RESEARCH ARTICLE OPEN ACCESS

Multi-Step Approach for Plant Disease IdentificationUsing Neural Network

Smita Upadhyay*, Prof. Krishna Hingrajiya**
**(Department of Computer Engineering, SAL Institute of Technology & Engineering Research, Ahmedabad Email: smitaupadhyay811@gmail.com)*
*** (Department of Computer Engineering, SAL Institute of Technology & Engineering Research, Ahmedabad Email: krishna.hingrajiya@sal.edu.in)*

ABSTRACT

Accurate identification of plant diseases is crucial for preventing reductions in agricultural productivity and quantity. The study of patterns that are visible to the human eye and indicate disease states in plants is known as plant pathology. Maintaining plant health and identifying plant diseases are essential to agriculture's long-term viability. Manually monitoring plant diseases is a challenging task. It necessitates a substantial amount of effort, and specialized knowledge in the field of plant diseases, and also demands a significant amount of time for processing. Therefore, image processing is utilized to identify plant illnesses by taking leaf images and comparing them with existing data sets. The dataset comprises many plant species in picture format. The primary objective of this research was to utilize image processing and MobileNet V2 with Transfer Learning to identify plant illnesses promptly and precisely. The role of agriculture in guaranteeing food security and economic stability is significant, and quick detection of plant diseases is essential in reducing crop loss. The objective of this research is to address the existing limitations in disease detection methods by proposing a new and automated approach for identifying plant diseases where we got 99.00 accuracy.

.

Keywords - Data Modeling, Image Enhancement, Image Translation, Plant Disease Detection, Real-Time Disease Monitoring

1. Introduction

Many economies rely heavily on agriculture, which provides food and raw materials for a wide range of industries [1]. Crop health and productivity are jeopardized by a range of diseases, resulting in substantial economic losses and food scarcity. Conventional illness detection methods depend on human proficiency, which can be time-consuming and prone to errors [2]. Advancements in computer vision, machine learning as well as in deep learning have the potential to enable automated, efficient, and dependable identification of plant diseases.
India is an agrarian nation, with over 70% of its population relying on agriculture [3]. Farmers possess a vast array of options regarding selecting diverse and suitable plants, as well as identifying the most suited insecticides for their plants [4]. Consequently, crop damage would cause productivity to drop sharply, which would eventually affect the economy.

As the plant's most sensitive component , the leaves show signs of disease early on [5]. It is essential to continuously monitor the crops for illnesses from their first stage of growth until they are ready for harvest.

Originally, the technique employed to monitor plants for illnesses was visually inspecting them with the naked eye [6]. To manually survey the crop fields using this labor-intensivemethod, skilled individuals are needed. Many approaches have been used recently to develop automated and semi-automated systems for diagnosing illnesses in plants [7]. The ability to detect diseases by simply seeing symptoms on plant leaves not only simplifies the process but also reduces costs [8]. Thus far, these methods have proven to be rapid, cost- effective, and more precise compared to the conventional approach of farmers manually observing.
Disease symptoms are typically observed on the fruit, stem, and leaves.Based on how the disease manifests itself, the plant leaf that is utilized for disease detection is chosen [9]. Agriculturalists frequently lack a thorough understanding of crop cultivation and the diseases that can impact their harvests [10]. Rather than consulting specialists

,farmers can increase crop productivity by using this article. [11].

A greenhouse, also known as a glasshouse or, if equipped with adequate heating, a hothouse, is a building constructed mostly with transparent materials, such as glass, that is used for cultivating plants that require controlled climatic conditions [12]. This report provides valuable assistance to greenhouse farmers by offering practical strategies in lightof the increasing significance of greenhouse farming [13]. A variety of approaches can beused to assess plant disease detection and analyze it according to various criteria.

1.1 Types of Plant Diseases

Plant diseases are usually caused by infectious organisms such as bacteria, viruses, and fungus. Symptoms are the outward signs associated with specific illnesses ,while signs are the detectable evidence of infection. Symptoms of fungal infections include leaf spot and yellowing, as well as visible spores, Mold, or mildew.

1.1.1 Bacteria Infection

Plant infections brought on by fungi are known as fungal diseases. Whether they are single- or multicellular, fungi infect plants by robbing them of nutrition and disintegrating their tissue. The most prevalent type of illness in plants is fungus-related. [14]Plants are susceptible to some distinct signs or visible consequences of the illness. Symptoms of fungal infections include spots on plant leaves, leaf yellowing, and berries with Birdseye spots. Along with some fungi as a growth and as a Mold.

Fig 1.1 Bacteria Infection

1.1.2 Fungal Infection

These could be deformities on the undersides of leaves or stems. These firsthand observations of the disease are referred to as infection symptoms. Bacteria are prokaryotic, single-celled creatures. Microbes are common and many of them might be advantageous, but some are cause illness in both plants and people. Bacterial symptoms are frequently more difficult to identify. compared to fungi, because bacteria are tiny. After making an contaminated stem, a possible milky white material known as "bacterial ooze." This is one sign that a bacterial infection is present. The other signs are moist, bacterially-filled areas on leaves that are saturated in water. As the illness worsens over time, the lesions eventually grow and provide the leaves with reddish-brown patches. Leaf spots are a common sign of bacterial illness. fruit stains. In contrast to fungal spots it contains veins on the leaf.

Fig 1.2 Bacteria Infection

1.1.3 Virus Infection

Infectious particles too small for a light microscope to detect are called viruses. They breakinto host cells and take control of host equipment to make the host millions of viral replicas. Since viruses don't exhibit symptoms in plants, Even with light, viruses cannotbe seen directly through a magnifying glass. However, there are indications that a virus infection is the cause of the mosaic leaf pattern and the yellowed or crumbled foliage that a traind eye can see. This traditional arrangement of the name of several plant viruses comes from discoloration like the virus known as tobacco mosaic. Moreover, a decline in plant Growth is frequently observed in viral infections as well.

Fig 1.3 Virus Infection

II. Literature Review

Paper 1:

Title: Plant disease detection using image processing and machine learning algorithm [15]
Publisher: Xidian University
Year: 2022
Dataset**:**Realtime Data

Summary**:** To extract the necessary information from the leaves, the k-means clustering technique is applied. Additionally, by acquiring a contrast image of the leaf and using neural networks, the characteristics of the leaf's diseased portion are improved. Each image is segmented using k-means clustering, which requires more time, on a sample collection of grouped images consisting of additional three cluster.
Accuracy : 89.8%

Paper 2

Title: Plant Infection Detection Using Image Processing [16]
Publisher: IjmerYear: 2018
Dataset: Plant Disease Database

Summary: The region of interest's (ROI) dimensions will be reduced in comparison to theoriginal picture. For texture analysis, the gray level co-occurrence matrix (GLCM) is among the finest techniques. Not It has a cheap learning cost, is simple to expand, supports dynamic neural networks, is easy to debug and modify, and is modular.
Accuracy : 98 % k-means and GLCM technique

Paper 3

Title: Plant diseases and pests detection based on deep learning [17]
Publisher: Springer Nature,Year: 2022
Dataset: Grape lead disease dataset
Summary: The traditional plant diseases and pests classification network is the same as the original picture classification method when there are more than two classes of plant diseases and pests to classify. The data obtained from the DCNN mode were used to classify nine distinct forms of rice illnesses.the accuracy achieved 97.5%.
Accuracy : 97.5 %

Paper 4

Title: Leaf Disease Detection using Image Processing [18]
Publisher: VIT university
Year: 2017
Dataset: Cotton leaf Dataset
Summary: we can boost crop output by using an adequate amount of pesticides to successfully control the pests, based on the amount of disease present in the leaf. By utilizing several segmentation and classification techniques, we can expand on this strategy.It takes a lot of time to extract features from RGB, HSV, YIQ, and dithered images.
Affected area prediction: 49.88 %

Paper 5

Title: Plant Disease Detection using CNN[19]

Dataset: Rice disease Image, Plant village dataset

Publisher: National College of Ireland

Year: 2020

Summary: Transfer Learning and RESNET 34 are used in this work. All levels are frozenby default in transfer learning, with the exception of the last two layers. These incorporate new weights and are specific to the plant disease classification task. By freezing, it is possible to train these layers independently of diseases without having to backpropagate the gradients. The latter layers are trained using the 1cycle policy in precisely this manner.The model did not show much benefit from augmentation, which helped CNN generalize more consistently.
Accuracy :97.2%

Paper 6
Title: Plant Disease Detection using Image Processing [20]
Year: 2020
Publisher: Ijert
Dataset: Plant village dataset

Summary: An end-to-end Android application with TFLite is part of the suggested system.The suggested method decided to create a plant disease detection Android app. Crop leaves are analyzed using Convolutional Neural Network models and algorithms to identify diseases and species. Bacteria can have more difficult-to-see symptoms than fungi becauseof their small size. If a sick stem is severed, a white substance may be visible.
Accuracy :90.34%

Paper 7

Title: Leaf Disease Detection Using Machine Learning [21]
Publisher: Journal of Seabold
Year: 2020
Dataset:Leafdataset
Summary: Each data point in an n-dimensional space is plotted in SVM, and the number of dimensions reflects the number of features being classified. Even though ongoing plant health monitoring and disease detection improve production quality and quantity, ML incurs costs each time.
Accuracy:88%

Paper 8

Title: Rice Leaf Disease Detection Using Machine Learning Techniques [22]
Publisher: International Conference on, Sustainable Technologies
Year: 2019
Summary: The accuracy of the algorithms' predictions regarding rice leaf diseases varied. Using test data, it was discovered that the decision tree had the best accuracy, scoring 97.9167%. Because the tree is constructed using information theory concepts like entropy and information gain, it takes longer.
F1 score : 96.5%.

III. Proposed Methodology

3.1 Methodology

The proposed methodology entails a sequential strategy to identify and diagnose diseases:

The images of plants that are exhibiting disease symptoms are obtained from the predetermined datasets of plants. These datasets were produced using various imaging technologies, such as cellphones or drones. The images are then used for data collection.

The gathered images are subjected to preprocessing, which consists of noise reduction, image enhancement, and segmentation, among other techniques, to distinguish between the areas that are healthy and those that are sick.

A deep learning model known as a MobileNet-v2 a convolutional neural network is trained using an image dataset that has already been preprocessed. MobileNet-v2 have demonstrated exceptional efficacy in completing image recognition tasks., and it will be finetuned for the classification of plant diseases.

A trained MobileNet-v2 is used to sort photos into healthy and disease-related groups If. the disease is present, the system will be able to recognize it in its specific form.

Real-time illness monitoring in the field is going to be made possible due to the integrationof the system with an intuitive user interface. When diseases are found, it will notify the appropriate parties so that action can be taken immediately

3.2 WorkFlow

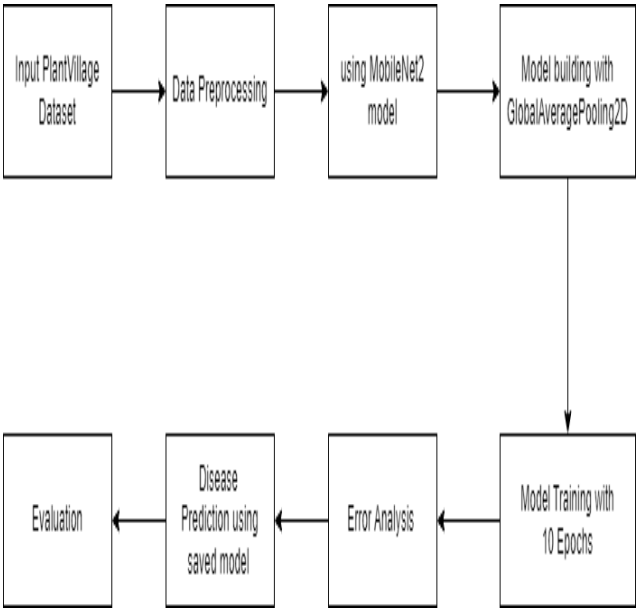


Fig 3.1 Work Flow

3.3 Data Set

This dataset contains below parameters
color: Original RGB images
grayscale: gray-scaled version of the raw images
segmented: RGB images with just the leaf segmented and color corrected.
In this Dataset it consists of 38 classes among which consist of images of types of plants:
1. Healthy 2. Infectious

3.4 Data Preparation

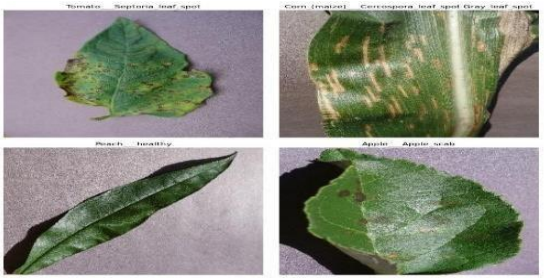
Here Dataset is prepared for plant leaf disease detection by first defining the directory path where the dataset is stored. It then iterates through each class folder within the dataset directory, retrieving the list of image files for each class. For eachimage file, the full path is constructed and appended to a list (`image_paths`), along with the corresponding class label, forming a structured dataset representation. Finally, this information is organized into a panda DataFrame (‘df’), containing two columns: 'image_path', storing the paths to individual images, and 'label', containing the corresponding class labels. This systematic approach streamlines subsequent data handling processes, facilitating efficient model training and evaluation.

	image_path	label
0	archive/plantvillage dataset/color\Apple___App...	Apple___Apple_scab
1	archive/plantvillage dataset/color\Apple___App...	Apple___Apple_scab
2	archive/plantvillage dataset/color\Apple___App...	Apple___Apple_scab
3	archive/plantvillage dataset/color\Apple___App...	Apple___Apple_scab
4	archive/plantvillage dataset/color\Apple___App...	Apple___Apple_scab

Fig 3.2 Data Preparation

3.5 Data Analysis

The creation of a captivating grid showcasing sample images for visual exploration, spotlighting a representative image from each distinct plant class present in the dataset. Commencing with the determination of the number of unique classes (num_classes) and the calculation of the grid layout based on the designated number of images per row (num_images_per_row), the figure's size dynamically adjusts to accommodate the computed number of rows. Through a meticulously crafted loop iterating over each unique plant class, a bespoke subplot emerges, poised to unveil a sample image.As the path to the inaugural image affiliated with the current class is unveiled from the DataFrame, OpenCV (cv2) steps in to summon the image into existence. Should the imagemanifest successfully, it graces the grid with its presence, crowned with its corresponding class label serving as a regal title. Any misadventures encountered along the path of image loading or retrieval are dutifully recorded for posterity. Ultimately, this mosaic of imagerypaints a vivid portrait of the diverse plant classes residing within the dataset, inviting observers to delve into the tapestry of botanical diversity and unearth the nuances embedded within.



3.3 Sample Images

3.5.2 Plant Leaf Analysis

The visualization of class distribution within the dataset through a horizontal bar plot. . Initially, it computes the occurrence count for each unique class label in the DataFrame utilizing the `value_counts () ` method. Subsequently, these class counts are depicted as horizontal bars, where the different plant species are indicated by the y-axis and the number of photos is represented by the x-axis.Additionally, data labels denoting the count of images for each class are displayed at the end of each respective bar. This visual representation furnishes a comprehensive overview of the dataset's class distribution, thereby accentuating any potential class imbalances that could impact model training and subsequent performance evaluation.

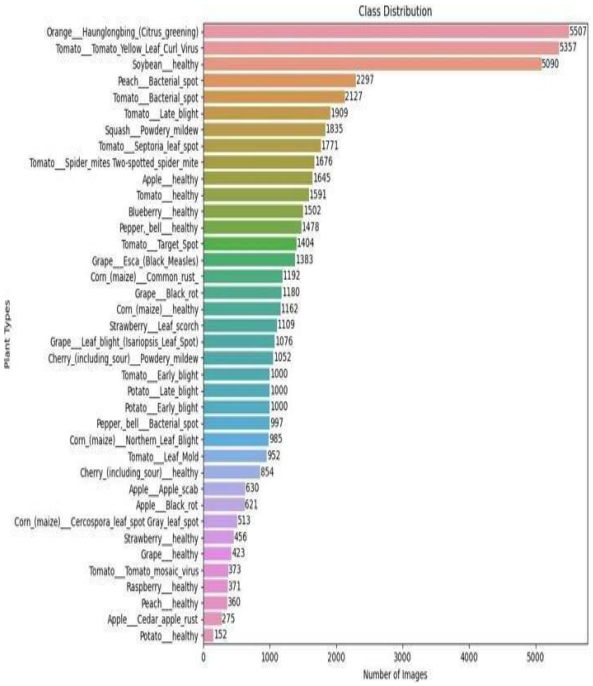


Fig 3.4 Plant Leaf analysis

3.5.3 Balancing The Dataset

The dataset exhibits a class imbalance, as seen in Figure 3.4.Thus, we use an oversampling method to our dataset. We created the balance_dataset function to make sure every class has the same amount of samples. In order to resample the minority classes with replacement to match the maximum class size, this function first determines the maximum class size. The balanced subsets are joined and shuffled after oversampling to guarantee a random

distribution. By effectively mitigating bias towards the majority class, this method improves the generalization capacity of the model. As such, this step is essential to enhancing on model's prediction performance on minority classes.

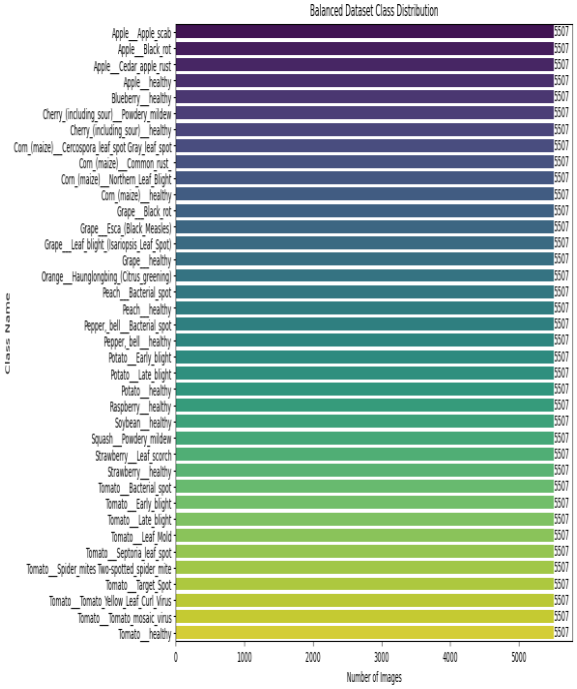


Fig 3.5 Balanced Dataset

3.6 Splitting Data into Training and Testing

The provided code segment orchestrates the preparatory steps vital for training the dataset, meticulously encoding class labels into numerical values and dividing it into test, validation, and training sets using scikit-learn's flexible train_test_split method. First, a fakedictionary called class_labels_dict is created, acting as the conduit to map each unique class label to a corresponding numerical index. Subsequently, the 'label' column in the DataFrame undergoes a transformative journey, shedding its textual guise and embracing anumerical identity through the seamless substitution of class labels with their numerical counterparts. The dataset then embarks on a voyage of division, traversing the realm of training, validation, and test sets, guided by the steady hand of train_test_split. The datasetin this adventure is split harmoniously into training and test sets in an 80:20 ratio, providing the foundation for further investigation. Further division ensues, as the training set is partitioned anew into training and validation cohorts, fostering an environment conducive to model refinement and validation. Finally, the class labels in each partition metamorphose into strings, unveiled in a ritualistic display, offering a glimpse into the diverse tapestry ofunique class labels adorning the training set. This meticulous preprocessing ritual not onlyorchestrates the harmonious division of the dataset but also ensures its compatibility with the discerning palates of machine learning algorithms, thus paving the way for fruitful model training, validation, and evaluation endeavors.

3.7 Data Augmentation and Generator Setup

To augment the training data with diverse transformations, an ImageDataGenerator named `train_datagen` is defined. This generator facilitates various augmentations such as rotation, shifting, shearing, zooming, and horizontal flipping, thereby enriching the dataset and enhancing model robustness. Utilizing the `flow_from_dataframe` method, batches of training data (referred to as `train_generator`) are generated from the DataFrame `train_df`, with specific configurations set to ensure optimal training conditions. These configurations include setting the target image size to (224, 224), defining a batch size of 40, and specifying the class mode as 'categorical', aligning with the categorical nature of the labels. Similarly, generators (`val_generator` and `test_generator`) are established for validation and test data, albeit with minimal augmentation to preserve the integrity of the validation and evaluation processes. Additionally, pixel values are rescaled to the range [0, 1], ensuring consistency across datasets and facilitating seamless integration with machine learning models. Through these meticulously crafted generators, the dataset is primed for training, validation, and evaluation, embodying a holistic approach toward robust model development and performance assessment.

3.8 Model Selection and Building

Here, the MobileNetV2 architecture, a lightweight convolutional neural network (CNN) pre-trained on ImageNet, as the base model for transfer learning. By leveraging pre-trained weights, the model benefits from features learned on a large-scale image dataset, enabling efficient extraction of relevant features from leaf disease images. The addition of Global Average Pooling and dense layers facilitates the adaptation of the base model to the specific task of classifying leaf diseases. Multi-class classification is made possible by the model's generation of probability distributions over the disease classes using the softmax activation function in the output layer.

In this configuration, the process commences with the instantiation of MobileNetV2 as thebase model, adorned with pre-trained weights sourced from ImageNet. The top classification layers are

deliberately excluded, thereby enabling the incorporation of customclassification layers tailored to the specific task at hand. Subsequently, the number of classes (referred to as `num_classes`) is determined based on the length of the dictionary housing class labels (`class_labels_dict`), ensuring alignment between the model architecture and the dataset's categorical structure. Custom classification layers are then meticulously crafted and added atop the base model, commencing with a GlobalAveragePooling2D layer for spatial data reduction, a Dense layer with 1024 units that uses ReLU activation to add non-linearity and encourage feature expression, comes next. Finally, a Dense layer is appended with softmax activation, facilitating multi-class classification by producing probability distributions across the various classes. The model is artfully constructed using the Model class, seamlessly melding the input from the base model with the output from the custom classification layers, thereby culminating in a unified architecture poised to embark on the journey of training and inference.

3.9 Model Compilation

Following the model construction, the next imperative task is to compile it, thereby configuring its optimization strategy, loss function, and evaluation metrics. In this endeavor, the Adam optimizer is chosen, boasting adaptive learning rates and robust performance across various scenarios. With a learning rate of 0.001, the optimizer navigates the optimization landscape, steering the model towards convergence with efficiency. Categorical cross-entropy is designated as the loss function, ideally suited for multi-class classification tasks, while accuracy emerges as the metric of choice, providing insight into the model's classification prowess. With these configurations meticulously crafted, the model stands poised for training, equipped with the tools necessary to navigate the intricacies of the dataset and refine its parameters towards achieving optimal

3.10 Model Training

With a holistic approach towards model training, the number of epochs is explicitly defined, set at 10 to ensure a comprehensive exploration of the dataset and facilitate convergence towards optimal performance. The model embarks on its training journey by fitting to the training data (`train_generator`), iteratively refining its parameters over the specified number of epochs. Concurrently, the validation data (`val_generator`) is leveraged to monitor the model's performance during training, safeguarding against overfitting and providing insights into its generalization capabilities. The training history, encapsulating crucial metrics such as training/validation loss and accuracy, is meticulously recorded andstored within the `history` variable. This treasure trove of data serves as a foundation for comprehensive analysis and visualization, offering valuable insights into the model's progression and facilitating informed decisions throughout the development process. By adhering to this structured approach, the groundwork is laid for the creation of a robust and effective plant leaf disease detection system, poised to deliver accurate diagnoses and contribute towards mitigating agricultural challenges.

3.11 Hyper Parameter Tunning

The code snippet performs hyperparameter tuning specifically targeting the fine-tuning of the model's convolutional layers. By setting the last 40 layers of the model as trainable, the code allows for more focused optimization on these layers during training. This approach is often employed in transfer learning scenarios where the initial layers of a pre-trained model capture generic features like edges and textures, while the later layers learn more task-specific features. By fine-tuning only a portion of the model's layers, computational resources are conserved while still enabling the adaptation of the model to the specific taskof classifying leaf diseases.

The learning rate for fine-tuning is set to a small value of 0.00001, with the goal of carefully modifying the trainable layers' weights in order to reduce the possibility of deviating from the previously learnt features..A smaller learning rate is typically used during fine-tuning to ensure more stable convergence and prevent catastrophic forgetting of previously learned features. The categorical cross-entropy loss function, which is widely used for multi-class classification tasks, is then used to create the model and the Adam optimizer with the given learning rate for fine-tuning. Furthermore, accuracy is measured as a statistic to trackthe model's performance throughout training.

Finally, the model is trained for 10 epochs on the fine-tuning dataset, which is typically a subset of the original training data. This limited number of epochs helps prevent overfitting and allows the model to further refine its parameters to better discriminate between different leaf disease classes. Overall, this hyperparameter tuning strategy aims to strike a balance between leveraging the pre-trained features and fine-tuning the model to achieve optimal performance on the specific task of leaf disease classification.

3.12 Fine Tuning Set up

In the process of fine-tuning the model, a crucial step involves defining a variable `n` to specify the number of layers to freeze. This strategic maneuver is

instrumental in mitigating overfitting while safeguarding the invaluable features gleaned by the pre-trained layers. By iteratively traversing through the layers of the model, excluding the last `n` layers, a deliberate decision is made to set `trainable` to True for each layer. This nuanced approach empowers these selected layers to undergo updates during the fine-tuning process, while simultaneously preserving the integrity of the weights in the remaining layers, ensuring that the foundational features learned by earlier layers remain unaltered. Through this meticulous orchestration of selective unfreezing, the model embarks on a journey of refinement, delicately balancing the incorporation of new insights with the preservation of established knowledge, thereby facilitating the attainment of optimal performance and generalization capabilities.

3.13 Adjust Learning Rate

In the intricate dance of fine-tuning, a critical aspect involves the calibration of a new learning rate (`learning_rate_finetune`). Unlike the initial training phase, where a relativelyhigher learning rate may be employed to facilitate rapid convergence, fine-tuning necessitates a more delicate approach. This new learning rate is typically set to a smaller magnitude, serving as a stabilizing force to prevent drastic alterations to the pre-trained weights and ensuring a smoother trajectory during the fine-tuning process. By embracing this nuanced adjustment, the model embarks on a journey of refinement with a tempered pace, delicately navigating the optimization landscape to seamlessly integrate new insights while preserving the integrity of the foundational knowledge gleaned from pre-training. Through the harmonious interplay of meticulous adjustments and strategic considerations, the model emerges rejuvenated, poised to elevate its performance and adaptability to the intricacies of the target task.

3.14 Model Compilation for Fine Tunning

In the final stage of model preparation for fine-tuning, a critical step involves recompilingthe model with the updated learning rate tailored for the fine-tuning phase. This recalibration ensures that the optimization process aligns with the nuanced requirements offine-tuning, where a smaller learning rate is imperative to facilitate stable convergence while preserving the integrity of pre-trained weights. To this end, The model is carefully configured with the modified learning rate and assembled using the Adam optimizer. This optimizer, renowned for its adaptive learning rate mechanisms, navigates the optimization landscape with precision, ensuring smooth fine-tuning without jeopardizing the pre-learned features. Furthermore, categorical cross-entropy is designated as the loss function, ideal for multi-class classification tasks, while accuracy emerges as the metric of choice to gauge the model's classification prowess. Through this judicious orchestration of model compilation, the stage is set for fine-tuning, as the model stands poised to refine its parameters with a measured stride, elevating its performance and adaptability to the intricacies of the target task.

3.15 Fine-Tuning Process

In the pivotal phase of fine-tuning, it is imperative to define the number of epochs (`epochs_finetune`) to guide the duration of this refinement process. This parameter dictates the extent to which the model is allowed to adapt and specialize to the intricacies of the specific task of plant leaf disease detection. Once established, The training data is used to fit the model. (`train_generator`) over the specified number of epochs, embarking on a journey of refinement and adaptation. Throughout this process, the validation data (`val_generator`) serves as a steadfast companion, diligently monitoring the model's performance and safeguarding against the perils of overfitting. By scrutinizing the interplay between training and validation metrics, the model navigates the optimization landscape with poise and precision, striking a delicate balance between exploration and exploitation. The fine-tuning training history, encapsulating crucial metrics such as training/validation loss and accuracy, is meticulously recorded and stored within the `history_finetune` variable. This repository of insights serves as a cornerstone for further analysis and visualization, illuminating the model's progression and evolution as it acclimates to the demands of the target task. Through this iterative process of refinement, the model harnesses the synergy between pre-rained features and newly acquired representations, culminating in a robust and specialized architecture primed for the nuanced challenges of plant leaf disease detection

3.16 Model Evaluation

To comprehensively assess the model's performance on unseen test data, we leverage the`evaluate` method, a quintessential tool for evaluating models on data that hasn't been encountered during training or validation. By invoking this method with the test data generator (`test_generator`), the model undergoes rigorous scrutiny, enabling the computation of both test loss and accuracy. These crucial performance metrics provide invaluable insights into the model's generalization capabilities and efficacy in real-world scenarios. The computed test loss and accuracy are then dutifully stored in the variables `test_loss` and `test_accuracy`, respectively, serving as a testament to the model's prowess and reliability in the face of novel challenges. Through this meticulous evaluation process, the model's true capabilities are unveiled,

empowering stakeholders with the confidence to deploy it in practical applications with unwavering assurance.

IV Result

4.1 Visualization of Training and Validation

Metrics Visualizing the trajectory of training and validation accuracy over epochs offers invaluable insights into the model's learning dynamics, enabling a comprehensive assessment of its progress and performance. By plotting these metrics, one gains a nuanced understanding of how the model's accuracy evolves throughout the training process. Such visual scrutiny is pivotal in discerning potential signs of overfitting or underfitting, as deviations between the training and validation accuracy curves may indicate the presence of such phenomena. Thus, through this graphical representation, stakeholders can effectively monitor the model's learning journey, identify areas of improvement, and iteratively refine the trainingstrategy to achieve optimal performance and generalization capabilities.

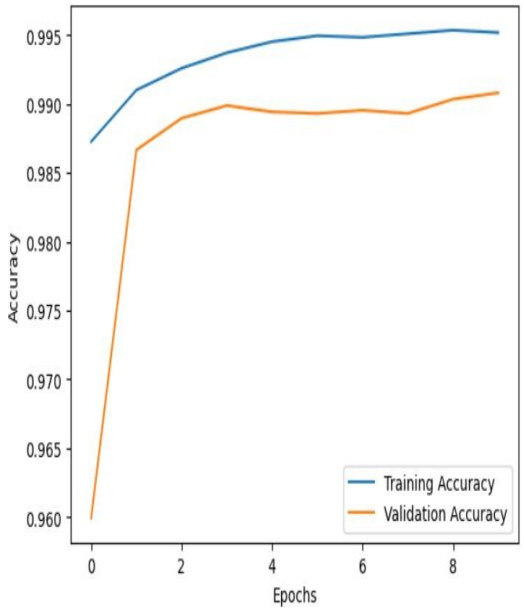


Fig 4.1 Training and Validation Accuracy

The visualization of training and validation loss over epochs offers a nuanced perspective on the model's error minimization and generalization prowess throughout the training process. By plotting these loss metrics, stakeholders get insightful knowledge about the effectivenessof the model's learning strategy and its ability to navigate the optimization landscape. This graphical representation serves as a compass, guiding the assessment of how effectively the model is converging towards minimizing its error while simultaneouslygauging its ability to generalize to unseen data. Fluctuations and trends observed in the loss curves provide actionable insights into the model's performance trajectory, allowing for informed adjustments to training strategies and architecture design. As such, this visual analysis serves as a cornerstone in the iterative refinement of models, facilitating the attainment of optimal performance and robust generalization capabilities.

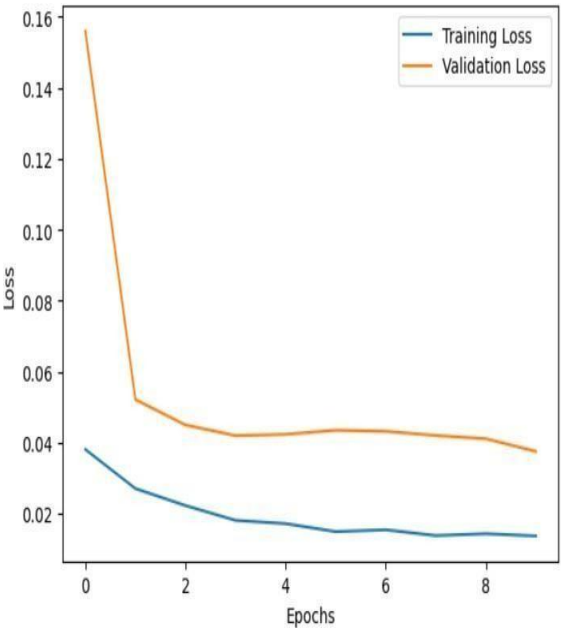


Fig 4.1 Training and Validation Loss

4.2 Confusion Matrix

The confusion matrix, also called the error matrix, is one of the most crucial instruments for assessing a classification model's performance. It offers a thorough analysis of the model's predictions and how well they match the available empirical data.

1. Actual Classes: The plant diseases included in the collection are identified by these actual labels. Every class corresponds to a certain plant disease, like apple Apple_scab, Blueberry well, and so forth.

2. Predicted Classes: The model uses these labels to categorize each input image in an effort to determine which illness is depicted in the plant image.

There are 38 classes in the dataset, and the confusion matrix is 38 by 38. The matrix's cells each indicate the number of times the actual class and the

anticipated.this 38 classes consist of both healthy and healthy leaves of the plant.

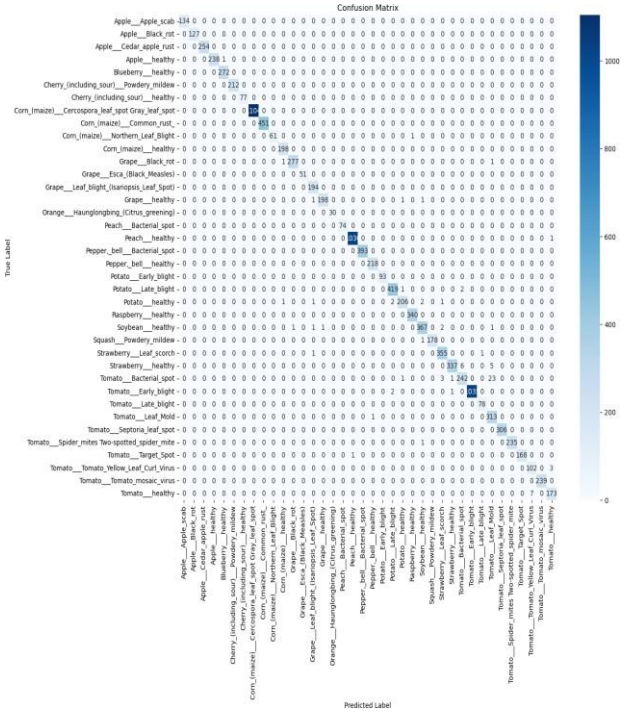


Fig 4.3 Confusion Matrix

From the above confusion matrix we can calculate True positive, False Positive , False Negative and True Negative

- True Positive: The values that can be derived from the confusion matrix's diagonal are known as True Positives.
- False Positive: The confusion matrix's TP value is absent, and the associated column's sum is what counts.
- False Negative: The confusion matrix's TP value is absent from the sum of the relevant columns.
- True Negative: The total of all columns' and rows' values, excluding the values for the class for which the values are being calculated.

4.3 ROC Curve

Plotting the True Positive Rate (TPR) on the y-axis and the False Positive Rate (FPR) on the x-axis, the ROC curve assesses the performance of the model. Every ROC curve for our 38 classes displays an AUC of 1.00, signifying flawless categorization. This indicates that the top left corner of each class's ROC curve is reached (TPR of 1, FPR of 0). The diagonal line from (0,0) to (1,1) in the plot represents a random guessing baseline within these 38 classes. Its AUC of 0.50 indicates that it has no discriminative potential. The model's exceptional ability to differentiate between positive and negative instances for each class is demonstrated by its flawless AUCs, highlighting its potential for precise diagnosis of plant diseases in agriculture.

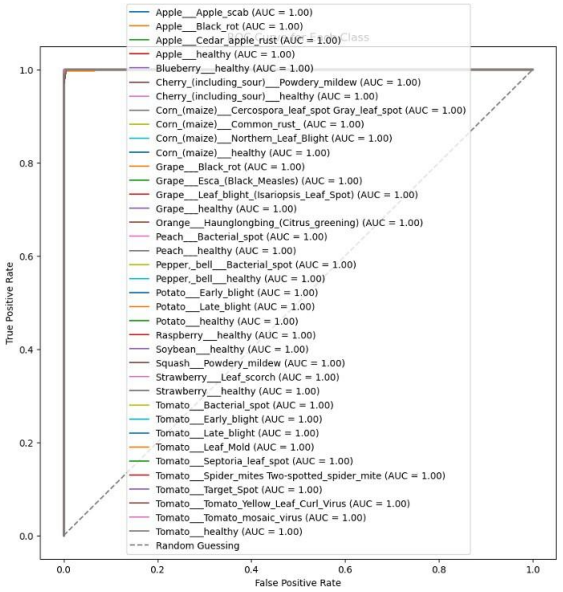


Fig 4.4 ROC Curve

4.4 Model Comparision

In this work, we evaluate three convolutional neural network (CNN) architectures on a plant disease classification task: MobileNetV2, ResNet50, and InceptionV3. Recall, f1-score, accuracy, and precision are among the evaluation measures.

Model	Accuracy	Precision	Recall	F1-score
MobileNet V2	99.00	99.00	99.00	99.00
Resnet50	94.23	92.30	94.25	93.26
Inception V3	94.89	95.20	94.12	94.65

Table 4.1 Model Comparison

MobileNetV2

MobileNet V2 outperformed all other models in terms of accuracy, precision, recall, and f1-score, with 99.00%. This suggests that MobileNetV2 is a strong model for this classification task and that it is very successful at correctly recognizing plant illnesses. High accuracy and efficiency are provided by the model's lightweight architecture, which is tailored for embedded and mobile vision applications.

ResNet50

ResNet50 obtained lesser metrics than MobileNetV2, but it was still performing well: 94.23% accuracy, 92.30% precision, 94.25% recall,

and 93.26% f1-score. Although ResNet50's deeper architecture, which makes use of residual connections to address the vanishing gradient issue, offers good performance, MobileNetV2's accuracy is superior for this particular task.

Inception V3
With accuracy of 94.89%, precision of 95.20%, recall of 94.12%, and f1-score of 94.65%, InceptionV3 displayed a well-rounded performance. InceptionV3, which uses inception modules to record multi-scale characteristics and is well-known for its intricate design, scores marginally worse overall than MobileNetV2 but beats ResNet50 in terms of precision and f1-score.

V Conclusion

This research proposal that focuses on utilizing image processing and MobileNet-v2 to detect plant diseases presents a potentially useful solution to the difficulties that are causedby disease outbreaks in the agricultural industry. The research helps to reduce crop losses and ensures that food security is maintained by contributing to the creation of an automated system with real-time illness detection capabilities. It is anticipated that the suggested methodology, which combines conventional MobileNet-v2 analysis with image processing, will produce findings that are both accurate and effective. This study demonstrates the potential for technology to revolutionize agriculture and to alleviate the limits of conventional techniques of disease detection.

This research presents a novel approach to the identification of plant illnesses, which has the potential to bring about a sea change in agricultural practices, as well as to contribute to the safety of food supplies and the viability of the economy. It establishes the groundwork for additional developments in the industry, solving the issues faced by plant diseases and providing a brighter future for agriculture as a whole.

By employing the MobileNetV2 architecture as a base model and fine-tuning its convolutional layers, we have developed a robust classifier capable of accurately identifying various leaf diseases from images. The utilization of transfer learning allowed us to leverage pre-trained features from ImageNet, lessening the computing burden and improving the model's capacity to extrapolate to unobserved data. We adjusted the model's hyperparameters to better suit the particular goal of classifying leaf diseases,optimizing its performance with careful adjustment of learning rates and layer trainability.

The outcomes of this research demonstrate the potential for AI-driven solutions to revolutionize disease diagnosis and management in agriculture. With the trained model, farmers and agricultural experts can swiftly and accurately identify leaf diseases, enabling timely intervention and treatment to mitigate crop losses. Moreover, the automated classification system facilitates early detection of diseases, allowing for proactive measures to prevent their spread and minimize economic losses. Additionally, the scalability and adaptability of the model make it appropriate for use in a range of agricultural environments, including large-scale plantations and small-scale farms.

Overall, this research highlights the transformative impact of deep learning in agriculture, offering a potent instrument for boosting crop health, raising output, and guaranteeing food security.

VI References

1] Rong Zhou, Shunichi Kaneko, Fumio Tanaka, Miyuki Kayamori, Motoshige Shimizu,"Disease detection of Cercospora Leaf Spot in sugar beet by robust template matching," *Computers and Electronics in Agriculture, 2014,*108, 58-70.

2] Jayme Garcia Arnal Barbedo, "A review on the main challenges in automatic plant disease identification based on visible range images," *Biosystems Engineering,*2015, 144,52-60.

3] D. Al Bashish, M. Braik and S. Bani-Ahmad, "A Framework for Detection and Classification of Plant Leaf and Stem Diseases," in *International Conference on Signal and Image Processing*, 2010.

4] Jagadeesh D. Pujari, Rajesh Yakkundimath, Abdulmunaf S. Byadg, "Image Processing Based Detection of Fungal Diseases In Plants," in *International Conference on Information and Communication*, 2015.

6] C. Zhang, X. Wang and X. Li, "Design of Monitoring and Control Plant Disease System Based on DSP&FPGA," in Second International Conference on Networks Security, Wireless Communications and TrustedComputing, 2010.

7] Elham Omrani, Benyamin Khoshnevisan, Shahaboddin Shamshirband, Hadi Saboohi, Nor Badrul Anuar, Mohd Hairul Nizam Md Nasir, "Potential of radial basis function based support vector regression

for apple disease detection," Journal of Measurement, 2014, 233-252.

8] H. Hashim, M. A. Haron, F. N. Osman and S. A. M. Al Junid, "Classification of Rubber Tree Leaf Disease Using Spectrometer," IEEE,2010 ,1-8.

9] Kumar, Surender & Kaur, Rupinder, "Plant disease detection using image processing- a review," International Journal of Computer Applications,2015, 124, 1-16.

10] Vijai Singh, A.K. Misra, "Detection of Plant Leaf Diseases Using Image Segmentation and Soft Computing Techniques," Information Processing in Agriculture,2017 ,4(10), 41-49.

11] Wang Jun and Wang Shitong "Image thresholding using weighted parzen window estimation.," Journal of Applied Sciences, vol. 8, no. 5, pp. 772-779, 2008.

12] Trimi Neha Tete, Sushma Kamlu, "Detection of Plant Disease Using Threshold, K- Mean Cluster and.," World Congress on Computing and CommunicationTechnologies., 2014, 35-37, 2014.

13] Sabrol, Hiteshwari & Kumar, Satish, "Recognition of Tomato Late Blight by using DWT and Component Analysis," in IEEE Conference, 2017.

14] S. D. Khirade and A. B. Patil, , "Plant disease detection using image processing," International Conference on Computing Communication Control and automation, 2015, 768-771.

15] N, Banupriya & chowdry, Rajaneni & Yogeshwari, & V, Varsha,, "Plant disease detection using image processing and machine learning algorithm.," 10.37896/jxu14.7/012 , 2022.

16] Meyyappan, Senthilkumar & Chandramouleeswaran, Sridhathan, "Plant Infection Detection Using Image Processing.," International Journal of Modern Engineering Research (IJMER),2018, 8, . 2249-6645.

17] Liu, J., Wang, X. Plant diseases and pests " Plant diseases and pests detection based on deep learning: a review," Plant Methods, 17, 1-18.

18] Radha, Suja. , "Leaf Disease Detection using Image Processing.," Journal of Chemical and Pharmaceutical Sciences..

19] E. Harte, "Plant Diseases detection using CNN ," 10.13140/RG.2.2.36485.99048 . , 2020.

20] Suresh, V & Krishnan, Mohana & Hemavarthini, M & Jayanthan, D, "Plant Disease Detection using Image Processing," INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY, 2020,9.

21] A. S. Tulshan and N. Raul, "Leaf Disease Detection Using Machine Learning," 2022 15, 1828-1832.

22] Ahmed, Kawcher & Shahidi, Tasmia & Irfanul Alam, Syed & Momen, Sifat. Ahmed, "Rice Leaf Disease Detection Using Machine LearningTechniques,"10.1109/STI47673.2019.9068096,2019,1-5.