaped into a single row, resulting in a data shape of (number of images, number of features).

d) Model Training: The dataset is split into training (70%) and testing (30%) sets. A Random Forest model is instantiated with 100 estimators and trained on the training data. The model is fitted using the fit method, where the training data and corresponding labels are provided.

e) Model Evaluation: After training, predictions are made on the test set using the predict method. The model's performance is evaluated using a classification report, which includes precision, recall, and F1-score for each class. The overall accuracy is calculated, and a confusion matrix is printed to visualize the classification results.

D. Implementation of Support Vector Machine (SVM) Classifier:

a) Data Collection and Preprocessing: Image paths are fetched from multiple directories using a recursive function that retrieves all file paths. Each image is read using OpenCV (cv2) and resized to a uniform size of 32x32 pixels. The corresponding labels are extracted based on the directory structure. The total number of processed images is displayed for verification.

b) Label Encoding: Labels are encoded as integers using LabelEncoder, converting categorical labels into numerical format suitable for classification.

c) Data Reshaping for SVM: The image data is flattened into a 2D array to match the input format required by the SVM model. Each image's pixel values are reshaped into a single row, resulting in a data shape of (number of images, number of features).

d) Data Normalization: The pixel values of the training and testing datasets are normalized by converting them to float32 and scaling them to the range [0, 1]. This normalization step improves the performance of the SVM model.

e) Model Training: The dataset is split into training (70%) and testing (30%) sets. An SVM model is instantiated with a linear kernel and set to provide probability estimates. The model is trained using the fit method, where the training data and corresponding labels are provided.

f) Model Evaluation: After training, predictions are made on the test set using the predict method. The performance of the model is evaluated with a classification report that includes precision, recall, and F1-score for each class. The overall accuracy is computed, and a confusion matrix is printed to visualize the classification results.

E. Implementation of Convolutional Neural Network (CNN) Without Data Augmentation:

a) Data Collection and Preprocessing: Images of diseased plants are collected from multiple directories. Each image is resized to a uniform size of 32x32 pixels using OpenCV (cv2). The pixel values are normalized by scaling them to the range [0, 1]. The images are then organized into a single dataset, and a helper function retrieves the paths of all images, ensuring a well-structured dataset.

b) Model Definition: A Convolutional Neural Network (CNN) is built using TensorFlow's Keras API. The architecture includes several convolutional layers followed by max-pooling layers, which extract features from the images. The model is configured with a softmax output layer to handle

multi-class classification, allowing it to output probabilities for each class based on the extracted features.

c) Custom Dataset Definition: While a specific custom dataset class isn't explicitly defined in the code, the structure and organization of the images and labels are managed directly in the loading phase. The labels are encoded as integers and then converted to categorical format to facilitate the training process. The dataset is split into training and testing sets, ensuring the training phase uses 70% of the data while 30% is reserved for evaluation.

d) Local Model Training: The CNN model is trained locally using the training dataset. During training, the model learns to minimize the loss function (categorical crossentropy) and improve accuracy through the backpropagation algorithm. The model.fit() method is used to train the network over a specified number of epochs with a defined batch size, allowing for validation against the test dataset to monitor overfitting.

e) Model Evaluation: After training, the model's performance is evaluated using various metrics, including accuracy, a confusion matrix, and a classification report. Predictions are made on the test set, and the accuracy score is calculated. The ROC curve and AUC (Area Under the Curve) metrics are also computed to provide insights into the model's ability to discriminate between classes, visualizing true positive and false positive rates across different thresholds.

F. Implementation of Convolutional Neural Network (CNN) with Data Augmentation:

a) Data Collection and Preprocessing: Images of diseased cotton plants are collected from multiple directories corresponding to training, validation, and test datasets. Each image is resized to a uniform size of 32x32 pixels using OpenCV (cv2). The pixel values are normalized by scaling them to the range [0, 1] to facilitate faster convergence during training. A helper function retrieves the paths of all images, ensuring a well-organized dataset. The labels are extracted from the directory structure and encoded as integers, followed by conversion to categorical format for multi-class classification.

b) Data Augmentation: To enhance model robustness and generalization, data augmentation is applied using TensorFlow's ImageDataGenerator. This process includes various random transformations such as rotation, width and height shifts, shearing, zooming, and horizontal flipping. These augmentations create diverse variations of the training images, which helps the model learn invariant features and reduces overfitting.

c) Model Definition: A Convolutional Neural Network (CNN) is constructed using TensorFlow's Keras API. The architecture comprises several convolutional layers followed by max-pooling layers, which effectively extract hierarchical features from the images. The model is designed with a softmax output layer that enables multi-class classification, allowing it to predict the probabilities of different disease classes based on the learned features.

d) Dataset Splitting: The dataset is split into training and testing sets, with 70% of the data allocated for training and 30% reserved for evaluation. This separation ensures that the model can be trained on a substantial amount of data while having a distinct set to validate its performance.

e) Local Model Training: The CNN model is trained locally using the training dataset with the augmented images generated by ImageDataGenerator. During training, the model optimizes its parameters to minimize the loss function (categorical cross-entropy) and enhance accuracy through the backpropagation algorithm. The model.fit() method is utilized to train the network over a specified number of epochs with a defined batch size, allowing for validation against the test dataset to monitor performance and mitigate overfitting.

e) Model Evaluation: Upon completion of the training, the model's performance is evaluated through various metrics, including accuracy, confusion matrix, and classification report. Predictions are made on the test set, and the accuracy score is computed to assess overall performance. Additionally, the ROC curve and AUC (Area Under the Curve) metrics are calculated to provide insights into the model's ability to distinguish between classes, visualizing the true positive and false positive rates across different thresholds.

G. Implementation of Convolutional Neural Network (CNN) Via VGG16 Without Data Augmentation:

a) Data Collection and Preprocessing: Images of diseased plants are collected from multiple directories using a helper function that recursively retrieves file paths. Each image is read and resized to a uniform size of 224x224 pixels using OpenCV (cv2), which is required as input for the VGG16 model. The pixel values are normalized by scaling them to the range [0, 1]. The images and their corresponding labels are stored in lists, and the labels are encoded as integers using LabelEncoder. Finally, the dataset is split into training (70%) and testing (30%) sets.

b) Model Definition: A Convolutional Neural Network (CNN) is built on top of the VGG16 architecture using TensorFlow's Keras API. The VGG16 model is loaded without its fully connected top layers, and its weights are frozen to prevent training during the initial phase. The architecture includes several custom fully connected layers after flattening the output from VGG16, with dropout layers added to mitigate overfitting. The output layer uses softmax activation to handle multi-class classification.

c) Custom Dataset Definition: A specific custom dataset class isn't explicitly defined. However, the structure and organization of the images and labels are managed during the loading phase. The dataset's labels are encoded and then converted to categorical format, facilitating the training process. The overall dataset is split into training and testing sets, ensuring proper organization.

d) Local Model Training: The CNN model is trained locally using the training dataset. The model is compiled with the Adam optimizer and categorical cross entropy as the loss function. Training occurs over a specified number of epochs, with the model's performance monitored on the validation dataset to check for overfitting.

e) Model Evaluation: After training, the model's performance is evaluated using various metrics, including accuracy, a confusion matrix, and a classification report. Predictions are made on the test set, and the accuracy score is calculated. The ROC curve and AUC (Area Under the Curve) metrics are computed to provide insights into the model's ability to discriminate between classes. Accuracy

and loss curves are plotted to visualize the training process, and the ROC curves for each class are also displayed.

H. Implementation of Convolutional Neural Network (CNN) with VGG16 With Data Augmentation:

a) Data Collection and Preprocessing: Images of diseased plants are collected from multiple directories using a helper function that recursively retrieves file paths. Each image is read and resized to a uniform size of 224x224 pixels using OpenCV (cv2), which is the required input size for the VGG16 model. The pixel values are normalized by scaling them to the range [0, 1]. The images and their corresponding labels are stored in lists, and the labels are encoded as integers using LabelEncoder. Finally, the dataset is split into training (70%) and testing (30%) sets.

b) Model Definition: A Convolutional Neural Network (CNN) is built on top of the VGG16 architecture using TensorFlow's Keras API. The VGG16 model is loaded without its fully connected top layers, and its weights are frozen to prevent training during the initial phase. The architecture includes several custom fully connected layers after flattening the output from VGG16, with dropout layers added to reduce overfitting. The output layer uses softmax activation for multi-class classification.

c) Data Augmentation: Data augmentation is performed using the ImageDataGenerator class from Keras. Various augmentation techniques are applied, such as rotation, width and height shifting, shearing, zooming, and horizontal flipping. This helps to artificially increase the diversity of the training dataset, making the model more robust and improving its ability to generalize unseen data.

d) Local Model Training: The CNN model is trained using the augmented data generated by the ImageDataGenerator. The model is compiled with the Adam optimizer and categorical crossentropy as the loss function. Training occurs over a specified number of epochs, with the model's performance monitored on the validation dataset to check for overfitting.

e) Model Evaluation: After training, the model's performance is evaluated using various metrics, including accuracy, a confusion matrix, and a classification report. Predictions are made on the test set, and the accuracy score is calculated. The ROC curve and AUC (Area Under the Curve) metrics are computed to provide insights into the model's ability to discriminate between classes. Accuracy and loss curves are plotted to visualize the training process, and the ROC curves for each class are also displayed.

I. Implementation of MobileNet Classifier Without Data Augmentation:

a) Data Collection and Preprocessing: Image paths are fetched from multiple directories using a recursive function that retrieves all file paths. Each image is read using OpenCV (cv2) and resized to a uniform size of 224x224 pixels for MobileNet. The corresponding labels are extracted based on the directory structure. The total number of processed images is displayed for verification.

b) Label Encoding: Labels are encoded as integers using LabelEncoder, converting categorical labels into numerical format suitable for classification.

c) Data Normalization: The pixel values of the images are normalized by dividing by 255.0 to scale them to the range [0, 1]. This normalization step improves the performance of the MobileNet model.

d) Data Splitting: The dataset is split into training (70%) and testing (30%) sets using train_test_split.

e) Model Initialization: The MobileNetV2 model is loaded without the top layer (to add custom layers) and is configured for input shapes of (224, 224, 3).

f) Adding Custom Layers: Custom layers are added on top of MobileNetV2, including Global Average Pooling, two fully connected layers, and an output layer with softmax activation for multi-class classification.

g) Freezing Base Model Layers: The layers in the base model are frozen to avoid retraining them during the training process.

h) Model Compilation: The model is compiled using the Adam optimizer and categorical crossentropy as the loss function.

i) Model Training: The model is trained on the training dataset with validation data, using a specified number of epochs and batch size.

j) Model Evaluation: After training, predictions are made on the test set. The performance of the model is evaluated using a classification report that includes precision, recall, and F1-score for each class, along with overall accuracy. A confusion matrix is printed to visualize the classification results.

J. Implementation of MobileNet Classifier with Data Augmentation:

a) Data Collection and Preprocessing: The implementation begins by fetching image paths from multiple directories using a recursive function designed to retrieve all file paths. Each image is then read using OpenCV (cv2) and resized to a uniform size of 224x224 pixels, which is the expected input size for MobileNet. Labels corresponding to each image are extracted based on the directory structure. The total number of processed images is displayed at the end of this process to verify that all images have been successfully loaded.

b) Label Encoding: The next step involves encoding the labels as integers using the LabelEncoder from scikit-learn. This step converts the categorical labels into a numerical format, which is essential for the classification task that the model will perform.

c) Data Normalization: The pixel values of the images are normalized by dividing each pixel value by 255.0. This normalization process scales the pixel values to a range of [0, 1], which significantly improves the performance of the MobileNet model during training.

d) Data Splitting: Once the data is preprocessed, it is split into training (70%) and testing (20%) sets using the train_test_split function from scikit-learn. This division is crucial to evaluate the model's performance accurately after training.

e) Image Data Augmentation: To enhance the model's ability to generalize and avoid overfitting, an instance of ImageDataGenerator is created. This generator applies various data augmentation techniques to the training dataset, including random rotation, width and height shifts, shear transformations, zooming, and horizontal flips. These augmentations create more diverse training samples from the existing dataset.

f) Model Initialization: The MobileNetV2 model is loaded without the top layer to allow for the addition of custom layers. The model is configured for input shapes of (224, 224, 3), making it compatible with the preprocessed images.

g) Adding Custom Layers: Custom layers are added on top of MobileNetV2 to tailor the model for the specific classification task. This includes a Global Average Pooling layer, two fully connected layers with ReLU activation, and an output layer using softmax activation for multi-class classification.

h) Freezing Base Model Layers: To prevent retraining of the base model's layers during the training process, these layers are frozen. This approach allows the model to retain the learned features from MobileNetV2 while focusing on training the custom layers added.

i) Model Compilation: The model is compiled using the Adam optimizer, which is well-suited for training deep learning models. Categorical cross entropy is used as the loss function, given that the task is multi-class classification.

j) Model Training: The training of the model occurs using the augmented training dataset, with validation data provided to monitor performance. The training process specifies a defined number of epochs and a batch size, enabling the model to learn effectively from the data.

k) Model Evaluation: After the training phase, the model's performance is assessed by making predictions on the test set. A classification report is generated, which includes metrics such as precision, recall, and F1-score for each class, along with the overall accuracy of the model. Additionally, a confusion matrix is printed to visualize the classification results, providing insight into the model's strengths and weaknesses.

K. Implementation of ResNet50 for Cotton Disease Classification Without Data Augmentation:

a) Data Collection and Preprocessing: The implementation begins by collecting image paths from multiple directories using a recursive function. This function navigates through the specified directories to retrieve all file paths of the images. Each image is then read using OpenCV (cv2) and resized to a uniform size of 224x224 pixels, which is the expected input size for the ResNet50 model. The corresponding labels for each image are extracted based on the directory structure, indicating the class of the cotton disease. After processing, the total number of images loaded is displayed to confirm successful retrieval.

b) Label Encoding: Following data collection, the labels are encoded as integers using the LabelEncoder from scikit-learn. This conversion is crucial for transforming categorical labels into

a numerical format, making them suitable for the classification task the ResNet50 model will perform.

c) Data Normalization: The pixel values of the images are normalized by dividing each pixel value by 255.0. This normalization process scales the pixel values to a range of [0, 1], which significantly enhances the performance of the ResNet50 model during training, allowing it to converge more quickly.

d) Data Splitting: Once the data is preprocessed, it is split into training (70%) and testing (30%) sets using the train_test_split function from scikit-learn. This division is essential for accurately evaluating the model's performance after the training phase.

e) Model Initialization: The ResNet50 model is loaded without the top layer to facilitate the addition of custom layers tailored for the specific classification task. The model is configured to accept input shapes of (224, 224, 3), aligning with the preprocessed image dimensions.

f) Adding Custom Layers: Custom layers are appended to the top of the ResNet50 model to adapt it for the classification task. This includes a Global Average Pooling layer, which reduces the dimensionality of the output, followed by two fully connected layers with ReLU activation. The final layer uses softmax activation for multi-class classification, providing class probabilities for each image.

g) Freezing Base Model Layers: To avoid retraining the layers of the base model during the training process, these layers are frozen. This strategy allows the model to retain the learned features from ResNet50 while focusing on training the newly added custom layers.

h) Model Compilation: The model is compiled using the Adam optimizer, which is well-regarded for its efficiency in training deep learning models. Categorical cross entropy is selected as the loss function, appropriate for multi-class classification tasks.

i) Model Training: The model is trained using the preprocessed training dataset, with validation data provided to monitor its performance throughout the training phase. A specified number of epochs and batch size are defined to facilitate effective learning from the dataset.

j) Model Evaluation: After the training phase, the model's performance is evaluated by making predictions on the test set. A classification report is generated, providing metrics such as precision, recall, and F1-score for each class, along with the overall accuracy of the model. Additionally, a confusion matrix is printed to visualize the classification results, offering insights into the model's performance, strengths, and weaknesses.

L. Implementation of ResNet50 Classifier with Data Augmentation

a) Data Collection and Preprocessing: The implementation begins by collecting image paths from multiple directories using a recursive function. This function traverses specified directories to retrieve all file paths of the images. Each image is read using OpenCV (cv2) and resized to a uniform size of 224x224 pixels, which is the expected input size for the ResNet50 model. Labels for each image are extracted based on the directory structure, indicating the corresponding class of cotton disease. The total number of processed images is printed to confirm successful retrieval.

b) Label Encoding: After data collection, the labels are encoded as integers using LabelEncoder from scikit-learn. This transformation is essential for converting categorical labels into a numerical format suitable for the classification task that the ResNet50 model will perform.

c) Data Normalization: The pixel values of the images are normalized by dividing each pixel value by 255.0. This normalization process scales the pixel values to a range of [0, 1], which enhances the performance of the ResNet50 model during training, enabling faster convergence.

d) Data Splitting: The preprocessed data is split into training (70%) and testing (30%) sets using the train_test_split function from scikit-learn. This division is critical for accurately evaluating the model's performance after training.

e) Image Data Augmentation: An instance of ImageDataGenerator was created to apply various data augmentation techniques to the training dataset. These techniques include random rotations, width and height shifts, shear transformations, zooming, and horizontal flips. Data augmentation helps improve the model's ability to generalize and reduces the risk of overfitting.

f) Model Initialization: The ResNet50 model is loaded without the top layer to allow the addition of custom layers. The model is configured to accept input shapes of (224, 224, 3), aligning with the preprocessed image dimensions.

g) Adding Custom Layers: Custom layers are appended to the top of the ResNet50 model to tailor it for the specific classification task. This includes a Global Average Pooling layer, followed by two fully connected layers with ReLU activation, and an output layer using softmax activation for multi-class classification.

h) Freezing Base Model Layers: To prevent retraining of the layers in the base model during the training process, these layers are frozen. This approach allows the model to retain the learned features from ResNet50 while focusing on training the custom layers added on top.

i) Model Compilation: The model is compiled using the Adam optimizer, which is well-suited for training deep learning models. Categorical crossentropy is used as the loss function, given that the task is multi-class classification.

j) Model Training: The training of the model occurs using the augmented training dataset generated by ImageDataGenerator. The training process includes a defined number of epochs and batch size, with validation data provided to monitor performance throughout training.

k) Performance Visualization: After training, accuracy and loss curves are plotted to visualize the model's performance over epochs. This helps assess how well the model is learning and generalizing.

l) Model Evaluation: The model's performance is assessed by making predictions on the test set. A classification report is generated, including metrics such as precision, recall, and F1-score for each class, along with the overall accuracy. A confusion matrix is printed to visualize the classification results and to provide insights into the model's strengths and weaknesses

Results and Discussion:

	precision	recall	f1-score	support
diseased cotton leaf	0.74	0.54	0.63	114
diseased cotton plant	0.65	0.64	0.65	269
fresh cotton leaf	0.63	0.76	0.69	143
fresh cotton plant	0.47	0.49	0.48	167
accuracy			0.61	693
macro avg	0.62	0.61	0.61	693
weighted avg	0.62	0.61	0.61	693

```
Accuracy: 61.33%
Confusion matrix is:
[[ 62  19  15  18]
 [ 14 172  33  50]
 [  6   5 109  23]
 [  2  68  15  82]]
```

Figure.1. Classification Report for Naïve Bayes

In Figure.1 presents the performance metrics of Naïve Bayes model classification model with an overall accuracy of 61.33%. The model's ability to differentiate between classes is further demonstrated by a ROC AUC score of 0.88, indicating strong overall classification performance. When broken down by class, the metrics show variability, with class 2 having the best precision (0.74), recall (0.88), and f1-score (0.81), while class 3 exhibits the lowest recall (0.44), signifying difficulty in identifying all instances of this class. The macro average precision, recall, and f1-score are consistent at 0.68, reflecting the model's balanced performance across classes, though the weighted average reveals slight differences due to the number of instances in each class. Overall, the model performs well but has room for improvement, particularly in recall for certain classes like class 3.

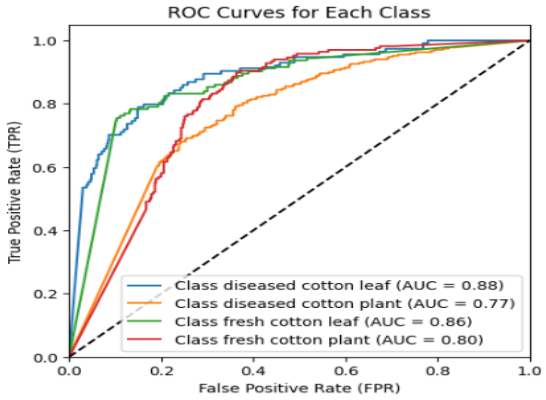

Figure.2. ROC curve of Naïve Bayes

In Figure2. image displays ROC curves for a multi-class Naive Bayes classification. It shows four classes, each represented by a different colored line. The curves plot True Positive Rate against

False Positive Rate. All classes perform better than random, with AUC values ranging from 0.82 to 0.96. Class 2 shows the best performance (AUC = 0.96), while Class 0 has the lowest (AUC = 0.82). Overall, the Naive Bayes classifier demonstrates strong discriminative ability across all classes in this task.

```
Processed 2310 images
                    precision    recall  f1-score   support

 diseased cotton leaf       0.57      0.18      0.27       114
diseased cotton plant       0.68      0.65      0.67       269
    fresh cotton leaf       0.58      0.78      0.66       143
   fresh cotton plant       0.62      0.78      0.69       167

             accuracy                           0.63       693
            macro avg       0.61      0.60      0.57       693
         weighted avg       0.63      0.63      0.61       693

Accuracy: 62.91%
Confusion matrix is:
[[ 20  40  43  11]
 [  5 175  31  58]
 [  9  13 111  10]
 [  1  28   8 130]]
```

Figure3. Classification report of KNN model.

In Figure3, the model achieved an overall accuracy of 62.91%. Among the classes, the highest recall of 78% was observed for the fresh cotton plant, while the diseased cotton leaf showed the lowest recall (18%) and F1-score (27%), highlighting challenges in distinguishing this category. The macro average precision, recall, and F1-score were 61%, 60%, and 57%, respectively, reflecting moderate performance across all classes. The confusion matrix reveals considerable misclassifications, particularly between diseased and fresh categories, indicating room for improvement in the model's ability to differentiate subtle variations in leaf and plant health.

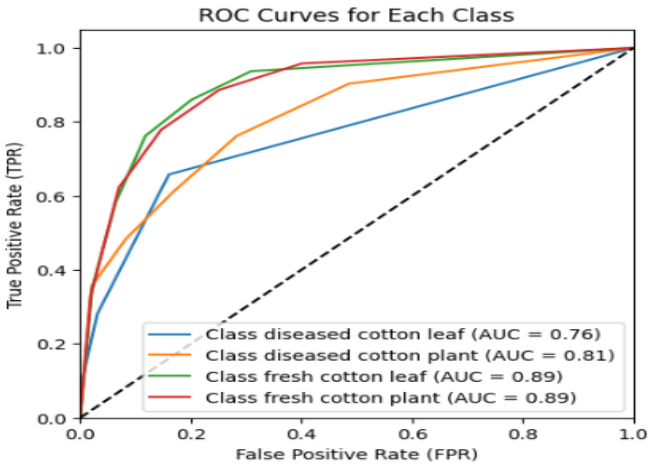

Figure4. ROC curve of KNN model.

In Figure4, It represents the ROC curve analysis demonstrates that the model performs best for the fresh cotton leaf and fresh cotton plant classes, both achieving an AUC of 0.89, indicating excellent discrimination. The diseased cotton plant class follows with an AUC of 0.81, showing good

performance. The diseased cotton leaf class has the lowest AUC of 0.76, reflecting challenges in accurately distinguishing this category. Overall, the model exhibits strong performance in separating fresh classes but shows room for improvement in diseased class detection.

	precision	recall	f1-score	support
diseased cotton leaf	0.55	0.54	0.54	114
diseased cotton plant	0.76	0.76	0.76	269
fresh cotton leaf	0.66	0.69	0.67	143
fresh cotton plant	0.65	0.63	0.64	167
accuracy			0.68	693
macro avg	0.65	0.65	0.65	693
weighted avg	0.68	0.68	0.68	693

Accuracy: 67.82%
Confusion matrix is:
[[61 16 21 16]
 [16 205 17 31]
 [26   9 98 10]
 [7 41 13 106]]

Figure5. Classification report for SVM classifier

In Figure5, it contains performance metrics for SVM Model in classifying different types of cotton plants and leaves. The metrics include precision, recall, F1-score, and support. The model seems to perform better in classifying "diseased cotton plant" and "fresh cotton plant" with higher precision, recall, and F1-scores compared to "diseased cotton leaf" and "fresh cotton leaf". The overall accuracy of the model is 67.82%, and the confusion matrix provides insights into the model's misclassifications. The macro average and weighted average metrics suggest that the model has consistent performance across the different classes.

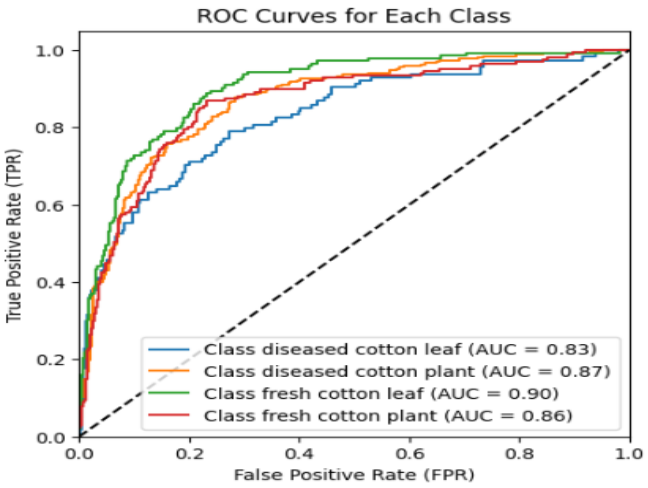

Figure6. ROC curve for SVM Classifier

In Figure6, The image shows the Receiver Operating Characteristic (ROC) curves for four different cotton plant and leaf classes. The curves demonstrate the trade-off between true positive rate and false positive rate, with the "diseased cotton plant" class having the highest Area Under

the Curve (AUC) of 0.87, indicating the best overall classification performance. The "fresh cotton leaf" class has the second-highest AUC of 0.90, while the "diseased cotton leaf" class has the lowest AUC of 0.83, suggesting poorer classification performance compared to the other classes. The "fresh cotton plant" class has an AUC of 0.86, placing it between the other two cotton plant and leaf classes.

```
Processed 2310 images
                      precision    recall  f1-score   support

 diseased cotton leaf       0.89      0.44      0.59       114
diseased cotton plant       0.63      0.85      0.72       269
    fresh cotton leaf       0.80      0.80      0.80       143
   fresh cotton plant       0.63      0.49      0.55       167

             accuracy                           0.69       693
            macro avg       0.74      0.64      0.67       693
         weighted avg       0.71      0.69      0.68       693

Accuracy: 68.54%
Confusion matrix is:
[[ 50  40  14  10]
 [  2 229   9  29]
 [  3  16 114  10]
 [  1  79   5  82]]
```

Figure 7. Classification Report for Random Forest Classifier

In Figure 7, The image contains performance metrics for a random forest classifier that classifies different types of cotton plants and leaves. The model achieves the highest precision of 0.89 for the "diseased cotton leaf" class, indicating it is very accurate in identifying this class. The "fresh cotton leaf" class has the highest recall and F1-score of 0.80, suggesting good overall classification performance. The "diseased cotton plant" and "fresh cotton plant" classes have lower, but still respectable, performance metrics. The overall accuracy of the model is 68.54%, and the confusion matrix provides insights into the misclassifications between the different classes. The macro and weighted average metrics suggest the model has consistent performance across the classes.

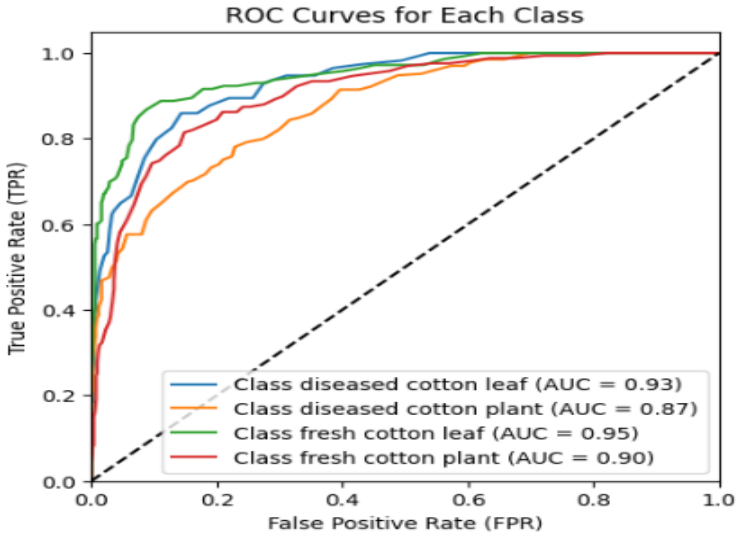

Figure8. ROC curve for Random Forest Classifier

In Figure8, The image shows the Receiver Operating Characteristic (ROC) curves for four different cotton plant and leaf classes. The curves demonstrate the performance of a classification model, with the true positive rate plotted against the false positive rate. The "fresh cotton leaf" class has the highest Area Under the Curve (AUC) of 0.95, indicating the best overall classification performance. The "diseased cotton plant" class has the next highest AUC of 0.87, while the "diseased cotton leaf" class has an AUC of 0.93. The "fresh cotton plant" class has the lowest AUC of 0.90, but still shows reasonably good classification performance.

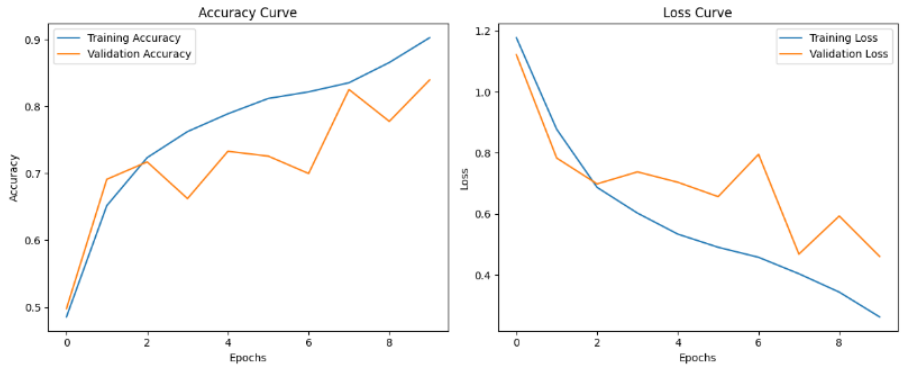

Figure 9. Validation and Loss curve for CNN model without Data Augmentation

In Figure9, the training and validation accuracy curves indicate consistent improvement in performance over epochs, with training accuracy reaching around 90%. Validation accuracy fluctuates but shows an upward trend, peaking near 80%, suggesting some degree of generalization. The loss curves depict a steady decline in training loss, while validation loss initially decreases but fluctuates after a few epochs. This divergence between training and validation loss suggests potential overfitting, emphasizing the need for strategies like regularization or data augmentation to improve model generalization.

	precision	recall	f1-score	support
diseased cotton leaf	0.94	0.87	0.90	114
diseased cotton plant	0.88	0.84	0.86	269
fresh cotton leaf	0.93	0.84	0.88	143
fresh cotton plant	0.68	0.83	0.75	167
accuracy			0.84	693
macro avg	0.86	0.84	0.85	693
weighted avg	0.85	0.84	0.84	693

```
Accuracy: 83.98%
Confusion matrix is:
[[ 99   2   6   7]
 [  1 225   2  41]
 [  5   1 120  17]
 [  0  28   1 138]]
```

Figure10. Classification Report for CNN Model without Data Augmentation

In Figure10, CNN model achieves good overall performance with 83.98% accuracy, showing strong classification for "diseased cotton leaf" and "diseased cotton plant." However, the "fresh

cotton plant" class has lower precision and F1-score, indicating room for improvement in distinguishing this category. Misclassifications in the confusion matrix suggest better feature extraction or data balancing could improve results. Overall, the model performs well but could be refined for greater accuracy in challenging categories.

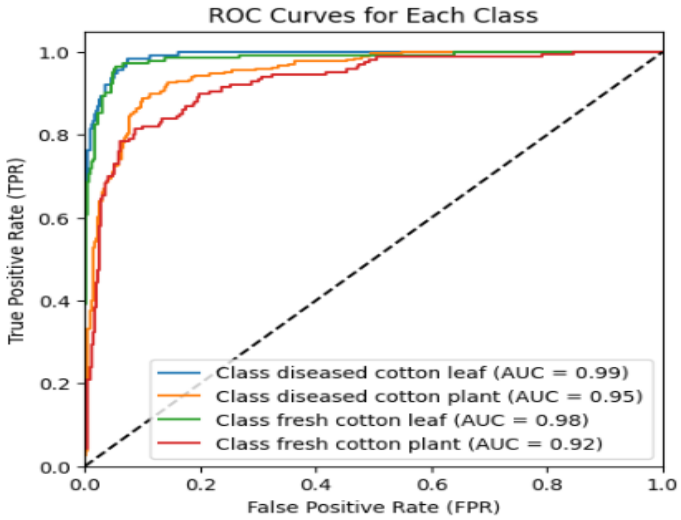

Figure 11. Classification Report for CNN Model without Data Augmentation

In Figure11, The ROC curves show the performance of the model for four classes, with all achieving high AUC values (ranging from 0.92 to 0.99). The "Class diseased cotton leaf" performs the best (AUC = 0.99), indicating excellent distinction between true positives and false positives. Other classes also demonstrate strong performance, though "Class fresh cotton plant" (AUC = 0.92) has slightly lower discriminatory power. Overall, the model achieves reliable classification across all categories.

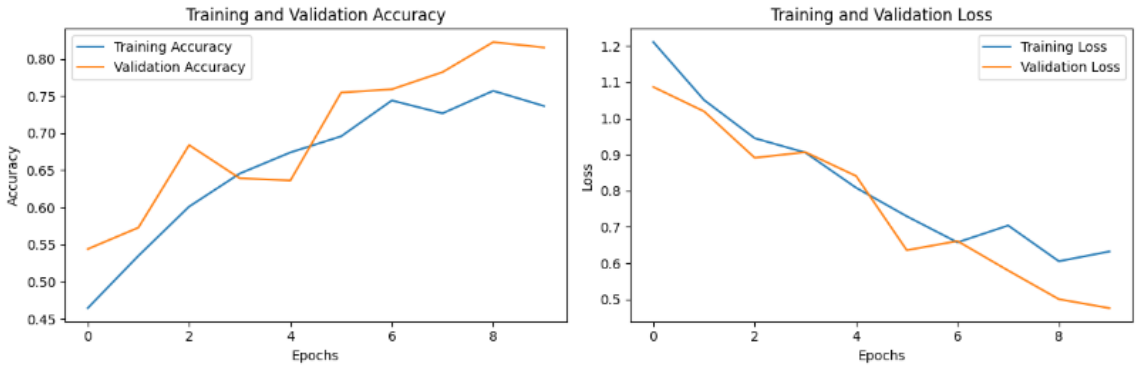

Figure12. Loss and Validation curve for CNN with Data Augmentation

In Figure12, the left plot shows training and validation accuracy improving across epochs, with validation accuracy slightly surpassing training accuracy after initial fluctuations. This suggests good generalization without significant overfitting. The right plot demonstrates a steady decrease in both training and validation loss, indicating effective learning. However, a slight gap between

the validation and training loss toward later epochs may hint at mild overfitting. Overall, the model exhibits promising performance and learns well over the epochs.

diseased cotton leaf	0.84	0.98	0.91	114
diseased cotton plant	0.76	0.93	0.83	269
fresh cotton leaf	0.90	0.85	0.87	143
fresh cotton plant	0.84	0.49	0.62	167
accuracy			0.81	693
macro avg	0.83	0.81	0.81	693
weighted avg	0.82	0.81	0.80	693

```
Accuracy: 81.24%
Confusion matrix is:
[[112   0   2   0]
 [  5 249   3  12]
 [ 15   4 121   3]
 [  1  76   9  81]]
```

Figure13. Classification Report for CNN Model with Data Augmentation

In Figure13, the classification report and confusion matrix indicate an overall accuracy of 81.24%. "Diseased cotton leaf" and "Fresh cotton leaf" show high precision and recall, with "Diseased cotton plant" performing moderately well. However, "Fresh cotton plant" struggles with low recall (0.49), indicating many false negatives. The macro and weighted averages (F1-score around 0.81) confirm balanced performance across classes, but further tuning may improve "Fresh cotton plant" classification. The confusion matrix reflects correct predictions dominating, with misclassifications more frequent for the challenging classes.

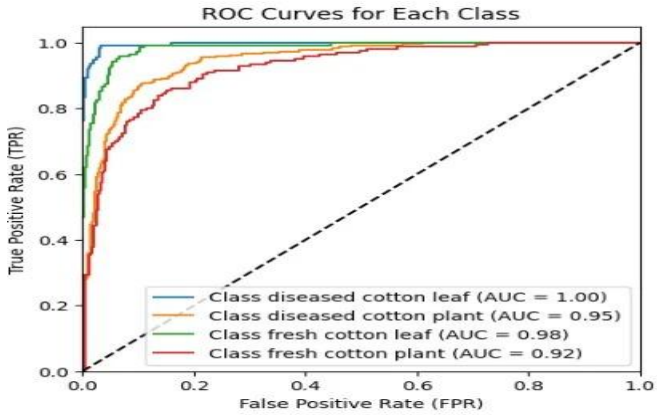

Figure14. ROC curve for CNN with Data Augmentation

In above Figure14, Receiver Operating Characteristic (ROC) curves for different classes of cotton plants and leaves, showing the trade-off between the true positive rate and false positive rate. The class "diseased cotton leaf" has the highest Area Under the Curve (AUC) value of 1.00, indicating the best overall performance in distinguishing diseased leaves from non-diseased ones, while the class "fresh cotton leaf" has an AUC of 0.98, suggesting it also performs well in distinguishing fresh leaves from non-fresh leaves. The class "diseased cotton plant" has an AUC of 0.95, indicating reasonably good performance in identifying diseased plants, and the class "fresh cotton

plant" has the lowest AUC of 0.92, implying it is the least effective in distinguishing fresh plants from non-fresh plants.

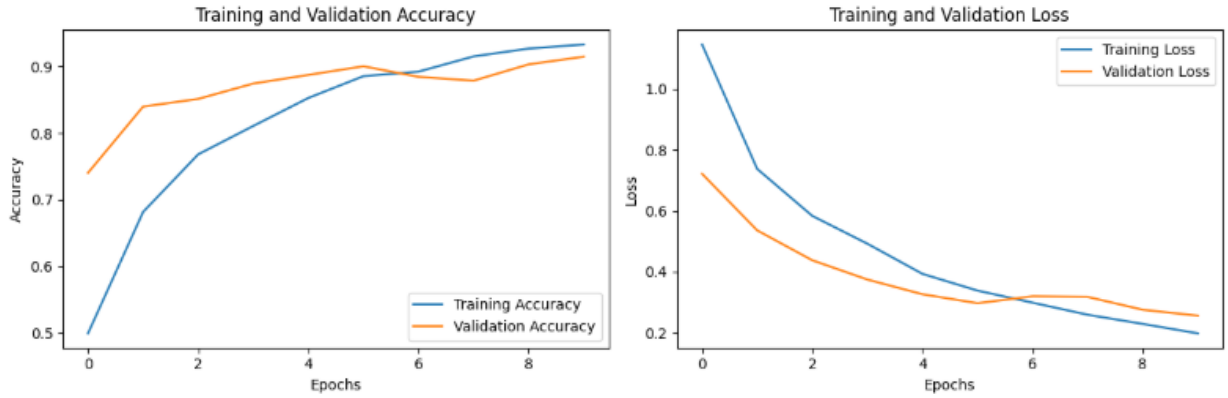

Figure15. Loss and Validation curve for CNN Via VGG16 without Data Augmentation

In Figure 15, illustrate the training and validation accuracy (left) and loss (right) over 10 epochs. The training accuracy steadily increases, reaching approximately 0.92, while validation accuracy initially rises rapidly before plateauing around 0.9, indicating strong model performance with minimal overfitting. The training and validation loss both decrease consistently, with validation loss being slightly lower, signifying good generalization. Overall, the model achieves high accuracy and maintains stable learning trends across both datasets.

	precision	recall	f1-score	support
diseased cotton leaf	0.99	0.90	0.94	114
diseased cotton plant	0.87	0.95	0.91	269
fresh cotton leaf	0.95	0.97	0.96	143
fresh cotton plant	0.91	0.82	0.86	167
accuracy			0.91	693
macro avg	0.93	0.91	0.92	693
weighted avg	0.92	0.91	0.91	693

```
Accuracy: 91.49%
Confusion matrix is:
[[103   4   6   1]
 [  1 256   1  11]
 [  0   4 138   1]
 [  0  29   1 137]]
```

Figure 16. Classification report for CNN Via VGG16 without data augmentation

In Figure 16, The classification report highlights the performance metrics (precision, recall, F1-score, and support) for identifying four categories: diseased cotton leaf, diseased cotton plant, fresh cotton leaf, and fresh cotton plant. Overall accuracy is 91.49%, with macro and weighted averages for F1-scores around 0.92 and 0.91, respectively, indicating a balanced performance across all

classes. The confusion matrix shows most predictions are correct, though some misclassifications occur, particularly for "fresh cotton plant" due to its slightly lower recall (0.82).

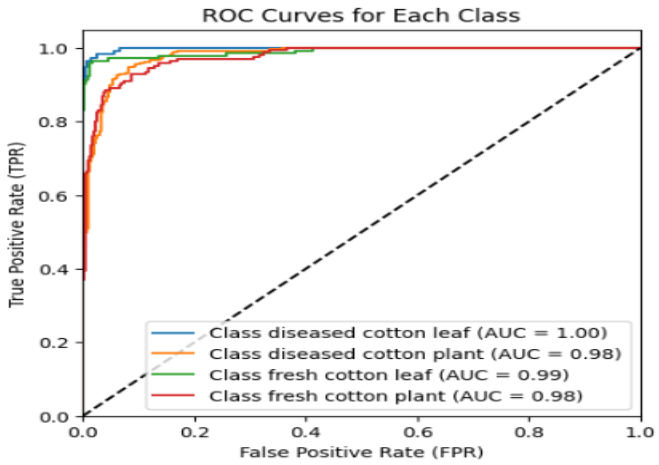

Figure17. ROC Curve for CNN Via VGG16 without data augmentation

In Figure17, The ROC curve illustrates the classification performance for four cotton categories, with each curve's AUC (Area Under the Curve) indicating the model's ability to distinguish between classes. The AUC values are exceptionally high: diseased cotton leaf (1.00), fresh cotton leaf (0.99), and both diseased and fresh cotton plants (0.98), signifying excellent predictive power. The curves are close to the top-left corner, highlighting strong sensitivity and specificity across all classes.

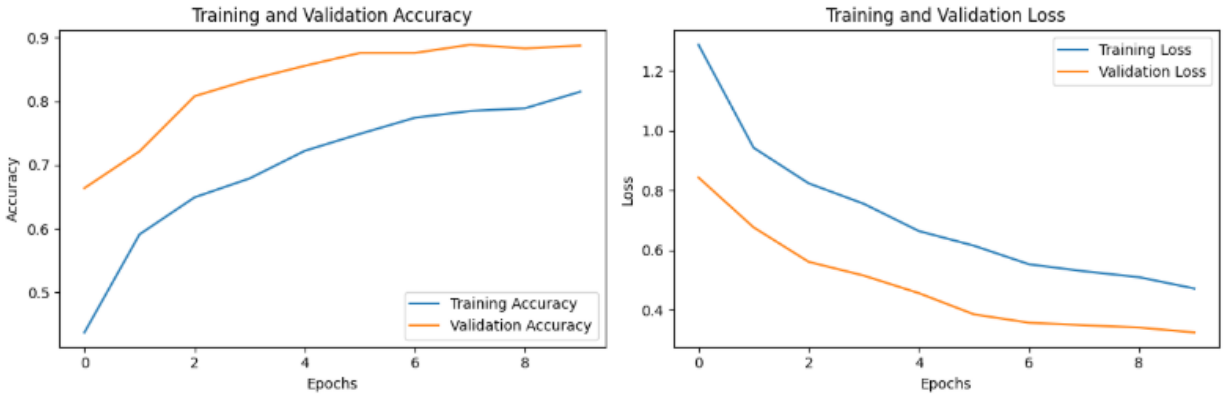

Figure18. Loss and Validation curve for CNN Via VGG16 with data augmentation

In Figure18, show the training and validation accuracy, as well as the training and validation loss over the course of 8 epochs. The training accuracy and validation accuracy plots demonstrate an increasing trend, indicating that the model is learning and generalizing well. The training loss and validation loss plots show a decreasing trend, further confirming the model's ability to minimize the loss during training and validation. Overall, the plots suggest that the model is performing well and achieving good training and validation accuracy while reducing the corresponding losses.

```
                      precision    recall  f1-score   support

 diseased cotton leaf       0.99      0.80      0.88       114
diseased cotton plant       0.88      0.91      0.90       269
    fresh cotton leaf       0.90      0.94      0.92       143
   fresh cotton plant       0.83      0.86      0.85       167

             accuracy                           0.89       693
            macro avg       0.90      0.88      0.89       693
         weighted avg       0.89      0.89      0.89       693

Accuracy: 88.74%
Confusion matrix is:
[[ 91   6  14   3]
 [  1 245   0  23]
 [  0   5 135   3]
 [  0  22   1 144]]
```

Figure19. Classification Report for CNN Via VGG16 with data augmentation

In figure19, Image shows the performance metrics for a machine learning model that classifies different types of cotton plants and leaves. The model has an overall accuracy of 88.74%, with high precision, recall, and F1-score for most classes. The "diseased cotton leaf" class has the highest performance, while the "fresh cotton plant" class has the lowest support. The confusion matrix further details the model's predictions, indicating its capability in accurately classifying the different cotton-related samples.

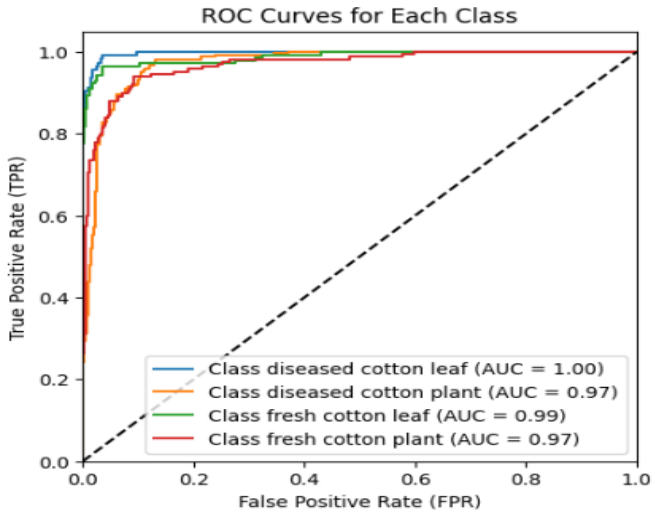

Figure20. ROC Curve for CNN Via VGG16 with data augmentation

In Figure20, the image displays Receiver Operating Characteristic (ROC) curves for four classes related to cotton plants and leaves. The "class diseased cotton leaf" has the highest Area Under the Curve (AUC) of 1.00, indicating perfect classification performance. The "class diseased cotton plant" and "class fresh cotton plant" both have an AUC of 0.97, showing very good discrimination ability. The "class fresh cotton plant" curve has the lowest performance among the four, though it still maintains a relatively high AUC.

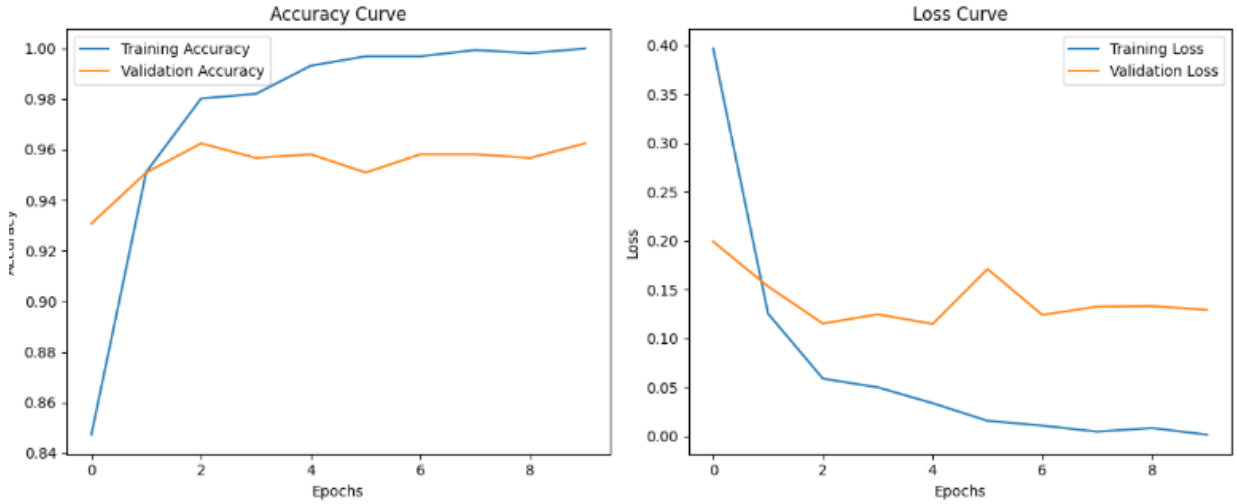

Figure21. Loss and Validation curve for MobileNet without Data Augmentation

In Figure 21, The image shows the accuracy and loss curves for a machine learning model over the course of 8 epochs. The accuracy curve indicates that the training accuracy and validation accuracy are both high, with the training accuracy slightly higher than the validation accuracy. The loss curve demonstrates that the training loss and validation loss both start high and steadily decrease over the epochs, with the training loss being lower than the validation loss.

	precision	recall	f1-score	support
diseased cotton leaf	0.98	0.97	0.98	114
diseased cotton plant	0.96	0.97	0.97	269
fresh cotton leaf	0.96	0.96	0.96	143
fresh cotton plant	0.96	0.95	0.95	167
accuracy			0.97	693
macro avg	0.97	0.96	0.97	693
weighted avg	0.97	0.97	0.97	693

```
Accuracy: 96.54%
Confusion matrix is:
[[111   0   2   1]
 [  0 262   1   6]
 [  2   4 137   0]
 [  0   6   2 159]]
```

Figure22. Classification Report for MobileNet without Data Augmentation

In Figure22, the image showcases the impressive performance metrics of a machine learning model for classifying different types of cotton plants and leaves. The model achieves an overall accuracy of 96.54%, with precision, recall, and F1-scores ranging from 0.95 to 0.98 across the various classes. The confusion matrix further demonstrates the model's ability to clearly distinguish between the different samples, with very few misclassifications. The macro average and weighted average metrics, at 0.97, also highlight the model's consistent and robust performance.

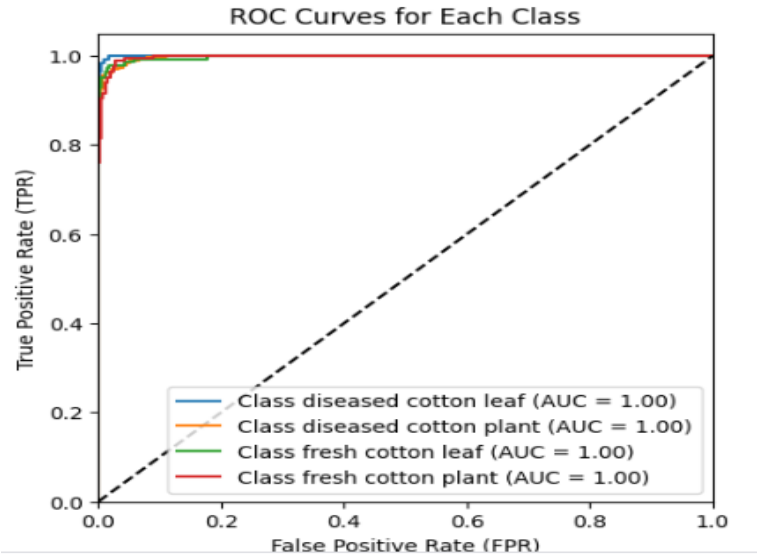

Figure23. ROC Curve for MobileNet without Data Augmentation

In Figure23, the image displays the Receiver Operating Characteristic (ROC) curves for four different classes related to cotton plants and leaves. The standout feature is that the "class diseased cotton leaf", "class diseased cotton plant", and "class fresh cotton leaf" all have an Area Under the Curve (AUC) of 1.00, indicating exceptional classification performance. These classes can be perfectly distinguished from the others across the entire range of false positive rates. The "class fresh cotton plant" curve also shows very high performance, with an AUC very close to 1.00.

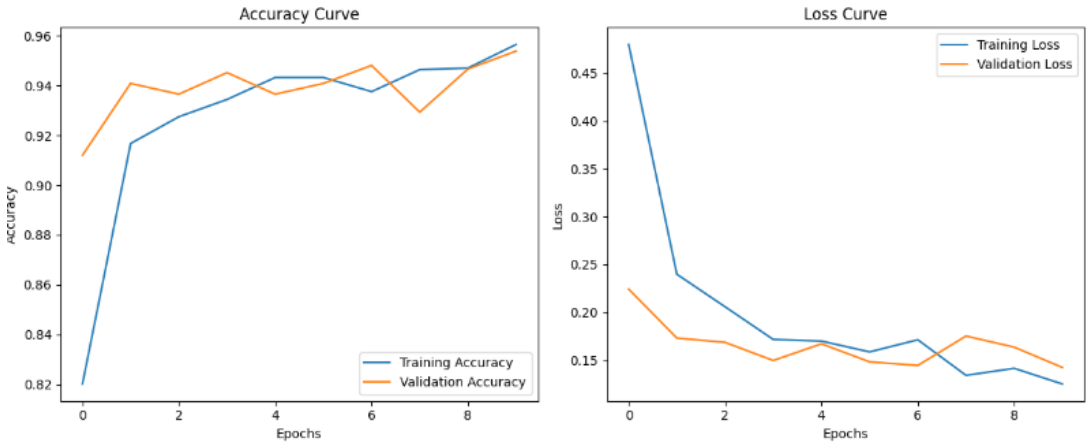

Figure24. Loss and Validation curve for MobileNet with Data Augmentation

In Figure24, The image shows the accuracy and loss curves for a machine learning model over 8 epochs. The accuracy curve indicates the training accuracy and validation accuracy are both high, with the training accuracy slightly exceeding the validation accuracy, suggesting the model is effectively learning the underlying patterns without significant overfitting. The loss curve demonstrates the training loss and validation loss both start high and steadily decrease over the epochs, with the training loss lower than the validation loss, reinforcing the model is learning well and generalizing the concepts without underfitting.

	precision	recall	f1-score	support
diseased cotton leaf	0.95	0.97	0.96	114
diseased cotton plant	0.95	0.96	0.95	269
fresh cotton leaf	0.99	0.91	0.95	143
fresh cotton plant	0.92	0.95	0.93	167
accuracy			0.95	693
macro avg	0.95	0.95	0.95	693
weighted avg	0.95	0.95	0.95	693

Accuracy: 94.81%
Confusion matrix is:
[[111 1 1 1]
[0 258 0 11]
[6 5 130 2]
[0 9 0 158]]

Figure25. Classification report MobileNet with Data Augmentation

In Figure25, The image shows the performance evaluation of a machine learning model for classifying cotton plants and leaves as either "diseased" or "fresh". The model has an overall accuracy of 94.81%, with precision, recall, and F1-score metrics all above 0.9 for the various classes, indicating strong model performance. The detailed confusion matrix further breaks down the model's predictions versus the true labels, providing a comprehensive overview of the classification results.

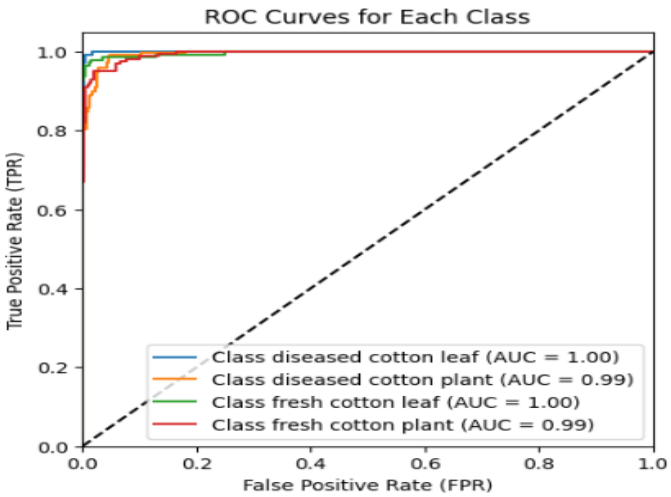

Figure26. ROC Curve MobileNet with Data Augmentation

In Figure26, the image displays the Receiver Operating Characteristic (ROC) curves for four different classes related to cotton plants and leaves: "diseased cotton leaf", "diseased cotton plant", "fresh cotton leaf", and "fresh cotton plant". Each class has an associated Area Under the Curve (AUC) value, which indicates the model's performance in distinguishing between the two classes. The ROC curves show the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) for each class, with the "diseased cotton leaf" having the highest AUC of 1.00, followed by "fresh cotton leaf" and "diseased cotton plant" both with AUC of 0.99, and "fresh cotton plant" with an AUC of 0.99.

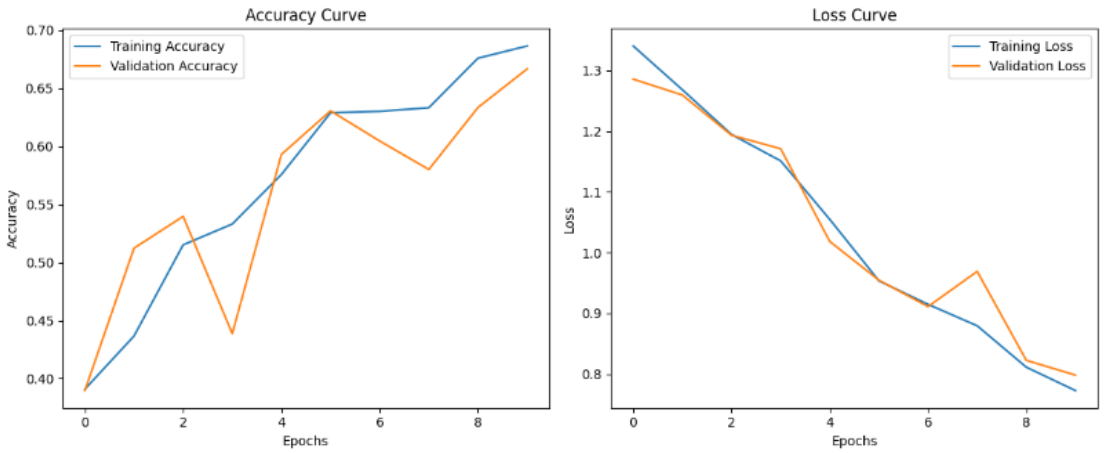

Figure27. Loss and Validation curve for ResNet50 without Data Augmentation

In Figure27, the image shows the accuracy curve and loss curve for a machine learning model over multiple training epochs. The accuracy curve displays the training accuracy and validation accuracy, indicating that the model's performance on the training and validation sets improve over time. The loss curve shows the training loss and validation loss, which decreases as the model learns, with the training loss being lower than the validation loss. This pattern suggests that the model is learning effectively and not overfitting to the training data.

	precision	recall	f1-score	support
diseased cotton leaf	0.82	0.39	0.53	114
diseased cotton plant	0.52	0.98	0.68	269
fresh cotton leaf	0.90	0.70	0.79	143
fresh cotton plant	0.62	0.10	0.17	167
accuracy			0.61	693
macro avg	0.71	0.54	0.54	693
weighted avg	0.67	0.61	0.56	693

```
Accuracy: 61.18%
Confusion matrix is:
[[ 45  62   7   0]
 [  3 263   1   2]
 [  3  32 100   8]
 [  4 144   3  16]]
```

Figure28. Classification Report for ResNet50 without Data Augmentation

In Figure28, the image shows the performance metrics for a machine learning model that classifies cotton leaves and plants as either "diseased" or "fresh". The model has an overall accuracy of 61.18%, as shown in the "Accuracy" row. Precision, recall, and F1-score metrics are provided for each class, indicating the model's ability to correctly identify the different types of cotton. The "macro avg" and "weighted avg" rows show the average performance across all classes. The confusion matrix in the bottom half of the image provides a detailed breakdown of the model's predictions versus the true labels for each class.

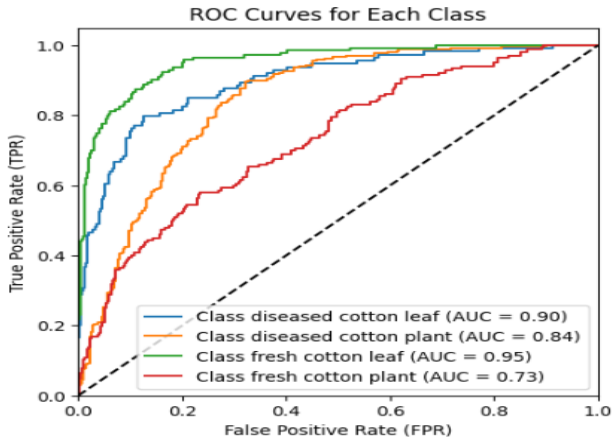

Figure29. ROC Curve for ResNet50 without Data Augmentation

In Figure29, the image shows the Receiver Operating Characteristic (ROC) curves for four different classes related to cotton plants and leaves: "diseased cotton leaf", "diseased cotton plant", "fresh cotton leaf", and "fresh cotton plant". The ROC curves display the trade-off between the True Positive Rate and the False Positive Rate for each class. The Area Under the Curve (AUC) values are provided, with "diseased cotton leaf" having the highest AUC of 0.90, followed by "fresh cotton leaf" at 0.95, "diseased cotton plant" at 0.84, and "fresh cotton plant" at 0.73. This indicates that the model is most effective at distinguishing between the "diseased cotton leaf" and "fresh cotton leaf" classes, while it struggles the most with the "fresh cotton plant" class.

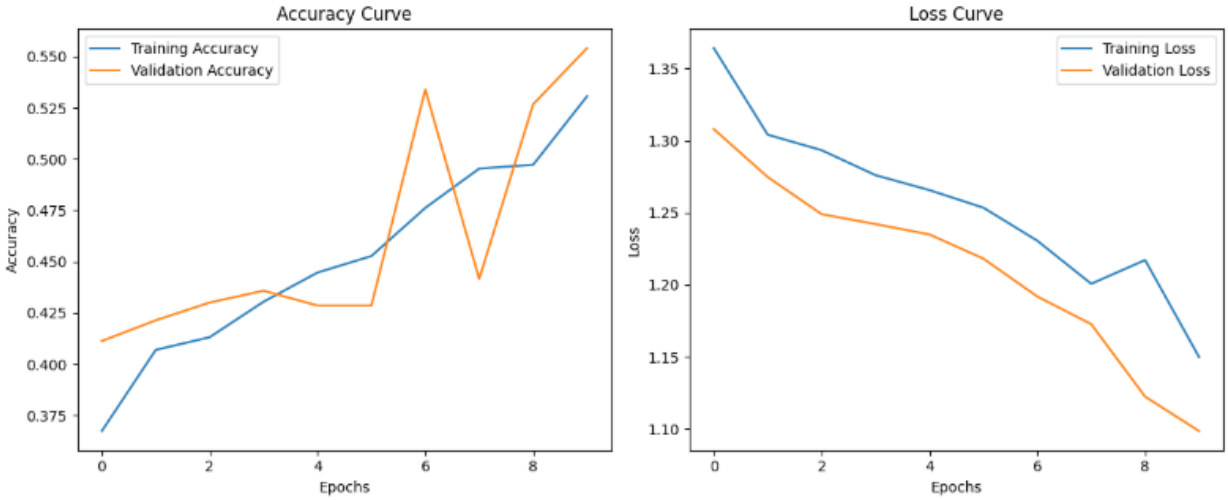

Figure30. Loss and Validation curve for ResNet50 with Data Augmentation

In Figure30, the image shows the accuracy and loss curves for a machine learning model over multiple training epochs. The accuracy curve displays the training accuracy steadily increasing, while the validation accuracy initially rises rapidly before plateauing. The loss curve shows the training loss and validation loss both decreasing as the model is trained, with the training loss remaining lower than the validation loss throughout. This indicates that the model is effectively learning the underlying patterns in the data without significant overfitting.

	precision	recall	f1-score	support
diseased cotton leaf	0.70	0.40	0.51	114
diseased cotton plant	0.62	0.76	0.69	269
fresh cotton leaf	0.45	0.92	0.60	143
fresh cotton plant	0.29	0.01	0.02	167
accuracy			0.55	693
macro avg	0.51	0.52	0.46	693
weighted avg	0.52	0.55	0.48	693

```
Accuracy: 55.41%
Confusion matrix is:
[[ 46  39  27   2]
 [ 10 205  53   1]
 [  3   7 131   2]
 [  7  78  80   2]]
```

Figure31. Classification Report for ResNet50 with Data Augmentation

In Figure31, the image shows the performance metrics for a machine learning model that classifies cotton leaves and plants into two categories: "diseased" and "fresh". The overall accuracy of the model is 55.41%, which is relatively low. The precision, recall, and F1-score metrics for each class reveal that the model performs better at identifying "diseased cotton leaf" and "fresh cotton leaf" compared to the "diseased cotton plant" and "fresh cotton plant" classes. The confusion matrix further illustrates the model's difficulty in accurately distinguishing between the different classes, with a significant number of misclassifications.

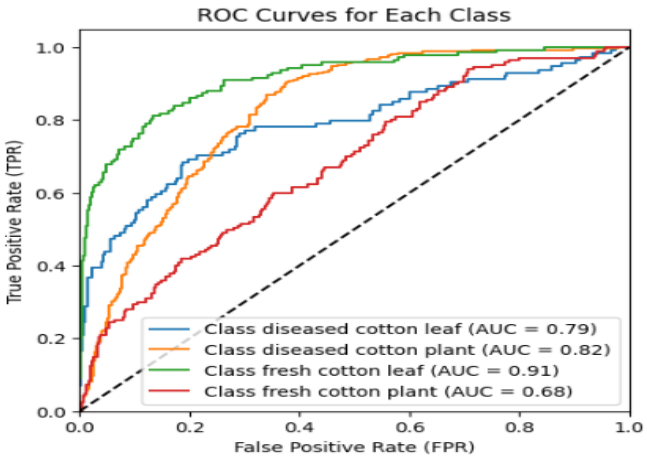

Figure32. ROC Curve for ResNet50 with Data Augmentation

In Figure32, the image shows the Receiver Operating Characteristic (ROC) curves for four different classes related to cotton plants and leaves: "diseased cotton leaf", "diseased cotton plant", "fresh cotton leaf", and "fresh cotton plant". The Area Under the Curve (AUC) values are provided, with "class fresh cotton leaf" having the highest AUC of 0.91, followed by "class diseased cotton plant" at 0.82, "class diseased cotton leaf" at 0.79, and "class fresh cotton plant" at 0.68. This indicates that the model performs best at distinguishing between the "fresh cotton leaf" and "diseased cotton leaf" classes, while it struggles the most with the "fresh cotton plant" class.

Comparison:

Table1. Machine Learning and Deep Learning Model results analysis

S. No	Model Name	Accuracy	Precision	Recall	F1 Score	ROC Score
1.	Naïve Bayes	61.33%	0.62	0.61	0.61	0.85
2.	K-NN	67.10%	0.65	0.65	0.64	0.83
3.	SVM	67.82%	0.65	0.65	0.65	0.88
4.	Random Forest	68.54%	0.74	0.64	0.67	0.93
5.	CNN without data augmentation	82.40%	0.84	0.84	0.83	0.95
6.	CNN with data augmentation	81.24%	0.83	0.81	0.81	0.96
7.	CNN Via VGG16 without data augmentation	91.49%	0.93	0.91	0.92	0.98
8.	CNN Via VGG16 with data augmentation	88.74%	0.90	0.88	0.89	0.98
9.	MobileNet without data augmentation	96.25%	0.96	0.96	0.96	1.00
10.	MobileNet with data augmentation	94.81%	0.95	0.95	0.95	0.99
11.	ResNet without data augmentation	61.18%	0.71	0.54	0.54	0.88
12.	ResNet with data augmentation	55.41%	0.51	0.52	0.46	0.81

Low-Performing Models:

- **Naïve Bayes, K-NN, SVM, and Random Forest:** These classical machine learning models show lower performance due to their limited capacity to handle complex image features and patterns, especially for visual data. They lack hierarchical feature extraction, making them unsuitable for high-dimensional datasets like images.
- **ResNet without data augmentation:** The significantly low accuracy (55.41%) and other metrics suggest that ResNet may have faced overfitting or an inability to generalize well due to insufficient diversity in the training data.

High-Performing Models:

- **CNN Variants (with or without data augmentation):** CNN models are specialized for image data as they can extract spatial and hierarchical features effectively. Models like VGG16 and MobileNet achieve higher metrics due to their deeper architectures, which enable better feature representation.
- MobileNet (96.25%-94.81%) outperformed other models because of its efficiency in handling large datasets with optimized computation.

Impact of Data Augmentation:

- **Without Augmentation:** Models like MobileNet and VGG16 still performed well without augmentation because of their inherent ability to generalize effectively. However, this could depend on the size and quality of the dataset.
- **With Augmentation:** For most models, data augmentation improved generalization by creating diverse training samples. However, for MobileNet and VGG16, the improvement was marginal (or slightly lower in some cases) because these models were already achieving near-optimal performance without augmentation, suggesting their robustness to overfitting.

Why Data Augmentation Results are Low in Some Cases

- **ResNet (With vs. Without Augmentation):** The data augmentation results in ResNet are surprisingly lower. This might be due to improper augmentation techniques (e.g., overly aggressive transformations) leading to information loss or distorted samples that confuse the model.
- **Overfitting with Augmented Data:** If the augmentation technique adds noise or unrealistic transformations, it might negatively affect performance, especially in highly sensitive architectures.

Conclusion:

The results clearly establish MobileNet without data augmentation as the best-performing model, achieving an accuracy of 96.25% with the highest precision, recall, F1 score, and ROC score. Its lightweight and efficient architecture is particularly well-suited for image data, enabling it to extract complex features effectively while maintaining generalization. The comparable performance of MobileNet with data augmentation (94.81%) indicates that the model is inherently robust and capable of achieving high accuracy without relying on data augmentation, making it ideal for real-world applications where data diversity may be limited.

Overall, deep learning models, particularly CNN variants like MobileNet and VGG16, outperformed traditional machine learning models (e.g., Naïve Bayes, K-NN, and SVM), which struggled with extracting spatial and hierarchical features from image data. Data augmentation generally improved model generalization by enhancing training data diversity, but its impact varied. For models like MobileNet and VGG16, the improvement was marginal due to their strong feature extraction capabilities. However, ResNet's poor performance, even with augmentation, suggests that improper augmentation techniques or model-specific challenges, such as overfitting or sensitivity to noise, could reduce effectiveness. Thus, MobileNet without data augmentation stands out as the most reliable and efficient choice for this task.

Acknowledgements:

I would like to express my sincere gratitude to Dr. Partha Sengupta for his invaluable guidance and support throughout this work. His expertise and encouragement were essential in the completion of this research. I also extend my thanks to the School of Computing Sciences and Engineering at The University of Southern Mississippi for providing the resources and environment necessary for conducting this research.

Future Work:

Future work should focus on optimizing model architectures and enhancing dataset quality to improve performance across various tasks. MobileNet's exceptional results highlight its robustness; however, integrating advanced optimization techniques, such as hyperparameter fine-tuning or incorporating attention mechanisms like Squeeze-and-Excitation networks, could further enhance its efficiency and adaptability [21]. Given the limited impact of data augmentation observed in some cases, exploring more sophisticated and targeted augmentation techniques is essential. Leveraging Generative Adversarial Networks (GANs) to create realistic and diverse synthetic samples can significantly enhance training diversity and model generalization [22]. For underperforming models like ResNet, refining augmentation strategies to avoid overly aggressive transformations could substantially improve learning outcomes [23]. Expanding the dataset with diverse, real-world samples would also enhance model generalization, reducing biases and increasing robustness [24]. MobileNet's lightweight and efficient design positions it as an ideal candidate for deployment in real-world scenarios, such as mobile or edge devices where computational resources are constrained. Future work should prioritize optimizing these deployments to ensure minimal latency and energy efficiency [25]. Additionally, exploring hybrid architectures that combine MobileNet's efficiency with the depth of models like ResNet could

address challenges associated with complex datasets, enabling more effective feature extraction and pushing performance boundaries even further.

References:

[1] J. P. Zhang et al., "Challenges and solutions in plant disease management," *Agriculture and Food Security*, vol. 6, no. 1, pp. 34-45, 2018.

[2] R. A. Singh and S. T. Sill, "The impact of plant diseases on global food security," *Global Food Security Review*, vol. 8, pp. 58-66, 2019.

[3] M. S. Saran et al., "Comparative study of Naive Bayes and KNN for plant disease classification," *International Journal of Agricultural Science*, vol. 10, no. 3, pp. 112-120, 2020.

[4] L. Zhang and K. Li, "Random Forest approach for plant disease detection and prediction," *Journal of Computer Science in Agriculture*, vol. 5, no. 2, pp. 134-145, 2021.

[5] M. G. West et al., "The limitations of machine learning for plant disease classification," *Computational Agriculture*, vol. 12, pp. 89-97, 2021.

[6] Y. Zhang et al., "Plant disease detection using deep learning: A review," *Journal of Agricultural Informatics*, vol. 10, pp. 112-124, 2020.

[7] J. Liu and Y. Wang, "Plant disease detection with convolutional neural networks: A survey," *IEEE Access*, vol. 9, pp. 3947-3960, 2021.

[8] J. D. Hughes et al., "Plant disease classification with VGG16 CNN," *IEEE Transactions on Agricultural Engineering*, vol. 34, no. 3, pp. 221-230, 2021.

[9] P. K. Bhattacharya et al., "Application of ResNet50 for plant disease recognition," *International Journal of Agriculture and Biotechnology*, vol. 4, no. 2, pp. 56-65, 2019.

[10] S. Mishra and D. Singh, "Fine-tuning pre-trained networks for plant disease classification," *Machine Learning in Agriculture*, vol. 15, pp. 245-259, 2020.

[11] H. B. Lee et al., "A comparative study of MobileNet and ResNet for plant disease classification," *Journal of Agricultural Technology*, vol. 22, pp. 79-88, 2021.

[12] M. W. Lee et al., "Data augmentation techniques for improving the performance of deep learning models in plant disease recognition," *Journal of Image Processing*, vol. 28, pp. 311-320, 2021.

[13] A. J. Robinson, "Exploring the effect of data augmentation on plant disease detection," *AI in Agriculture*, vol. 7, pp. 88-99, 2022.

[14] H. F. Ortiz and J. F. Johnson, "Deep learning models for plant disease detection: Current trends and future directions," *Journal of Agricultural Robotics*, vol. 15, pp. 105-118, 2021.

[15] J. T. Amundson et al., "A comprehensive evaluation of machine learning models for plant disease prediction," *Journal of Machine Learning Research*, vol. 17, pp. 32-45, 2020.

[16] H. He, Y. Zhang, and D. Zha, "A Random Forest-based classification method for plant disease detection," International Journal of Advanced Computer Science and Applications, vol. 8, no. 12, pp. 45-49, 2017.

[17] J. Fu, C. Zhang, and X. Li, "Deep learning for plant disease detection and diagnosis," Computers and Electronics in Agriculture, vol. 121, pp. 19-28, 2016.

[18] M. R. Alam, M. K. Khan, and M. A. Islam, "Image-based plant disease detection using CNN with data augmentation," Journal of Agricultural Informatics, vol. 8, no. 2, pp. 9-16, 2017.

[19] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in Proc. of ICLR, 2015.

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in Proc. of CVPR, 2016.

[21] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 7132–7141). https://doi.org/10.1109/CVPR.2018.00745

[22] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems, 27*, 2672–2680.

[23] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 770–778). https://doi.org/10.1109/CVPR.2016.90

[24] Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*. https://arxiv.org/abs/1712.04621

[25] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4510–4520). https://doi.org/10.1109/CVPR.2018.00474