BANANA LEAF DISEASE DETECTION USING HYBRID DEEP LEARNING MODEL

A PROJECT REPORT

Submitted by

ADITYA S NAIR (720419104001)

PANDIARAJ M (720419104019)

SANJAY R J (720419104022)

VINAYAK P K (720419104031)

in partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

CMS COLLEGE ENGINEERING AND TECHNOLOGY

COIMBATORE

ANNA UNIVERSITY : CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY : CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that the project report "**BANANA LEAF DISEASE DETECTION USING DEEP LEARNING HYBRID MODEL**" is the bonafide work of "**ADITYA S NAIR, PANDIARAJ M, SANJAY R J, VINAYAK P K**" who carried out the project work under my supervision.

SIGNATURE
Dr. G. Chitra Ganapathy, M.E., Ph.D.,
Professor
Head of the Department
Department of Computer Science and Engineering
CMS College of Engineering and Technology
Coimbatore-641032.

SIGNATURE
Mr. S. Prabakaran , M.Tech., (Ph.D).,
Assistant Professor
Supervisor
Department of Electronics Communication and Engineering
CMS College of Engineering and Technology
Coimbatore-641032.

Submitted for Viva Voce examination held on ……

Internal Examiner External Examiner

ACKNOWLEDGEMENT

We thank our Management Trustees of **CMS EDUCATIONAL AND CHARITABLE TRUST COIMBATORE** for providing us all the facilities required to complete the project work.

We record our indebtedness to our **Principal Dr.N.Sudha,M.E.,Ph.D** for her valued guidance and sustained encouragement for the successful completion of this project work.

.

We express our heartiest thanks to **Dr.G.Chitra Ganapathy,M.E.,Ph.D, Professor** and Head of Department of Computer Science and Engineering ,CMS College of Engineering and Technology, for her encouragement and valuable guidance in carrying out our project work.

We express our heartfelt gratitude to our Project Supervisor **Mr S Prabakaran, M.Tech.,(Ph.D.), Assistant Professor**, CMS College of Engineering and Technology, Coimbatore, for his valuable and timely support for our project work.

We express our heartfelt thanks to Project Co-ordinator **Mr.S.Dinesh Kumar, M.E.,Ph.D, Assistant Professor**, CMS College of Engineering and Technology, Coimbatore, for his valuable and timely support for our project work.

We also express thanks to our parents, friends for their encouragement and best wishes in the successful completion of this dissertation.

ABSTRACT

In India, the agricultural sector supports over 70% of the population. One of India's most important agricultural activities is banana growing, however the sector faces challenges due to the impact of pests and diseases. The insects that carry the infection from plant to plant have damaged the various parts of the plants. Early detection of these threats is essential to limiting farmers' financial losses. The national economy and overall banana output will be negatively impacted by this issue. Therefore, early detection of these illnesses is crucial. Banana disease detection on the plantation has been more difficult than expected. It is crucial for farmers to assess the plant health without expensive labor, by employing advanced technologies to detect diseases on banana leaves. In order to classify diseases affecting banana leaves, the research propose a hybrid Deep Learning (DL) model for banana leaf disease classification using Convolutional Neural Network (CNN) and Long Short Term Memory (LSTM) called CNN-LSTM. The data set was gathered from the Mendeley database. CNN was used to extract the features, while LSTM was used to classify the images. Next, we develop a standalone CNN model and evaluate it against CNN-LSTM for classifying banana leaf diseases. According to the simulation results, CNN-LSTM will perform more accurately than the CNN model. To prevent further disease spread, the hybrid model's early detection will allow farmers to use pesticides.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

List of Abbreviations

ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
LSTM	Long Short Term Memory
GDP	Gross Domestic Product
WHO	World Health Organization
SVM	Support Vector Machine
NN	Neural Network
SSD	Single-Shot Detector
BXW	Banana Xanthomonas Wilt
GLCM	Gray-Level Co-Occurrence Matrix

CHAPTER 1

INTRODUCTION

Agriculture is critical to human survival because it is one of the key contributors to a country's GDP. Most poor countries rely on agricultural exports to supplement their national income. Banana cultivation and manufacturing have become a crucial element of the global Agro business since they are a good source of essential nutrients such as manganese, magnesium, potassium, calcium, and iron. People all around the globe consume bananas because of their reputation as an immediate source of energy owing to the crop's high vitamin content. According to Wikipedia, Western countries import approximately 15% of the global banana harvest. India accounts for roughly a quarter of the world's total banana output, according to data on banana exports and domestic production. Other significant banana-producing countries include the Philippines, Indonesia, Ecuador, and Brazil, which account for another 20% of total production. The United States is the world's largest importer of bananas, accounting for more than 18% of global supply.

Over 130 countries, typically in warmer climates, cultivate bananas for human consumption. Southeast Asia is credited as the place of their birth [13]. Bananas are a popular staple meal and the world's second most extensively cultivated fruit, behind citrus. Bananas rank fifth on the list of the most traded commodities, below coffee, cereals, sugar, and cocoa. Bananas can be utilized in two ways: as a dessert item or as a culinary essential. Raw or cooked, they benefit human health thanks to the presence of bioactive compounds such as phenolics, biogenic amines, carotenoids, and phytosterols. They are also rich in antioxidants. Historically, bananas have been utilized to alleviate the signs of some

degenerative diseases. A daily fruit and vegetable consumption of 400 grams is advised by the WHO. Bananas are farmed in a variety of locales to meet global demand. India is the world leader in banana agriculture, accounting for 27% of global banana production. The world's banana plantations cover 5,517,027 hectares and produce a total of 128,778,738 tons. Bangladesh's banana plantations cover 48,850 hectares and produce 833,309 metric tons of fruit [17].

The Banana's Importance in Nutrition Banana peel: The banana peel is frequently tossed carelessly into the land or water, which can have harmful environmental consequences. Because of the various vitamins and nutrients they contain, banana peels are an excellent source of feed for cattle due to their high carbohydrate content and low protein level. Banana peel can be utilized in the fermentation process to produce ethanol, citric acid, and other bioactive substances at a low cost.

A fully ripe bananas are an important part of many third-world and poor countries' diets since they are high in nutrients. Ripe bananas have numerous culinary applications. Bananas are a highly nutritious dietary choice due to their high caloric content, tissue-building components, vitamins, protein, and minerals. This food contains enough levels of calories, numerous nutrients, and enzymes. Bananas are an important part of the inhabitants of southern India's healthy, varied, and supplemented diet. As a result, it's a superb nutritional breakfast that's easy on the stomach and helps to prevent diarrhea and worms [11].

Banana plants are vulnerable to a wide range of diseases. Because of their varied pathogen origins, leaf spot diseases are the major cause of crop losses worldwide. Several frequently identified leaf spot diseases involve Sigatoka diseases like Yellow Sigatoka (Pseudocercospora Musicola), Black Sigatoka (Mycosphaerella Fijiensis), Exserohilum Leaf Spot (Exserohilum Rostratum),

Cordana Leaf Spot (Cordana Musae), Banana Freckle Disease (Phyllosticta Musarum), Eumusae Leaf Spot (Mycosphaerella Eumusae). The banana farms have been hit by a number of diseases. The illness is visible on the leaves, stems, flowers, fruits, roots, and suckers. The two most frequent leaf diseases are Xanthomonas and Sigatoka.

Plant-borne diseases drastically reduce banana harvests. Depending on the severity of the condition, this could cut yield in half. It is essential to evaluate the severity of the illness in order to devise control measures that will prevent the progression of the disease. Because the infection first appears as leaf specks, it is normally treated by spraying three times at 10-15 day intervals with carbendazim, propicanozole, or mancozeb and tempol in the correct mix. As a result, it is critical to detect the pest or disease as soon as feasible. Traditional procedures for pest and disease identification are hampered by human ignorance of agricultural science. The increased use of digital cameras and computers in agriculture has resulted in the widespread use of automated disease diagnosis models. Recently, DL-powered computer image processing has been used to the problem of evaluating the severity of plant diseases [4].

1.1 MOTIVATION

Traditional and non-scientific techniques of identifying the best crop for a farmer's land are a major issue in a country where farming employs around half of the population. Academics find it challenging to perform case studies in developing countries due to a lack of precise and up-to-date information. The authors of each of these key published works concentrated on a design model for forecasting crop development adaptability, which we discovered to be a shortcoming. However, we do not believe it will reach the farms.

1.2 TASK UNDERTAKEN

- Collect crop recommendation dataset from a trustable website
- Done pre-processing and visual analysis of crop recommendation dataset
- Develop an ML model to predict suitable crops from the relevant features
- Identify the best model to predict suitable crops by using classification metrics.
- Create a mobile app using the best model

1.3 AIM/OBJECTIVE OF THE RESEARCH:

- Crop sustainability modeling in a given state for the unique soil type and climate conditions is a key step in constructing a trustworthy and accurate model.
- Selecting the best crops for the site, ensure that the farmer does not lose money.
- Provide a profit analysis for various crops based on the previous year's data.

1.4 SCOPE OF THE RESEARCH:

- Farmers no need to wait for an expert suggestion
- Farmer identify their suitable crop with a single click on mobile
- Farmer’s economic levels will increase.

CHAPTER 2

LITERATURE SURVEY

Chapter 2 discusses the recent research work on banana leaf disease detection using image processing, ML, and DL technology.

TITLE 1: An automated segmentation and classification model for banana leaf disease detection [7]

YEAR OF PUBLICATION: 2022

For the purpose of automating the classification of banana leaf diseases, this research introduces a method for segmenting images. Banana plant diseases are identified and categorized based on the images. Cost-effective and efficient plant health monitoring is now available to farmers. Segmenting images is essential for analysis and data extraction. This section of image processing separates the object of study from its background so that it can be examined in more detail. The success of subsequent image processing steps depends on the accuracy of the initial image segmentation steps. The data is divided and sorted using a hybrid fuzzy C-means method. Banana plant characteristics such as color, shape, and texture were also obtained for use in making diagnoses of banana plant diseases. Several quantitative markers were researched to compare the proposed method to existing DL algorithms for diseases such as Sigatoka, dry leaves, and bacterial wilt, in comparison to healthy plants.

TITLE 2: **Banana Leaf Disease Detection Using GLCM-Based Feature Extraction and Classification Using Deep Convoluted Neural Networks (DCNN) [8]**

YEAR OF PUBLICATION: 2022

In this paper, they offer a feature extraction method for detecting leaf diseases that employ a gray-level co-occurrence matrix (GLCM) and a DCNN. A database of agricultural records was used to compile and analyze the data set. GLCM was used to extract the features, while DCNN was used to classify the image. Then, classify the collected features of the disease-affected area with GLCM-DCNN to enhance the accuracy of illness identification. The suggested method is shown to have better performance ratings in simulations. With this technique, farmers will be able to detect diseases early and use pesticides to stop their spread.

TITLE 3: Research on Banana Leaf Disease Detection Based on the Image Processing Technology [6]

YEAR OF PUBLICATION: 2022

Banana quality and yield were adversely affected by the diseases, which inspired the development of an image processing-based method for diagnosing banana leaf diseases. The original image of banana leaf disease was captured on a smartphone, and the green backdrop, which included healthy leaves and weeds, was removed using a color segmentation approach. After an image has been segmented using OSTU, where the intra-cluster or inter-cluster grey variance is minimized or maximized respectively, the area threshold approach is used to extract the whole disease regions. Then, use these carefully selected color attributes to build a feature vector for each banana leaf disease using standard industry-wide image data. Banana leaf diseases can be identified with the help of a minimal Euclidean distance classifier. Grey leaf spot disease was detected at a

rate of 91.7% and Sigatoka disease was detected at a rate of 90% from the original banana leaf image.

TITLE 4: Disease detection in banana trees using an image processing-based thermal camera [1]
YEAR OF PUBLICATION: 2021

The study suggests employing a thermal camera to monitor banana trees for signs of disease. Several thresholding techniques and image processing algorithms are used to make the detection. A thermal camera is used to capture the image, which is then adjusted in advance. After that, an image registration technique is applied to the thermal camera's output to guarantee perfect spatial correspondence with the digital camera's image. The efficiency of the suggested method is determined by comparing the processed image to the "ground truth" image captured by the digital camera. In order to measure how well the strategy works, they employ performance measures. Parameter values over 80%, such as recall 85.4%, precision 89.35%, F measure 87.33%, and accuracy 92.8%, demonstrate the superiority of the proposed approach.

TITLE 5: Detection and Classification of Banana Leaf diseases using ML and DL Algorithms [9]
YEAR OF PUBLICATION: 2022

In order to identify and categorize diseases in banana leaves, this research provides three models using KNN, SVM, and Alexnet. This inquiry focuses on the diagnosis and categorization of the Leaf spot and Sigatoka diseases. The model is taught to differentiate between diseased and good leaves using RGB color images with and without a background. After the data has been cleaned up and the images have been preprocessed, they may be utilized to train a model. KNN, SVM, and AlexNet all passed the test with an excellent accuracy score.

TITLE 6: AI-powered banana diseases and pest detection [12]

YEAR OF PUBLICATION: 2019

To develop a detection model, they retrained three distinct CNN architectures using transfer learning. Data from 18 different categories of banana plant images (disease by plant component) were utilized to inform the creation of six separate models. In terms of accuracy, discovered that ResNet50 and InceptionV2 outperformed MobileNetV1. Most models constructed using these architectures have an accuracy of 90% or higher in detecting banana diseases and pests, resulting in cutting-edge outcomes. These experimental findings stood up to cutting-edge theoretical alternatives. They evaluated MobileNetV1's single-shot detector (SSD) with the hopes of eventually putting its detecting abilities to use on a mobile device. Using performance and validation markers, many automated sickness detection algorithms were compared.

TITLE 7: Classification, Detection, and Diagnosis of Banana Leaf Diseases using DL Technique [16]

YEAR OF PUBLICATION: 2019

The author proposes a DL model for identifying and treating plant diseases using images of healthy leaves and ill plants. Introduced a deep learning improvement to aid in the classification of diseases on banana leaves and the subsequent diagnosis of those diseases. To classify image datasets, they uses an CNN. Early data show that the suggested approach is robust even in challenging conditions with changing background complexity, size, image resolution, and position.

TITLE 8: Design and Evaluation of a CNN for Banana Leaf Diseases Classification [2]

YEAR OF PUBLICATION: 2020

Images were taken from a freely available online database since they were deemed to be relevant to the current investigation. Banana leaf RGB images were used to teach a CNN to identify plant diseases. The no-regularization model, the Dropout model, and the weight regularization model were all taken into account. Accelerating NN training with the Adam optimizer, and fine-tuning hyper-parameters with the K-fold cross-validation technique. In addition, were utilized to evaluate the NNs' efficacy. CNN outperformed competing models in predicting diseases in banana plants despite a lack of data.

TITLE 9: Eight Convolutional Layered Deep CNN-based Banana Leaf Disease Prediction [3]

YEAR OF PUBLICATION: 2022

The researchers propose a unique eight-layer Deep CNN to provide the most precise forecasts about banana leaf disease. The KAGGLE repository's Banana leaf dataset is used in the model. The dataset includes 937 high-resolution images that have been preprocessed to distinguish between four disease categories: Cordana, Healthy, Pestalotiopsis, and Sigatoka. The proposed Deep CNN with Eight Convolutional Layers featured a single input layer, a dense layer, an average pooling layer, and an output layer. The eight convolution layers in the model were followed by max-pooling layers for a total of sixteen layers. There are a total of 777 images in the training set, 80 in the validation set, and 80 in the testing set for the banana leaf dataset. In this case, the banana leaf dataset was used for both training and testing the eight-layer Deep CNN. When compared to other models of CNNs, the metrics produced by the proposed method are

superior.

TITLE 10: Portable Sigatoka Spot Disease Identifier on Banana Leaves Using Support Vector Machine (SVM) [15]

YEAR OF PUBLICATION: 2021

The goal of this research is to help banana farmers diagnose the Sigatoka spot by developing a portable method for detecting the illness of banana leaves. The system's output indicates both the categorization and area of the leaf. Using an SVM, banana leaf images were resized, leveled, segmented, features extracted, and categorized. With an overall accuracy of 90%, the proposed method successfully differentiated between healthy and Sigatoka-infected banana leaves.

CHAPTER 3

PROPOSED METHODOLOGY

In Chapter 3, we dive deep into the methodology we provide for gathering data and selecting the optimal model. To begin, we use data from the Mendeley website to develop an approximate diagnosis of banana leaf disease. The collection includes images from three different conditions: Healthy, Segatoka, and Xanthomonas. RGB image data has been gathered. In order to make sense of the information gathered, data analysis is conducted. Achieving the desired level of precision in a DL model requires more than simply feeding the collected raw image into it. Time spent on the preliminary processing of collected data is time well spent. The pre-processing in this investigation consists of resizing, rescaling, and augmentation. Data is separated into three sets after initial processing: train, validate, and test. Eighty percent of the data set is made up of training data, while the remaining twenty percent is made up of validate and test data. We have finished building the DL model. In this work, DL models like CNN and CNN-LSTM are employed. Eighty percent of the data is used to teach the model, and ten percent is utilized for validation. After DL models have been trained and validated, the final 10% of data is used for testing. Finally, the best model for detecting banana leaf diseases is selected. Metrics like accuracy and loss are utilized to evaluate the model. Figure 3.1 depicts the proposed system's flowchart.

Banana leaf disease recognition is just one area where several DL models have been deployed, but not with the desired accuracy. The purpose of this study is to develop a hybrid DL model that improves the model's accuracy score. The sections that follow provide additional detail regarding each step that must be taken during the research.

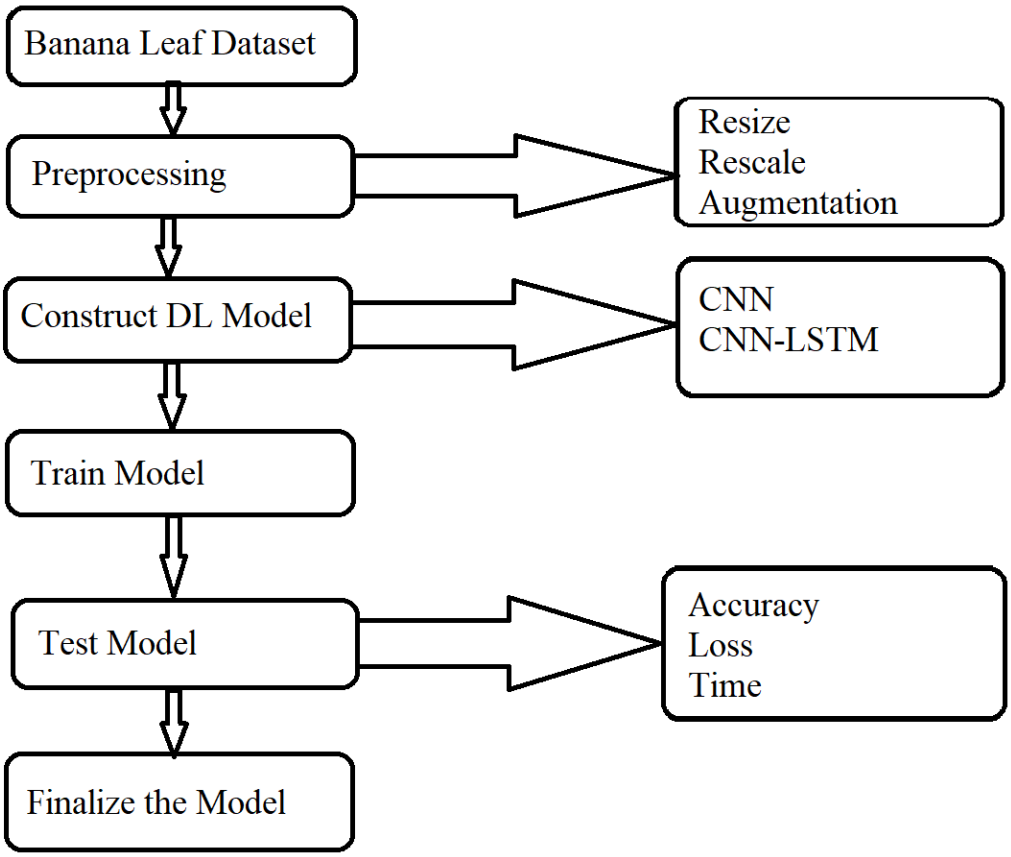


Fig. 3.1. Flowchart of the proposed system

3.1. DATA COLLECTION:

Data is DL's lifeblood. DL systems require a large amount of data and precise labeling from the start. In the world of computer science, an old adage says, "Garbage in, trash out." High-quality labeled data is required in DL projects for both model training and testing. However, acquiring this data can be time-consuming and expensive.

For academics, data collection is a time-consuming and arduous process. Nursing researchers must first evaluate the information demands of their study as well as potential data sources. Both newly collected data and previously collected data can be utilized. Some research issues benefit tremendously from the use of previously collected data, such as records and information. If researchers discover

that the material at their disposal is insufficient, they will need to seek further information. There are two more sorts of information: primary data and secondary data. The gathering of information from study participants by direct interaction with the researcher or a professional data collector is referred to as primary data collection. Secondary sources for data collection include clinical documents and government records. Before selecting a data-collecting technique and designing a data-collection plan, nurse researchers must first determine the data needs of their study. When choosing on a data collection approach, ethical guidelines, expenses, timeliness, population suitability, and availability of study subjects should all be considered. In this study, interviews were the primary method of data gathering.

3.1 DATA:

The banana leaf's healthy and diseased images are collected from the Mendeley site [5]. The diseased images taken in this research are Segatoka and Xanthomonas. The sample healthy images are given in Figure 3.2.

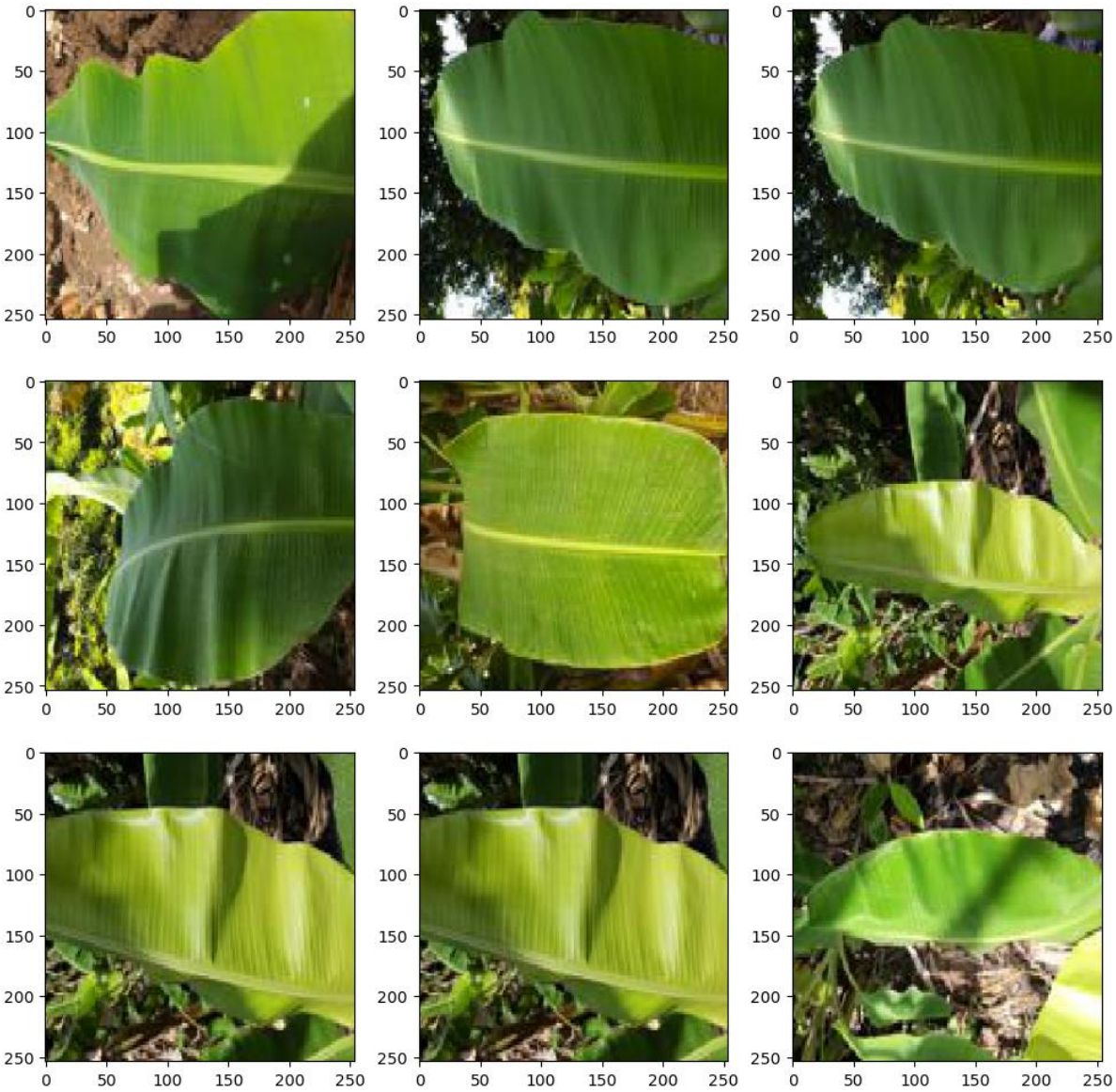


Fig. 3.2. Healthy banana leaf

Mycosphaerella produces a black streak on the leaves. The Fijians think that the Segatoka infection is produced by a swarm of fungi that are all interconnected. The collapse of leaves caused by massive viral sores impairs photosynthesis and ultimately kills the plant. The lesions initially appear rusty brown, indicating that the infection is in its early stages. They develop a deeper color and eventually succumb to despair as they mature. The sample images of Segatoka are given in Figure 3.3.

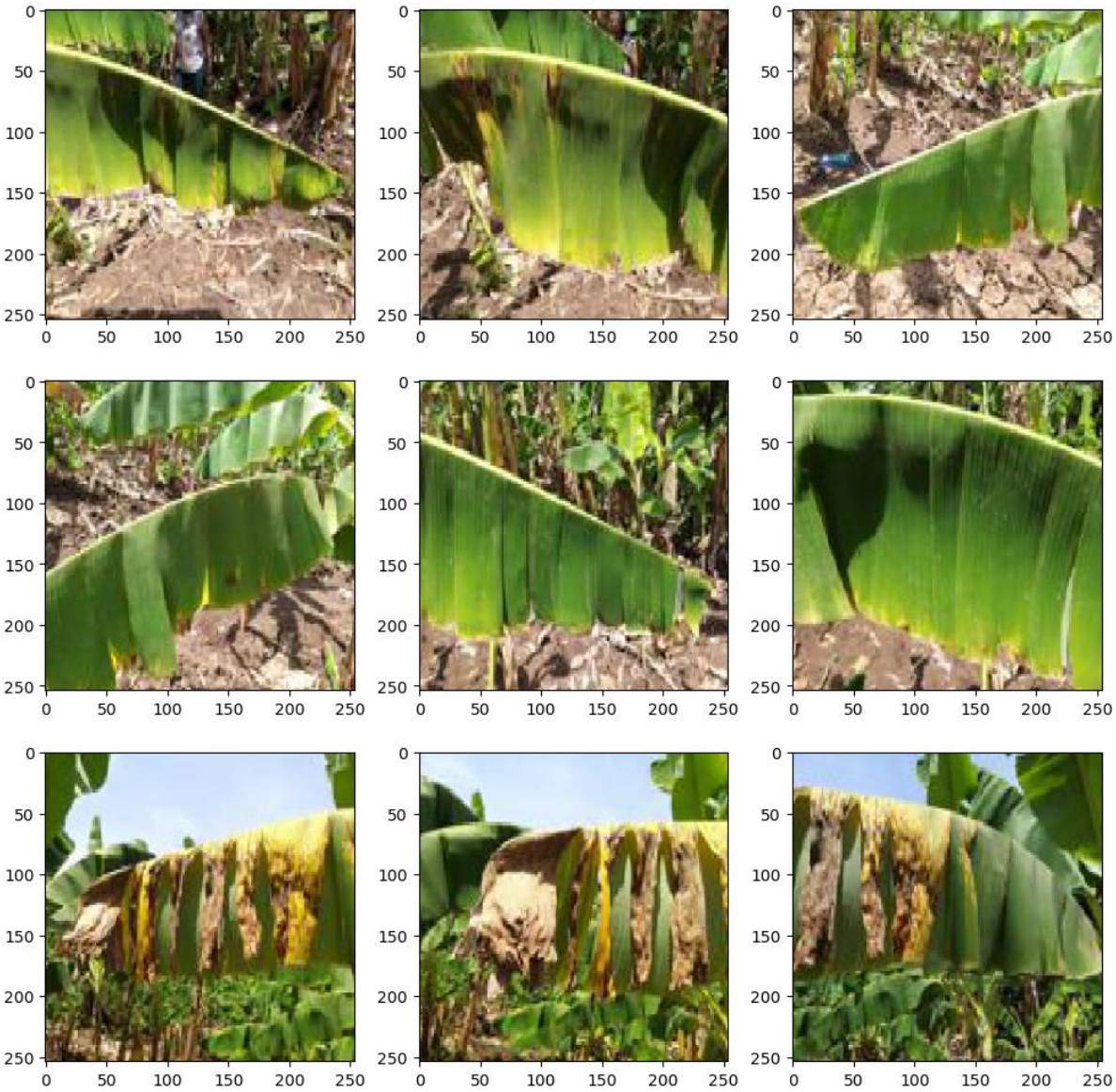


Fig. 3.3. Banana leaf affected by Segatoka

The bacteria Xanthomonas Campestris is responsible for Banana Xanthomonas Wilt (BXW). BXW is just one of many banana-infecting viruses, yet it has done a lot of damage. BXW has caused a 30-52% drop in banana output. In its early phases, the diseased plant has a delicate yellow-orange color, and in its later stages, it turns a dark brown color. If not managed effectively, it leads to mortality, which can cause a complete loss of yield. Effective treatments include both field cleanliness and removal of damaged plant parts or spraying with Streptocycline 200 ppm every 10 days following the first obvious symptom. The

spread can be halted by spraying a 0.3% copper oxychloride solution. The sample images of Xanthomonas are given in Figure 3.4.

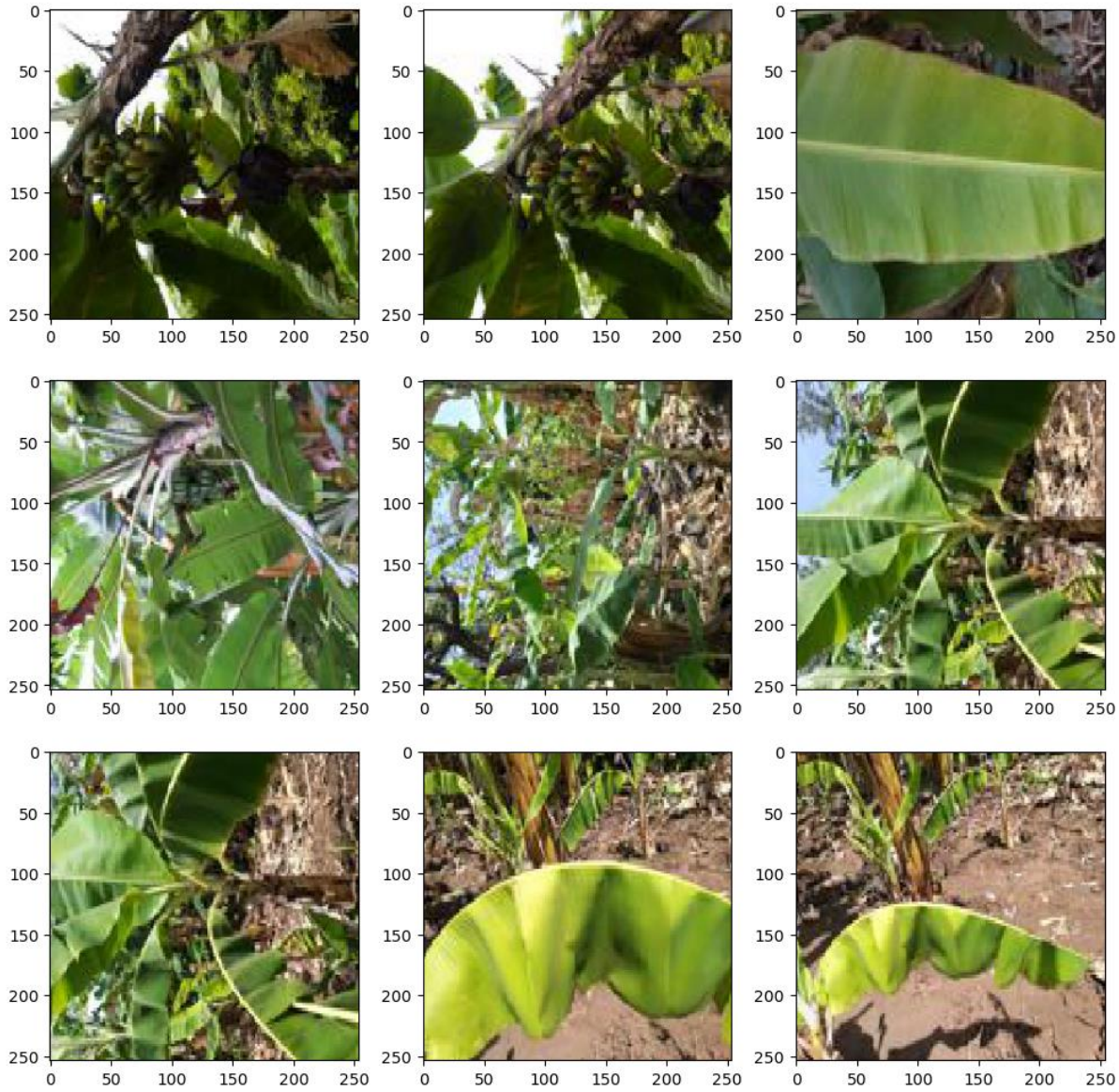


Fig. 3.4. Banana leaf affected by Xanthomonas

3.2. DATA PRE-PROCESS:

Data pre-processing is very important in this research. The collected images are undergone some processing techniques which are detailed below.

3.5.1 Resize:

In the discipline of DL, "image resizing" refers to the process of enlarging or shrinking an image while retaining its original proportions. Image scaling is a common pre-processing step in DL problems involving image data. There are a few instances where downsizing an image to a lesser size would be beneficial. To ensure that a DL model can handle all inputs consistently, scale all input images to the same size, for example. Reducing the resolution of an image while scaling it can also help to keep processing costs down.

Only a few of the approaches for scaling images in DL are nearest-neighbor, bilinear, and bicubic interpolation. The way these algorithms use the original image to determine the new pixel values for the scaled version varies. When determining which interpolation strategy to utilize, the trade-off between computing efficiency and image quality can differ from one DL assignment to the next.

3.5.2 Rescale:

Pixel values are typically unsigned integers ranging from 0 to 255. Although these pixel values can be supplied straight into Neural Network (NN) models without any preprocessing, doing so can lead to modeling issues, such as slower-than-desired model training. Instead, it can be beneficial to prepare the image pixel values before modeling, such as centering and standardizing the values or simply scaling them to the range 0-1 to make them easier to deal with.

Most images have pixel values that range from 0 to 255. When large integer numbers are utilized to process inputs, learning in NNs is hampered or halted. As a result, it is advised to normalize the pixel values so that all values fall between 0 and 1. Images can be viewed normally even if their pixel values are not in the 0 to 1 range. Simply divide each pixel value by 255, the maximum allowable

pixel value, to achieve this. This is done for all channels, regardless of the range of pixel values present in the image.

3.5.3 Augmentation

Deep NNs require a vast quantity of training data to produce accurate results while avoiding over-fitting. However, gathering enough data for training may be difficult. A number of variables could make gathering adequate data exceedingly difficult or impossible:

- The initial stage in constructing a training dataset is to obtain images and then label them. If you're working on an image classification task, for example, you'll want to use precise class labels. The method can be fairly costly due to the time and effort necessary in manually tag the training data. To accurately classify images, for example, you'll need to recruit expensive domain experts.
- The process of merely acquiring training images might be difficult at times. Working with healthcare data is highly regulated, and gaining access to it is difficult. In some circumstances, collecting the training images is desirable, but it comes at a high cost.

Shear: Images are frequently "sheared" along an axis in order to produce or correct perception angles. It is frequently used to improve images so that computers can understand the human perspective better. The shear is done on the image by the value of 0.2.

Zoom: The zoom augmentation technique is employed when zooming in on an image. The image is enlarged at random by either a magnification factor or by adding extra pixels to the edges. This technique makes advantage of the zoom_range option of the ImageDataGenerator class. To supply the percentage value of the zooms, we can use a float, range array. When you specify a float

value for the zoom-in, the range is [1-floatValue, 1+floatValue]. If the float value is set to 0.5, the resulting range is [0.5, 1.5], with the first half of the interval representing a 50% zoom-in and the second half being a 150% zoom-out. Zoom, like the brightness parameter, has upper and lower bounds that cannot be changed. Any zoom value less than 1.0 enlarges the image, while any number greater than 1.0 reduces its size. In this research, 0.2% zoom is employed.

Horizontal flipping: An image that has been flipped has been rotated 90 degrees around its vertical or horizontal axis. Depending on whether you flip horizontally or vertically, the axis of rotation will change orientation. The horizontal orientation of an image's pixel rows and columns is flipped in horizontal flip augmentation. Vertical flip enhancement is the process of flipping the horizontal orientation of pixels in an image from top to bottom.

3.3. DATA SPLIT:

The term "split" is used when data is divided into smaller pieces. Data testing occurs on one side of a two-way split, while model training occurs on the other. In data science, data splitting is essential, especially when developing models from unstructured data. The precision of data models and their supporting systems, such as DL, is enhanced by this method. When data is divided into two parts, one part is used for training models. Parameters for a wide variety of models can be determined by analyzing a training set. After the training data set has been employed, the test data set is next used. Data from training and testing must be compared to guarantee the final model's accuracy. In DL, many different types of data sets are used. The third and final group is the development set, which is employed to fine-tune the instruction procedure. It is an unusual practice to divide data into subsets determined by factors like the size of the original dataset or the number of predictors in a particular model. In this project, we divide the data into three sets: training, validating, and testing. The training set comprised 80% of the data, 10% of the validation set, and 10% of the test set.

3.4. DL MODEL:

DL and artificial intelligence are being used extensively in image classification. Classifying images entails categorizing them based on the features they share. These features are discovered by the algorithm and used to classify and label images.

3.4.1 CNN

In this study, a convolutional network is employed, which includes a convolutional layer, activation function, pooling layer, regularization, and optimizer parameters. The architecture of the CNN model is given in Figure 3.5.

Convolutional layer: The primary building block of CNNs is the convolutional layer. The technical term for its component parts is "kernels," which are convolutional filters. The input image is convolved with these filters to produce the feature map [10].

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 252, 252, 32)      896

 batch_normalization_6 (Batc  (None, 252, 252, 32)     128
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 126, 126, 32)     0
 2D)

 conv2d_5 (Conv2D)           (None, 124, 124, 128)     36992

 batch_normalization_7 (Batc  (None, 124, 124, 128)    512
 hNormalization)

 max_pooling2d_5 (MaxPooling  (None, 62, 62, 128)      0
 2D)

 conv2d_6 (Conv2D)           (None, 60, 60, 256)       295168

 batch_normalization_8 (Batc  (None, 60, 60, 256)      1024
 hNormalization)

 max_pooling2d_6 (MaxPooling  (None, 30, 30, 256)      0
 2D)

 conv2d_7 (Conv2D)           (None, 28, 28, 64)        147520

 batch_normalization_9 (Batc  (None, 28, 28, 64)       256
 hNormalization)

 max_pooling2d_7 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 12544)             0

 dense_3 (Dense)             (None, 64)                802880

 batch_normalization_10 (Bat  (None, 64)               256
 chNormalization)

 dense_4 (Dense)             (None, 3)                 195

=================================================================
Total params: 1,285,827
Trainable params: 1,284,739
Non-trainable params: 1,088
_________________________________________________________________
```

Fig. 3.5. CNN Architecture

- Kernel: A grid of numbers or values represents the kernel. Individual numbers are the kernel weights. When a CNN is first trained, its kernel weights are based on a random number generator.
- Convolutional Operation: First, we'll go over the CNN input format. The standard NN accepts data in vector format, whereas the CNN accepts data as a multichannel image. The kernel initially scrolls

through the entire image in both directions. In addition, the dot product is calculated, which involves multiplying the values of both the input image and the kernel by themselves and then adding the resulting scalar to get a final value. Sliding is physically impossible with continuous practice. The feature map in the output illustrates the computation of dot product values. The figure shows the crucial calculations performed at each stage.

Pooling Layer: Subsampling the feature maps is the main task of the pooling layer. Convolutional methods are used to generate these maps. This approach reduces huge feature maps to manageable proportions. Simultaneously, it retains the large majority of the most relevant data (or features) throughout the pooling process [14]. Before the pooling method, the stride and kernel are both assigned sizes, exactly as they are during the convolutional operation. A variety of pooling mechanisms can be used by different pooling layers.

Activation Function: Activation functions in CNN have the fundamental job of mapping input to output. The bias (if any) is applied to the weighted sum of the neuron input to calculate the input value. To identify whether or not a neuron fires in response to a given input, the activation function produces the corresponding output. ReLU serves a practical use in the context of CNN. It takes the input and converts all of the numbers to positive values. Sigmoid and $tanh$ activation functions have been replaced by a simpler and more effective technique.

CNN regularization: Over-fitting and under-fitting are discussed in further detail in later sections, and the regularization employs a variety of simple notions to help prevent problems.

- A typical method of generalization is a dropout. Throughout each training iteration, neurons are selectively destroyed at random.
- The batch normalization procedure is utilized to ensure the efficacy of the final activations. The distribution of the results is Gaussian with one standard deviation. The result at each layer can be made equal by taking away the mean and dividing by the standard deviation.

Optimizer selection: The learning process involves two major issues: the first is the selection (optimizer) of the learning algorithm, and the second is the application of multiple upgrades other than the learning algorithm to improve the output. All supervised learning algorithms seek to decrease error (the difference between actual and expected output) by maximizing a loss function based on a large number of learnable elements (such as biases, weights, and so on). The optimizer's name is Adam, and the loss function is the sparse categorical cross-entropy.

3.5.2 CNN-LSTM

Memory blocks are special units that are part of the LSTM's recurrent buried layer. Memory blocks have special multiplying units called gates and memory cells that connect to themselves and keep the temporal state of the network [18]. Each memory unit had its own input and output gate in the initial architecture. According to the state of the input gate, input activations are routed into the memory cell. When a cell is stimulated, a signal is sent out through its output gate and into the rest of the network. The memory block ultimately included the forget gate.

Input gate: The input gate is responsible for adding fresh data to the cell's current state. As illustrated in the previous diagram, the process of information addition is divided into three steps.

- Using a sigmoid function to modify the required cell state additions. This, like the forget gate, filters the information from h_{t-1} and y_t
- Making a vector of all the values that can be added to the present state of the cell (depending on h_{t-1} and y_t). The tanh function, whose output ranges from 1 to 1, is employed for this purpose.
- The current state of the cell is increased by multiplying the sigmoid gate's value by the vector generated by the $tanh$ function.

Output gate: An output gate has three distinct phases of operation:

- A vector with values between -1 and +1 is generated when the $tanh$ function is added to the cell state.
- Building a filter with h_{t-1} and y_t values to regulate the values output from the vector created in the previous phase. In this filter, a sigmoid function is utilized once more.
- The value of the regulatory filter is multiplied by the vector formed in step 1, with the result delivered as an output and impacting the hidden state of the following cell.

Forget gate: A forget gate is in charge of deleting the cell's state. Filter multiplication discards or downgrades information that the LSTM no longer needs to comprehend. This is essential in order to get the most out of the LSTM network.

As inputs, this gate accepts h_{t-1} and y_t . x_t represents the input at a particular time step, while h_{t-1} represents the hidden state from the preceding cell or its output. The output is produced by multiplying the inputs by the weight matrices and then adding a bias. The sigmoid function is then applied

to this integer. Each number in the cell state is represented as a scalar in the output vector of the sigmoid function, with values ranging from 0 to 1. To determine which observations to retain and which to discard, the sigmoid function is applied. A '0' at the forget gate's output indicates that the gate has instructed the cell state to disregard the value at that location. If the forget gate receives a ‘1’, it means it wants to keep all the data. The cellular state is used as a multiplication on the sigmoid function's vector output.

Each vector in these LSTM components can be analytically determined as follows.

$$Y = y_t \oplus h_{t-1}$$

$$g_t^i = \sigma(W^i . Y + b^i)$$

$$g_t^f = \sigma(W^f . Y + b^f)$$

$$g_t^o = \sigma(W^o . Y + b^o)$$

$$m_t = \tanh\,(W^m . Y + b^m)$$

$$c_t = g_t^f \odot c_{t-1} + g_t^i \odot m_t$$

$$h_t = g_t^o \odot \tanh\,(c_t)$$

σ → Sigmoid Function

W → Weighted Matrices

$\odot$ → Element-wise Multiplication

b → Bias Vectors.

During training, these values are further refined. Each LSTM unit functions like a state machine after three gates have their weights. The CNN-LSTM architecture is a combination of CNN and LSTM networks which is illustrated in Figure 3.6, with the CNN serving as the feature extractor and the LSTM as a classification system.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 252, 252, 64)      1792

 batch_normalization (BatchN  (None, 252, 252, 64)     256
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 84, 84, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 82, 82, 128)       73856

 batch_normalization_1 (Batc  (None, 82, 82, 128)      512
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 27, 27, 128)      0
 2D)

 conv2d_2 (Conv2D)           (None, 25, 25, 128)       147584

 batch_normalization_2 (Batc  (None, 25, 25, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 8, 8, 128)        0
 2D)

 conv2d_3 (Conv2D)           (None, 6, 6, 256)         295168

 batch_normalization_3 (Batc  (None, 6, 6, 256)        1024
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 2, 2, 256)        0
 2D)

 reshape (Reshape)           (None, 4, 256)            0

 lstm (LSTM)                 (None, 4, 256)            525312

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 1024)              1049600

 batch_normalization_4 (Batc  (None, 1024)             4096
 hNormalization)

 dense_1 (Dense)             (None, 256)               262400

 batch_normalization_5 (Batc  (None, 256)              1024
 hNormalization)

 dense_2 (Dense)             (None, 3)                 771

=================================================================
Total params: 2,363,907
Trainable params: 2,360,195
Non-trainable params: 3,712
_________________________________________________________________
```

Fig. 3.6. CNN-LSTM Architecture

CHAPTER 4

RESULT AND DISCUSSION

The outcome of the CNN and CNN-LSTM models is discussed in this chapter. After data collection and processing, the DL model is trained and tested.

4.1 TRAINING AND VALIDATION PHASE

The accuracy of the CNN model in the training and validation stage is given in Figure 4.1. In the figure, the black plot represents the training accuracy while the green indicates the validation accuracy. The total epochs used in the training and validation phase is 15. The training accuracy starts with a value of 0.58 and ends with a value of 0.9. And, the validation accuracy starts and ends with scores of 0.28 and 0.89. The epoch counts are taken on the x-axis and accuracy values are taken in the y-axis.

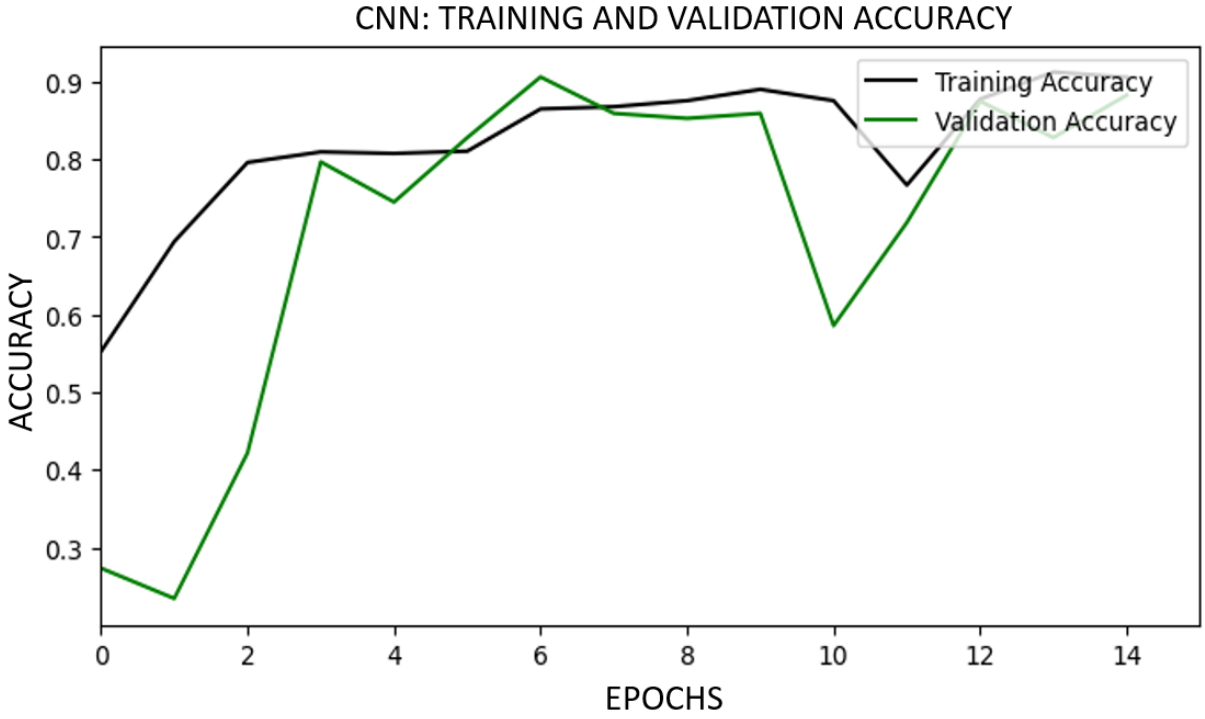


Fig. 4.1. CNN: Training and validation accuracy

Figure 4.2 displays the loss of the CNN model during the training and validation phases. The green line in the picture denotes the validation loss, whereas the black plot shows the training loss. The number of epochs employed during training and testing is 15. The training loss begins at 15 and ends at 0.284. Scores of 1.65 and 0.344 are the beginning and ending points of the validation loss, respectively. The number of epochs is plotted along the x-axis, while loss metrics are plotted along the y-axis.

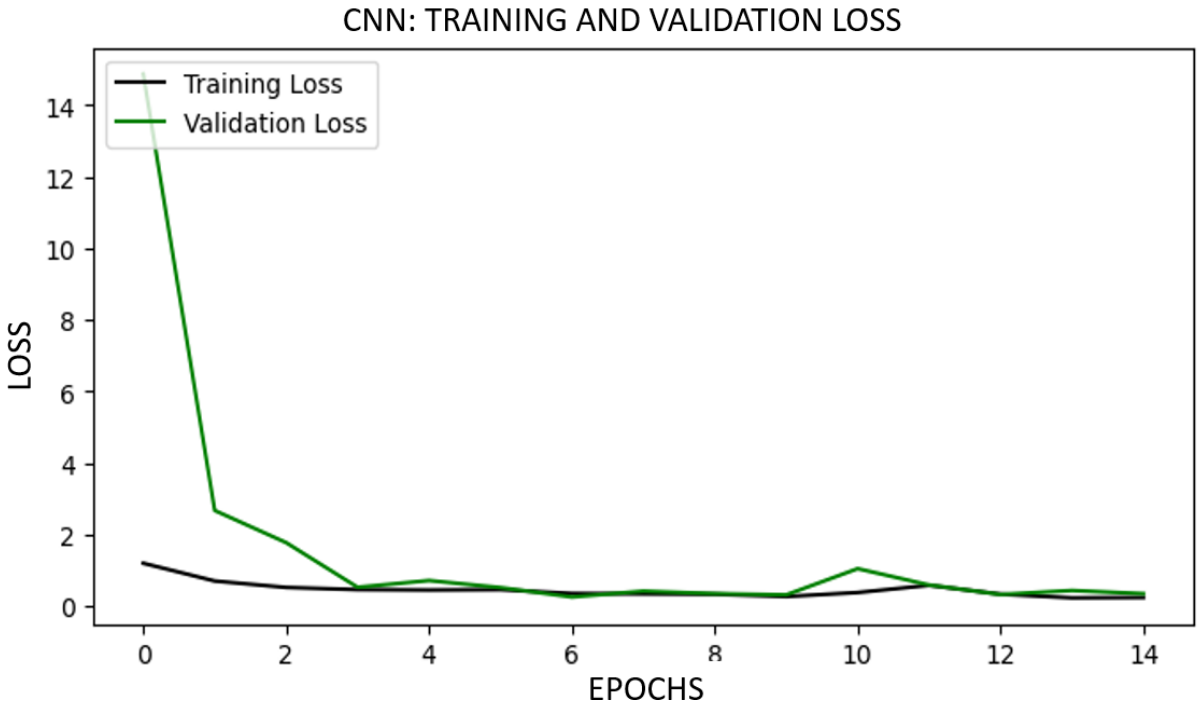


Fig. 4.2. CNN: Training and validation loss

The hybrid model taken for the research is CNN-LSTM. The accuracy of the CNN-LSTM model in the training and validation stage is given in Figure 4.3. In the figure, the blue plot represents the training accuracy while the red indicates the validation accuracy. The total epochs used in the training and validation phase is 15. The training accuracy starts with a value of 0.58 and ends with a value of 0.96. And, the validation accuracy starts and ends with scores of 0.472 and 0.945. The epoch counts are taken on the x-axis and accuracy values are taken on the y-axis.

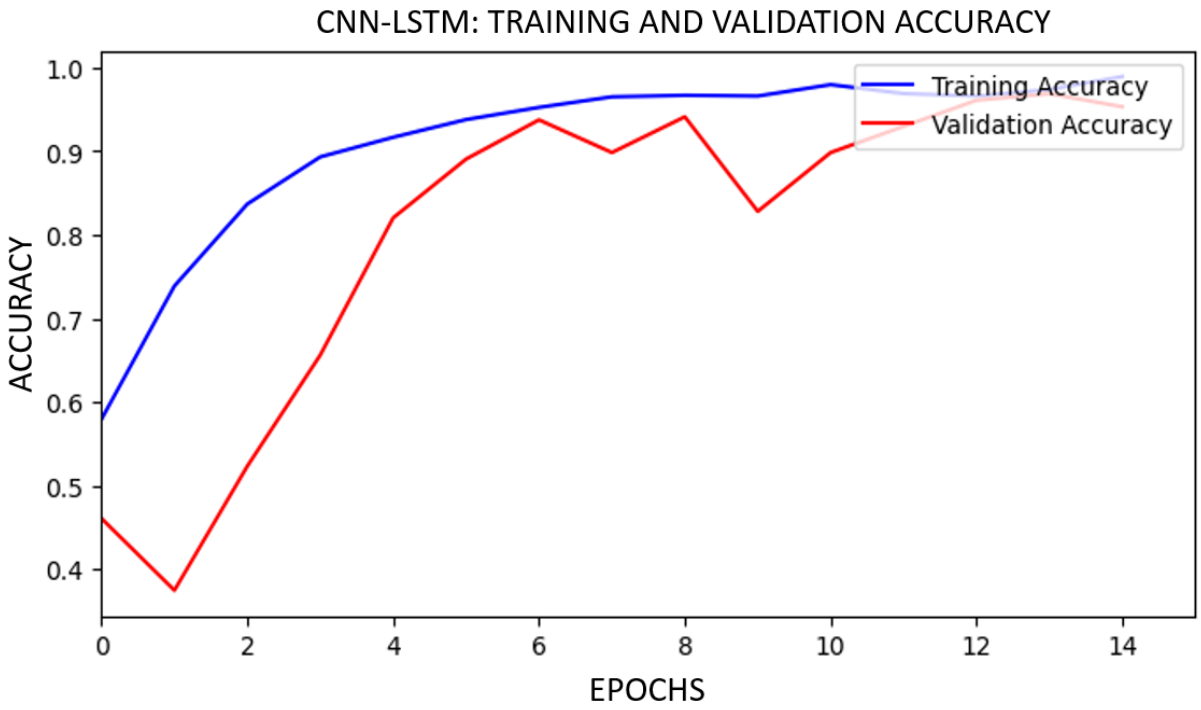


Fig. 4.3. CNN-LSTM: Training and validation accuracy

Figure 4.4 displays the loss of the CNN-LSTM model during the training and validation phases. The red line in the picture denotes the validation loss, whereas the blue plot shows the training loss. The number of epochs employed during training and testing is 15. The CNN-LSTM training loss begins at 2.9 and ends at 0.142. Scores of 1.2 and 0.175 are the beginning and ending points of the validation loss, respectively. The number of epochs is plotted along the x-axis, while loss metrics are plotted along the y-axis.

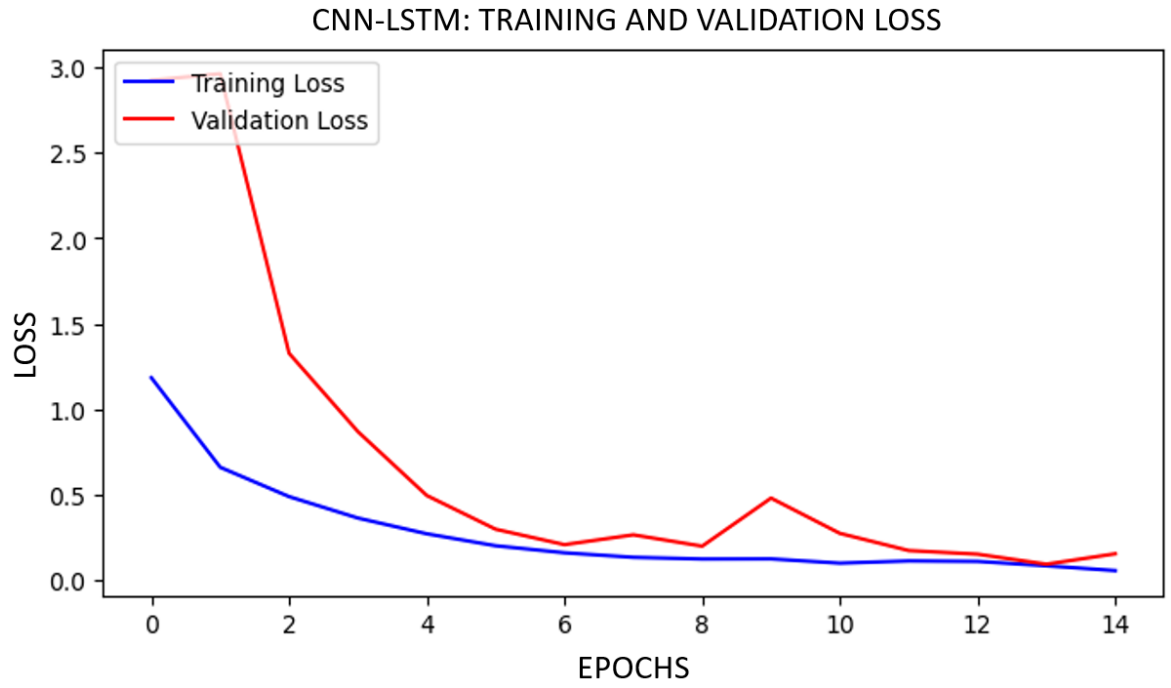


Fig. 4.4. CNN-LSTM: Training and validation loss

4.2 TESTING PHASE

After training and validation, both models CNN and CNN-LSTM are tested using the 10% of data. For testing, a total of 128 images are taken. The images are a mixture of healthy, and diseased banana leaves. The outcome of the CNN model on test data is illustrated in Figure 4.5 and the outcome of the CNN-LSTM model is given in Figure 4.6.

CNN TEST TESULT

```
test = tf.keras.preprocessing.image_dataset_from_directory(
    "Bananaleaf/test/",
    seed=123,
    shuffle=True,
    image_size=(254,254),
    batch_size=32
)
```

```
Found 128 files belonging to 3 classes.
```

```
yp = model.evaluate(test)
```

```
4/4 [==============================] - 4s 865ms/step - loss: 0.1960 - accuracy: 0.9219
```

Fig. 4.5. CNN performance on test data

CNN-LSTM TEST RESULT

```
test = tf.keras.preprocessing.image_dataset_from_directory(
    "Bananaleaf/test/",
    seed=123,
    shuffle=True,
    image_size=(254,254),
    batch_size=32
)
```

```
Found 128 files belonging to 3 classes.
```

```
yp = cnnlstmclassifier.evaluate(test)
```

```
4/4 [==============================] - 4s 1s/step - loss: 0.1730 - accuracy: 0.9453
```

Fig. 4.6. CNN-LSTM performance on test data

The performance of CNN and CNN-LSTM models are compared using accuracy, loss, and time taken to predict the results. The CNN model gives an accuracy of 0.9219 and a loss of 0.196. The time taken to predict the test images is 865ms. The CNN-LSTM model produces an accuracy and loss score of 0.9453 and 0.173. The CNN-LSTM takes 1 second to predict the banana leaf disease from 128 images.

Table 1. DL Model comparison

Model	Accuracy	Loss	Time
CNN	0.9219	0.196	865ms
CNN-LSTM	0.9453	0.173	1000ms

All the values in Table 1 are converted into a bar graph for better visualization. The performance of the model is compared and it is given in Figure 4.7. Next, figure 4.8 shows the comparison of time taken by the model to test the banana leaf images.

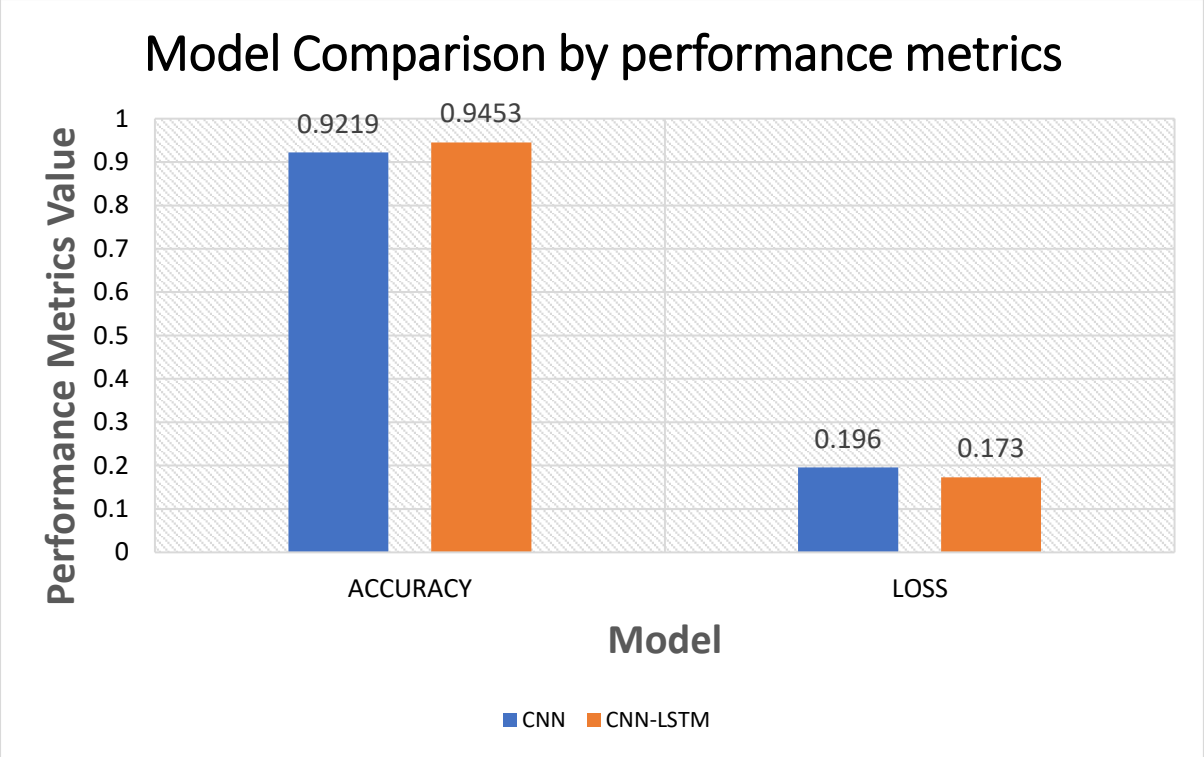


Fig. 4.7. DL Model comparison by performance metrics

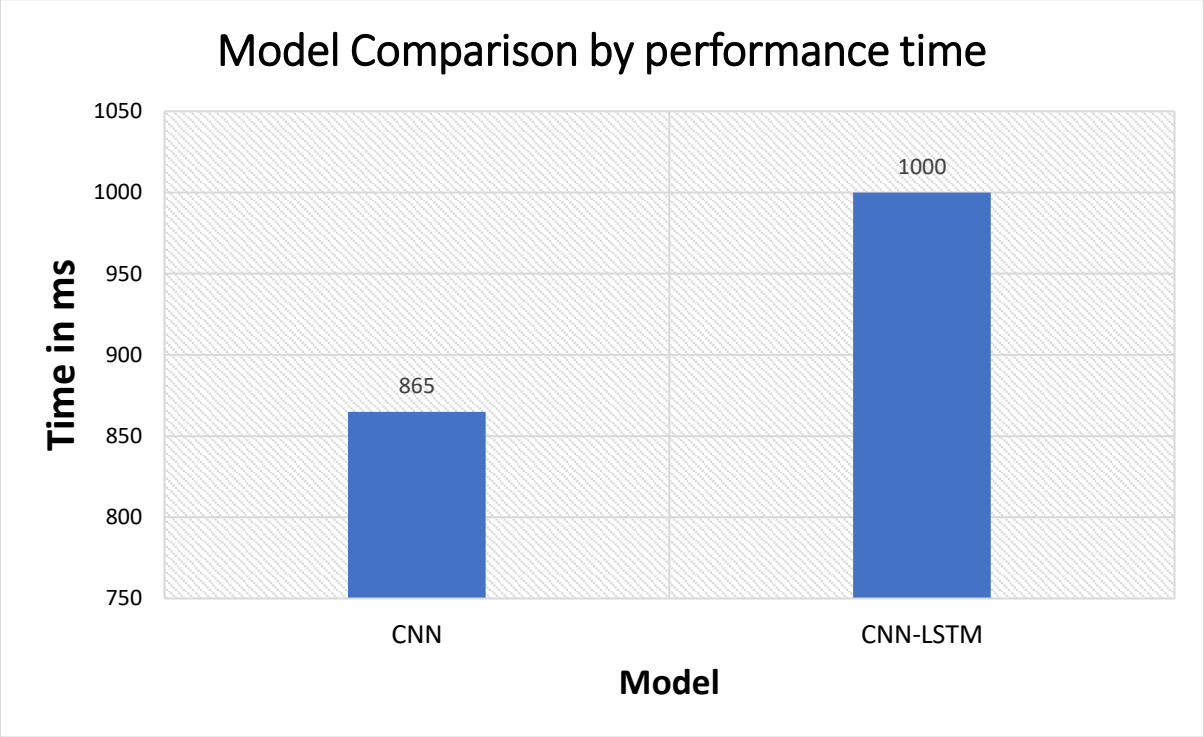


Fig. 4.8. DL Model comparison by time

This comparison helps to identify the best model for banana leaf disease detection. The CNN-LSTM gives the highest accuracy than the CNN model.

CHAPTER 5

CONCLUSION

Agriculture is crucial to the development of the economy and the provision of food for the world's population. This highlights the significance of the efficient management of agricultural commodities. Preventing diseases like banana Xanthomonas and Sigatoka from infecting bananas is essential since bananas are a crucial crop that is grown all over the planet. This study suggests a hybrid DL-based approach for detecting and classifying banana disease by analysing images of banana leaves. The suggested hybrid DL approach is most effective for detecting banana leaf disease, as evidenced by its high accuracy (94.5% on test samples) and minimal time needs. The images used in this study are from the image-sharing platform Mendeley. We then resize, rescale, and enhance the gathered images.

Future studies will involve developing and implementing the suggested approach to recognize the banana leaf diseases in the agricultural farm by employing the drone camera rather than examining an individual plant. Moreover, an accessible mobile app will be developed for the farmers to immediately recognize the disease and respond accordingly.

APPENDICES

Src Code:

import library

In [1]:
```
import os
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import cv2
from glob import glob
import warnings

warnings.filterwarnings('ignore')
%matplotlib inline

import seaborn as sns
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix


import tensorflow as tf
from tensorflow.keras import Input
from tensorflow.keras.models import Model, load_model, save_model ,Sequent
from tensorflow.keras.layers import Input, Activation, BatchNormalization,
from tensorflow.keras.layers import MaxPooling2D, concatenate, Dense, Flat
from tensorflow.keras.layers import LSTM

from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, Red

from tensorflow.keras import backend as K
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import models, layers

```

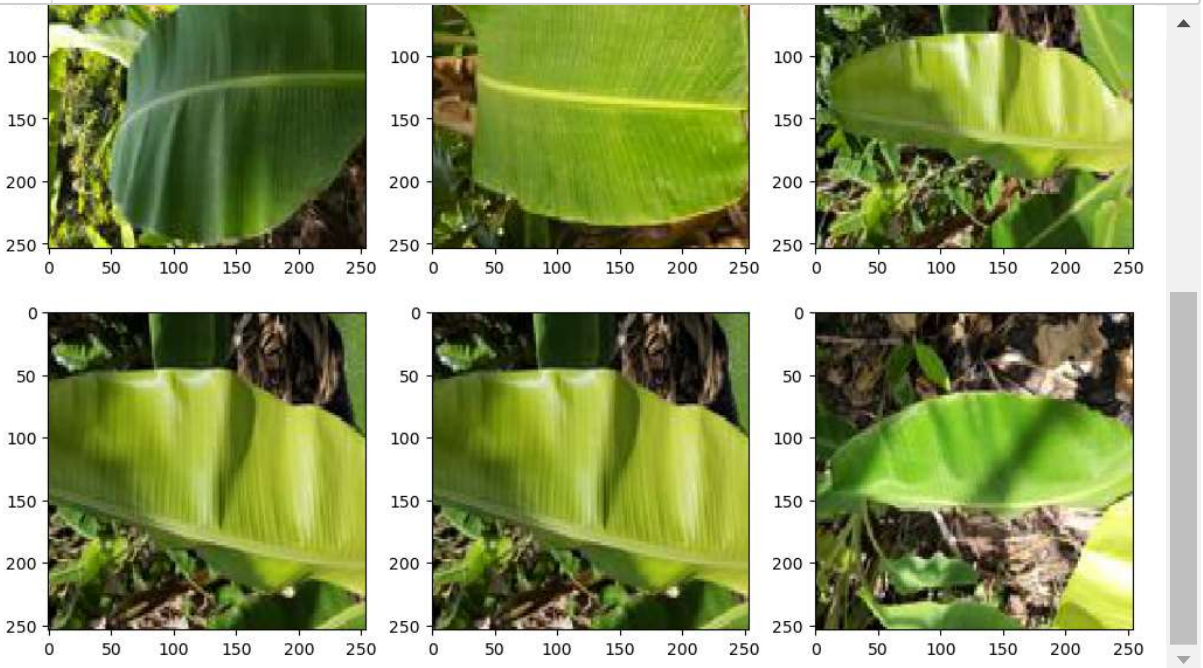
READ DATA

In [2]:
```
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    "Bananaleaf/train/",
    seed=123,
    shuffle=True,
    image_size=(254,254),
    batch_size=32
)
```

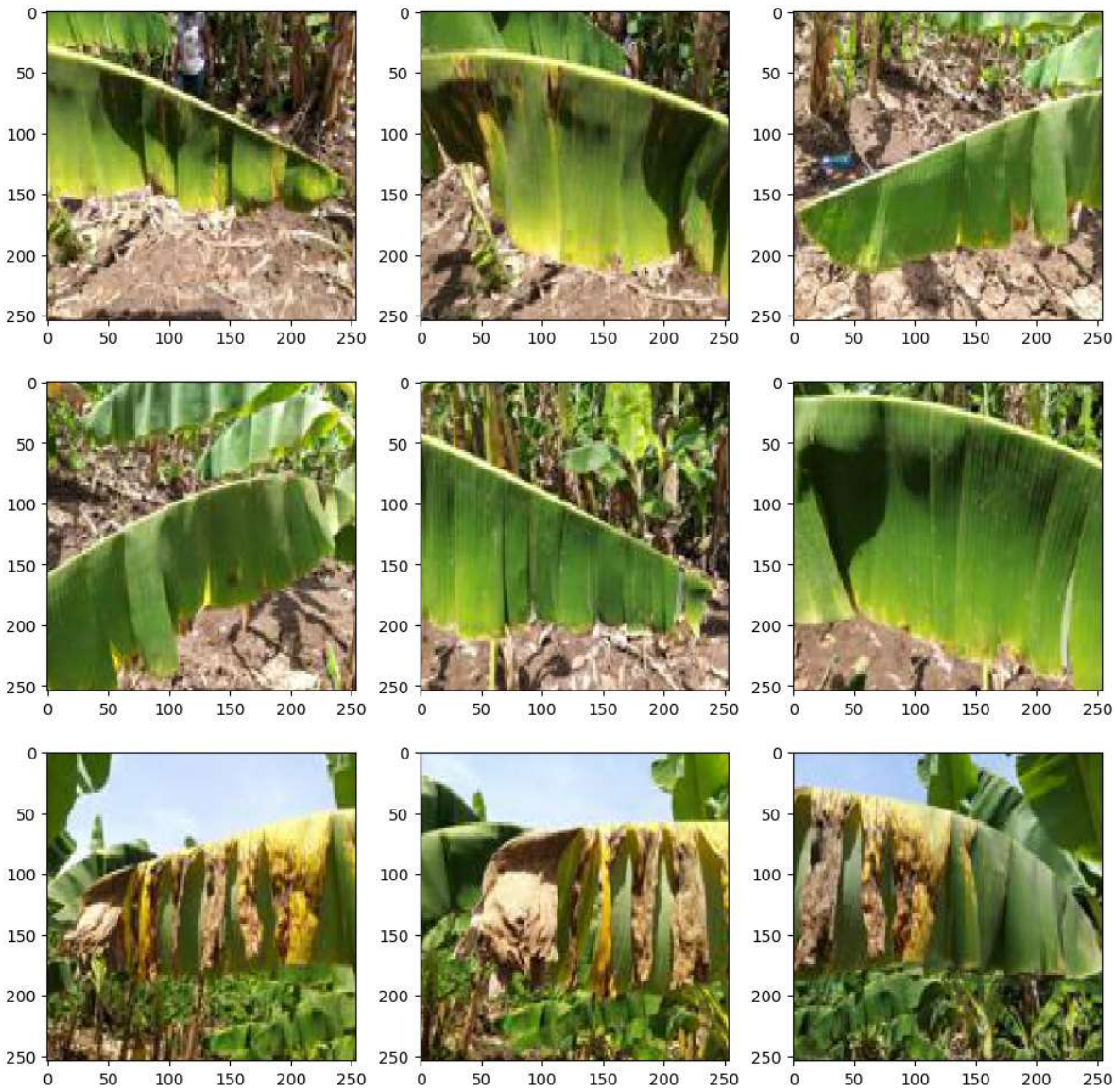
```
Found 1158 files belonging to 3 classes.
```

In [3]:
```
from tensorflow.keras.preprocessing.image import load_img
folder_path = "Bananaleaf/train/"
healthy_images = os.listdir(folder_path + '/healthy/')
segatoka_images = os.listdir(folder_path + '/segatoka/')
xamthomonas_images = os.listdir(folder_path + '/xamthomonas/')
dataset=[]
lab=[]


def plot_state(state):
    plt.figure(figsize= (12,12))
    for i in range(1, 10, 1):
        plt.subplot(3,3,i)
        img = load_img(folder_path + "/" + state + "/" + os.listdir(folder
        plt.imshow(img)
    plt.show()
```

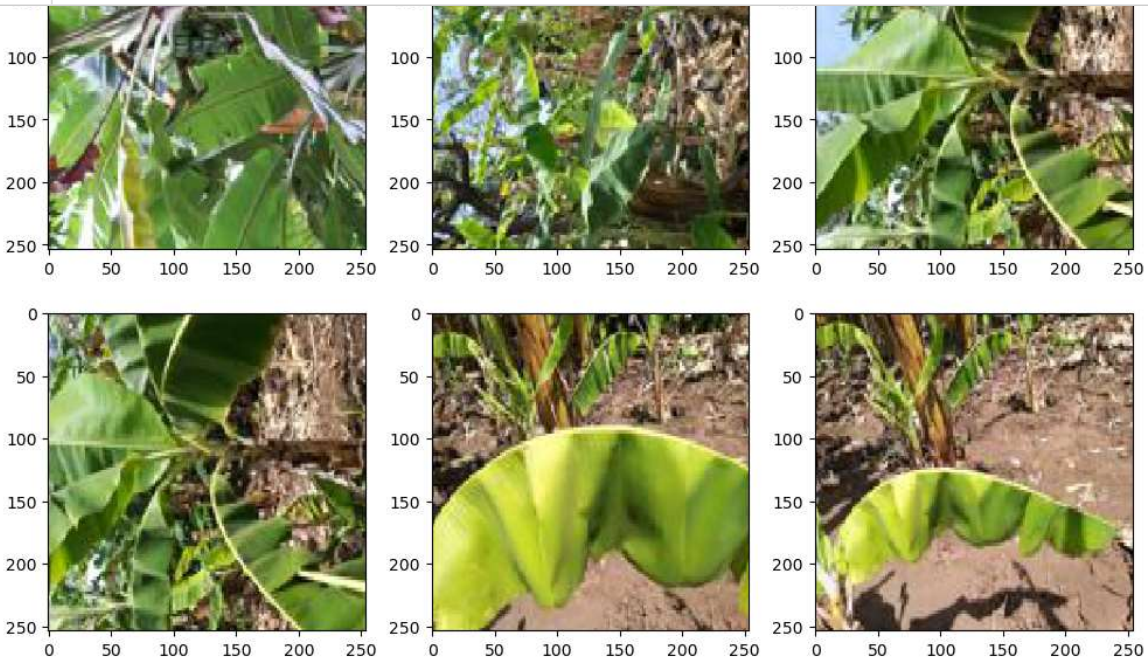
In [53]:
```
plot_state('healthy')
```



In [4]:
```
plot_state('segatoka')

```



In [55]:
```
plot_state('xamthomonas')
```



PREPROCESSING

In [5]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train = ImageDataGenerator(rescale = 1./255,
                           shear_range = 0.2,
                           zoom_range = 0.2,
                           horizontal_flip = True,
                           validation_split=0.2)

valid = ImageDataGenerator(rescale = 1./255,validation_split=0.2)
```

SPLIT DATA

In [25]:
```
def get_dataset_partitions_tf(ds, train_split=0.9, val_split=0.1, shuffle=
    assert (train_split + val_split) == 1

    ds_size = len(ds)

    if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)

    train_size = int(train_split * ds_size)
    val_size = int(val_split * ds_size)

    train_ds = ds.take(train_size)
    val_ds = ds.skip(train_size)

    return train_ds, val_ds
```

In [57]:
```
train_ds, val_ds = get_dataset_partitions_tf(dataset)
```

CNN-LSTM MODEL

In [4]:

```
input_shape = (254, 254, 3)
n_classes = 3

cnnlstmclassifier=Sequential([

                Conv2D(64,3,activation='relu',input_shape=(254,254,3)),
                BatchNormalization(),
                MaxPooling2D(3),

                Conv2D(128,3,activation='relu'),
                BatchNormalization(),
                MaxPooling2D(3),
                Conv2D(128,3,activation='relu'),
                BatchNormalization(),
                MaxPooling2D(3),
                Conv2D(256,3,padding='valid',activation='relu'),
                BatchNormalization(),
                MaxPooling2D(3),
                Reshape((4, 256)),
                LSTM(256, activation="relu", return_sequences=True),
                Flatten(),
                Dense(1024,activation='relu'),
                BatchNormalization(),
                Dense(256,activation='relu'),
                BatchNormalization(),
                Dense(3,activation='sigmoid')
])

cnnlstmclassifier.summary()

```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 252, 252, 64)      1792

 batch_normalization (BatchN  (None, 252, 252, 64)     256
 ormalization)

 max_pooling2d (MaxPooling2D  (None, 84, 84, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 82, 82, 128)       73856

 batch_normalization_1 (Batc  (None, 82, 82, 128)      512
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 27, 27, 128)      0
 2D)

 conv2d_2 (Conv2D)           (None, 25, 25, 128)       147584

 batch_normalization_2 (Batc  (None, 25, 25, 128)      512
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 8, 8, 128)        0
 2D)

 conv2d_3 (Conv2D)           (None, 6, 6, 256)         295168

 batch_normalization_3 (Batc  (None, 6, 6, 256)        1024
 hNormalization)

 max_pooling2d_3 (MaxPooling  (None, 2, 2, 256)        0
 2D)

 reshape (Reshape)           (None, 4, 256)            0

 lstm (LSTM)                 (None, 4, 256)            525312

 flatten (Flatten)           (None, 1024)              0

 dense (Dense)               (None, 1024)              1049600

 batch_normalization_4 (Batc  (None, 1024)             4096
 hNormalization)

 dense_1 (Dense)             (None, 256)               262400

 batch_normalization_5 (Batc  (None, 256)              1024
 hNormalization)

 dense_2 (Dense)             (None, 3)                 771

=================================================================
Total params: 2,363,907
Trainable params: 2,360,195
```

```
Non-trainable params: 3,712
```

In [23]:

```
cnnlstmclassifier.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

CNN-LSTM TRAINING RESULT

In [174]:
```
import tensorflow as tf
tf.config.run_functions_eagerly(True)
history = cnnlstmclassifier.fit(
    train_ds,
    batch_size=32,
    validation_data=val_ds,
    verbose=1,
    epochs=15,)
```

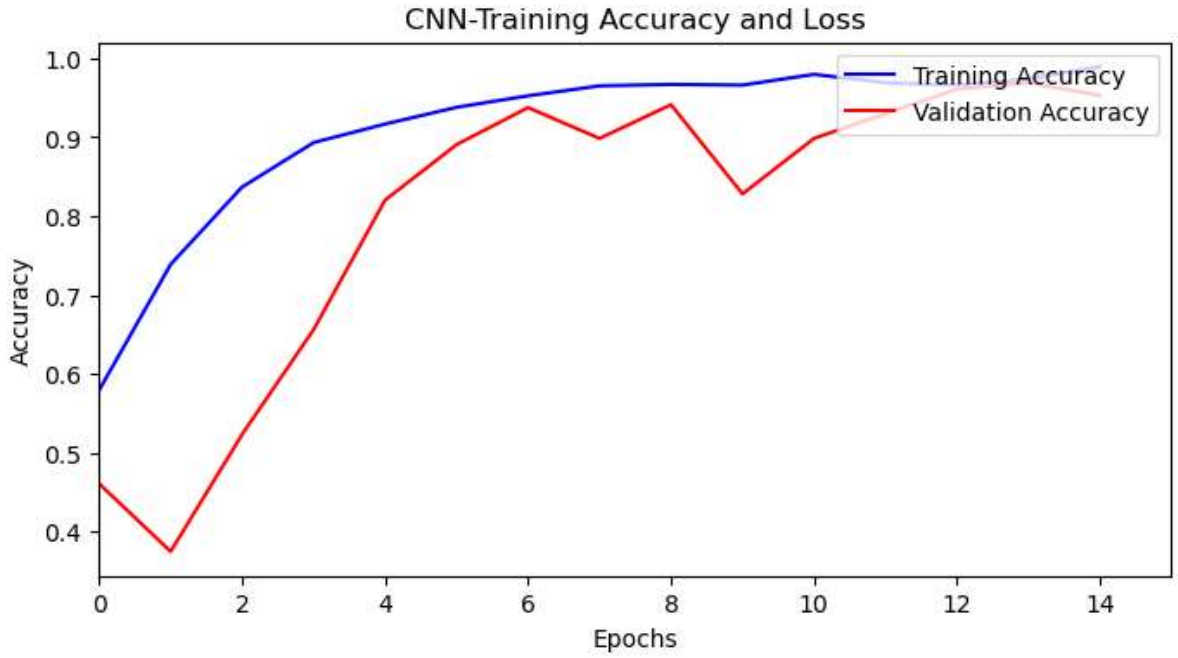
```
Epoch 1/15
33/33 [==============================] - 140s 4s/step - loss: 1.1837 - accura
cy: 0.5796 - val_loss: 2.9257 - val_accuracy: 0.4609
Epoch 2/15
33/33 [==============================] - 140s 4s/step - loss: 0.6595 - accura
cy: 0.7388 - val_loss: 2.9636 - val_accuracy: 0.3750
Epoch 3/15
33/33 [==============================] - 139s 4s/step - loss: 0.4877 - accura
cy: 0.8369 - val_loss: 1.3269 - val_accuracy: 0.5234
Epoch 4/15
33/33 [==============================] - 139s 4s/step - loss: 0.3621 - accura
cy: 0.8932 - val_loss: 0.8682 - val_accuracy: 0.6562
Epoch 5/15
33/33 [==============================] - 143s 4s/step - loss: 0.2699 - accura
cy: 0.9167 - val_loss: 0.4941 - val_accuracy: 0.8203
Epoch 6/15
33/33 [==============================] - 139s 4s/step - loss: 0.1999 - accura
cy: 0.9379 - val_loss: 0.2967 - val_accuracy: 0.8906
Epoch 7/15
33/33 [==============================] - 142s 4s/step - loss: 0.1586 - accura
cy: 0.9524 - val_loss: 0.2064 - val_accuracy: 0.9375
Epoch 8/15
33/33 [==============================] - 140s 4s/step - loss: 0.1322 - accura
cy: 0.9650 - val_loss: 0.2638 - val_accuracy: 0.8984
Epoch 9/15
33/33 [==============================] - 142s 4s/step - loss: 0.1223 - accura
cy: 0.9669 - val_loss: 0.1975 - val_accuracy: 0.9412
Epoch 10/15
33/33 [==============================] - 139s 4s/step - loss: 0.1230 - accura
cy: 0.9660 - val_loss: 0.4789 - val_accuracy: 0.8281
Epoch 11/15
33/33 [==============================] - 139s 4s/step - loss: 0.0980 - accura
cy: 0.9796 - val_loss: 0.2724 - val_accuracy: 0.8984
Epoch 12/15
33/33 [==============================] - 139s 4s/step - loss: 0.1111 - accura
cy: 0.9689 - val_loss: 0.1718 - val_accuracy: 0.9297
Epoch 13/15
33/33 [==============================] - 140s 4s/step - loss: 0.1083 - accura
cy: 0.9660 - val_loss: 0.1506 - val_accuracy: 0.9609
Epoch 14/15
33/33 [==============================] - 139s 4s/step - loss: 0.0826 - accura
cy: 0.9738 - val_loss: 0.0919 - val_accuracy: 0.9688
Epoch 15/15
33/33 [==============================] - 140s 4s/step - loss: 0.0544 - accura
cy: 0.9893 - val_loss: 0.1530 - val_accuracy: 0.9531
```

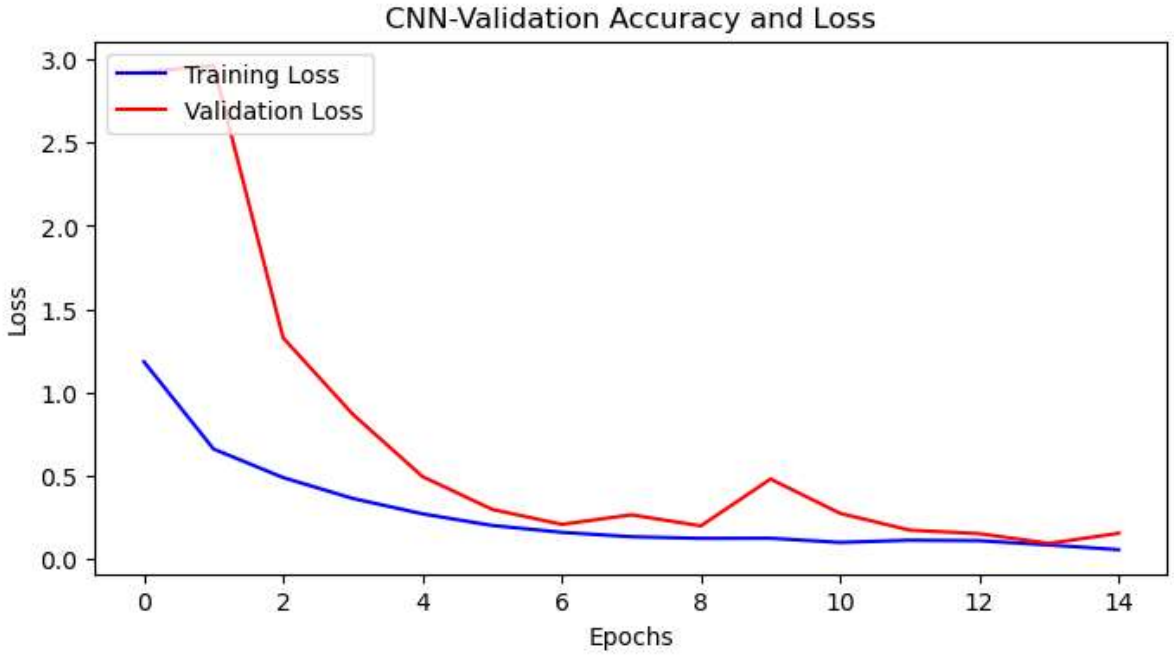
In [175]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
```

In [176]:

```
epochs=15
plt.figure(figsize=(8, 4))
plt.plot(range(epochs), acc, 'b',label='Training Accuracy')
plt.plot(range(epochs), val_acc, 'r',label='Validation Accuracy')

plt.xlim(0, 15)
plt.legend(loc='upper right')
plt.title('CNN-Training Accuracy and Loss')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')

plt.figure(figsize=(8, 4))
plt.plot(range(epochs), loss, 'b', label='Training Loss')
plt.plot(range(epochs), val_loss, 'r', label='Validation Loss')
plt.legend(loc='upper left')
plt.title('CNN-Validation Accuracy and Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```





CNN-LSTM TEST RESULT

In [177]:
```
test = tf.keras.preprocessing.image_dataset_from_directory(
    "Bananaleaf/test/",
    seed=123,
    shuffle=True,
    image_size=(254,254),
    batch_size=32
)
```

```
Found 128 files belonging to 3 classes.
```

In [179]:
```
yp = cnnlstmclassifier.evaluate(test)
```

```
4/4 [==============================] - 4s 1s/step - loss: 0.1730 - accuracy:
0.9453
```

CNN

In [6]:

```
input_shape = (254, 254, 3)
n_classes = 3

model=Sequential([

                Conv2D(32,3,activation='relu',input_shape=(254,254,3)),
                BatchNormalization(),
                MaxPooling2D((2,2)),

                Conv2D(128,3,activation='relu'),
                BatchNormalization(),
                MaxPooling2D((2,2)),

                Conv2D(256,3,activation='relu'),
                BatchNormalization(),
                MaxPooling2D((2,2)),

                Conv2D(64,3,activation='relu'),
                BatchNormalization(),
                MaxPooling2D((2,2)),

                Flatten(),
                Dense(64,activation='relu'),
                BatchNormalization(),
                Dense(n_classes,activation='softmax')

])
model.build(input_shape=input_shape)
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 252, 252, 32)      896

 batch_normalization_6 (Batc  (None, 252, 252, 32)     128
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 126, 126, 32)     0
 2D)

 conv2d_5 (Conv2D)           (None, 124, 124, 128)     36992

 batch_normalization_7 (Batc  (None, 124, 124, 128)    512
 hNormalization)

 max_pooling2d_5 (MaxPooling  (None, 62, 62, 128)      0
 2D)

 conv2d_6 (Conv2D)           (None, 60, 60, 256)       295168

 batch_normalization_8 (Batc  (None, 60, 60, 256)      1024
 hNormalization)

 max_pooling2d_6 (MaxPooling  (None, 30, 30, 256)      0
 2D)

 conv2d_7 (Conv2D)           (None, 28, 28, 64)        147520

 batch_normalization_9 (Batc  (None, 28, 28, 64)       256
 hNormalization)

 max_pooling2d_7 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 12544)             0

 dense_3 (Dense)             (None, 64)                802880

 batch_normalization_10 (Bat  (None, 64)               256
 chNormalization)

 dense_4 (Dense)             (None, 3)                 195

=================================================================
Total params: 1,285,827
Trainable params: 1,284,739
Non-trainable params: 1,088
_________________________________________________________________
```

In [60]:
```
model.compile(
    optimizer='adam',
    loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=False),
    metrics=['accuracy']
)
```

CNN TRAINING RESULT

In [61]:
```
import tensorflow as tf
tf.config.run_functions_eagerly(True)
history = model.fit(
    train_ds,
    batch_size=32,
    validation_data=val_ds,
    verbose=1,
    epochs=15,)
```

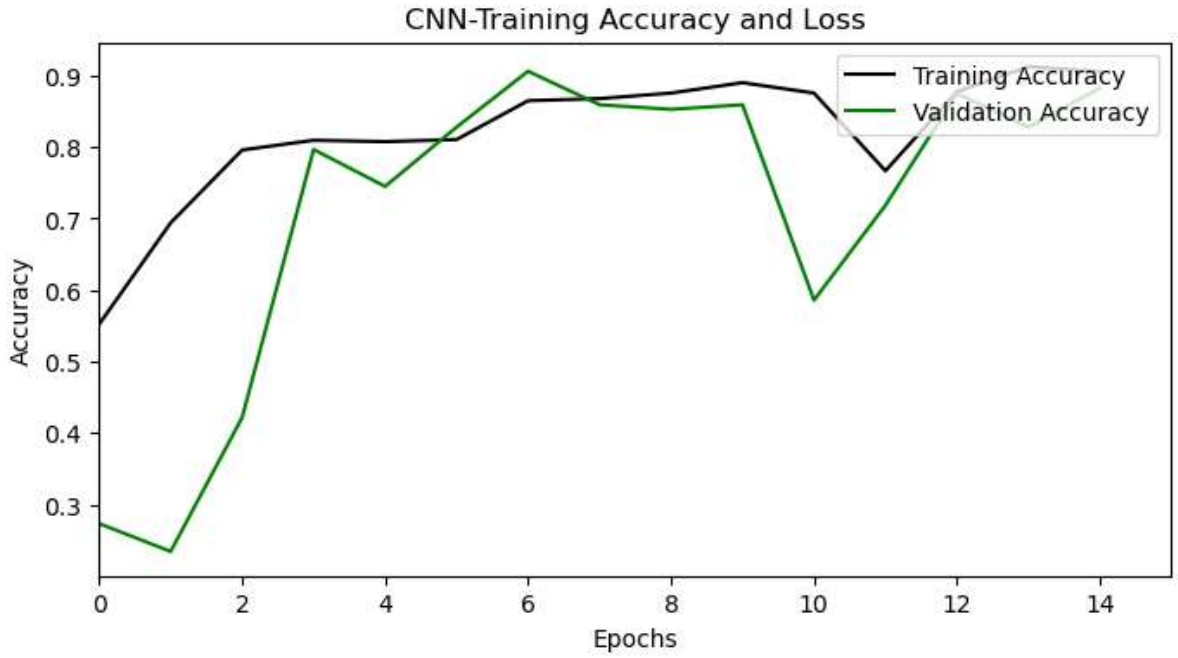
```
Epoch 1/15
33/33 [==============================] - 117s 3s/step - loss: 1.2003 - accura
cy: 0.5524 - val_loss: 14.8766 - val_accuracy: 0.2734
Epoch 2/15
33/33 [==============================] - 113s 3s/step - loss: 0.7037 - accura
cy: 0.6942 - val_loss: 2.6755 - val_accuracy: 0.2344
Epoch 3/15
33/33 [==============================] - 112s 3s/step - loss: 0.5237 - accura
cy: 0.7961 - val_loss: 1.7764 - val_accuracy: 0.4219
Epoch 4/15
33/33 [==============================] - 112s 3s/step - loss: 0.4630 - accura
cy: 0.8097 - val_loss: 0.5285 - val_accuracy: 0.7969
Epoch 5/15
33/33 [==============================] - 112s 3s/step - loss: 0.4518 - accura
cy: 0.8078 - val_loss: 0.7168 - val_accuracy: 0.7451
Epoch 6/15
33/33 [==============================] - 109s 3s/step - loss: 0.4614 - accura
cy: 0.8107 - val_loss: 0.5196 - val_accuracy: 0.8281
Epoch 7/15
33/33 [==============================] - 111s 3s/step - loss: 0.3548 - accura
cy: 0.8650 - val_loss: 0.2565 - val_accuracy: 0.9062
Epoch 8/15
33/33 [==============================] - 114s 3s/step - loss: 0.3446 - accura
cy: 0.8680 - val_loss: 0.4200 - val_accuracy: 0.8594
Epoch 9/15
33/33 [==============================] - 110s 3s/step - loss: 0.3335 - accura
cy: 0.8757 - val_loss: 0.3494 - val_accuracy: 0.8529
Epoch 10/15
33/33 [==============================] - 111s 3s/step - loss: 0.2698 - accura
cy: 0.8903 - val_loss: 0.3168 - val_accuracy: 0.8594
Epoch 11/15
33/33 [==============================] - 111s 3s/step - loss: 0.3817 - accura
cy: 0.8757 - val_loss: 1.0485 - val_accuracy: 0.5859
Epoch 12/15
33/33 [==============================] - 116s 3s/step - loss: 0.5825 - accura
cy: 0.7670 - val_loss: 0.5933 - val_accuracy: 0.7188
Epoch 13/15
33/33 [==============================] - 124s 4s/step - loss: 0.3345 - accura
cy: 0.8777 - val_loss: 0.3263 - val_accuracy: 0.8750
Epoch 14/15
33/33 [==============================] - 115s 3s/step - loss: 0.2327 - accura
cy: 0.9126 - val_loss: 0.4382 - val_accuracy: 0.8281
Epoch 15/15
33/33 [==============================] - 115s 3s/step - loss: 0.2422 - accura
cy: 0.9058 - val_loss: 0.3493 - val_accuracy: 0.8828
```

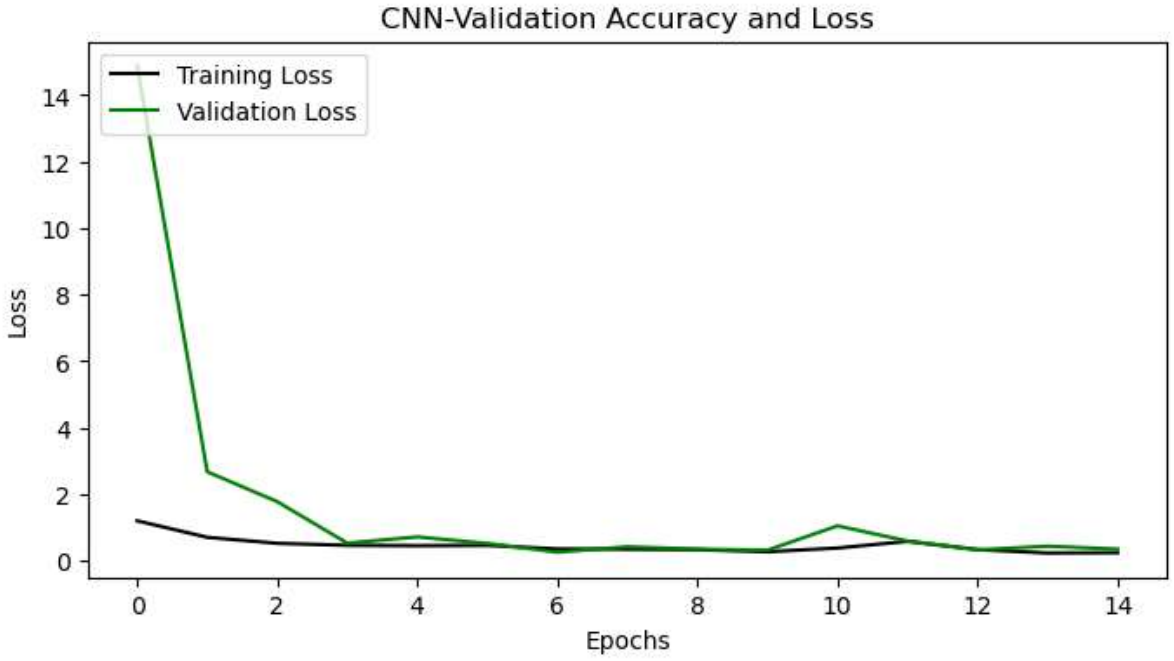
In [64]:

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']
```

In [67]:

```
epochs=15
plt.figure(figsize=(8, 4))
plt.plot(range(epochs), acc, 'k',label='Training Accuracy')
plt.plot(range(epochs), val_acc, 'g',label='Validation Accuracy')

plt.xlim(0, 15)
plt.legend(loc='upper right')
plt.title('CNN-Training Accuracy and Loss')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')

plt.figure(figsize=(8, 4))
plt.plot(range(epochs), loss, 'k', label='Training Loss')
plt.plot(range(epochs), val_loss, 'g', label='Validation Loss')
plt.legend(loc='upper left')
plt.title('CNN-Validation Accuracy and Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.show()
```





CNN TEST TESULT

In [68]:
```
test = tf.keras.preprocessing.image_dataset_from_directory(
    "Bananaleaf/test/",
    seed=123,
    shuffle=True,
    image_size=(254,254),
    batch_size=32
)
```

```
Found 128 files belonging to 3 classes.
```

In [69]:
```
yp = model.evaluate(test)
```

```
4/4 [==============================] - 4s 865ms/step - loss: 0.1960 - accurac
y: 0.9219
```

REFERENCES

1. Anasta, N., F. X. A. Setyawan, and H. Fitriawan. "Disease detection in banana trees using an image processing-based thermal camera." In IOP Conference Series: Earth and Environmental Science, vol. 739, no. 1, p. 012088. IOP Publishing, 2021.
2. Criollo, Antonio, Miguel Mendoza, Eduardo Saavedra, and Gustavo Vargas. "Design and evaluation of a convolutional neural network for banana leaf diseases classification." In 2020 IEEE Engineering International Research Conference (EIRCON), pp. 1-4. IEEE, 2020
3. Devi, M. Shyamala, Baddela Vamshikrishna, J. Arun Pandian, Paladugu Venkata Rao, and Shaik Rusum Yaseen. "Eight Convolutional Layered Deep Convolutional Neural Network based Banana Leaf Disease Prediction." In 2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), pp. 313-317. IEEE, 2022.
4. Faostat, https://www.fao.org/faostat/en/#data , 2019.
5. https://data.mendeley.com/datasets/rjykr62kdh/1
6. Jianqing, Huang, Yuan Qi, Liu Debing, and Zhang Jiarong. "Research on Banana Leaf Disease Detection Based on the Image Processing Technology." In 2022 IEEE 5th International Conference on Computer and Communication Engineering Technology (CCET), pp. 76-79. IEEE, 2022.
7. Krishnan, V. Gokula, J. R. V. P. Deepa, Pinagadi Venkateswara Rao, V. Divya, and S. Kaviarasan. "An automated segmentation and classification model for banana leaf disease detection." Journal of Applied Biology and Biotechnology 10, no. 1 (2022): 213-220.
8. Mahendran, T., and K. Seetharaman. "Banana Leaf Disease Detection Using Glcm Based Feature Extraction And Classification Using Deep Convoluted Neural Networks (Dcnn)." Journal of Positive School Psychology 6, no. 10 (2022): 2553-2562.
9. Niraj Chaudhari and Rahul Patil, “Classification, Detection and Diagnosis of Banana Leaf Diseases using Deep Learning Technique”, International Journal of Current Engineering and Technology, E-ISSN 2277 – 4106, P-ISSN 2347 – 5161, 2021
10. O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458 (2015).
11. Premakumar, K., and D. S. Khurdiya. "Effect of microwave blanching on the nutritional qualities of banana puree." Journal of Food Science and Technology-Mysore- 39, no. 3 (2002): 258-260.

12. Selvaraj, Michael Gomez, Alejandro Vergara, Henry Ruiz, Nancy Safari, Sivalingam Elayabalan, Walter Ocimati, and Guy Blomme. "AI-powered banana diseases and pest detection." Plant Methods 15 (2019): 1-11.

13. Singh, Balwinder, Jatinder Pal Singh, Amritpal Kaur, and Narpinder Singh. "Bioactive compounds in banana and their associated health benefits–A review." Food chemistry 206 (2016): 1-11.
14. Sun, Manli, Zhanjie Song, Xiaoheng Jiang, Jing Pan, and Yanwei Pang. "Learning pooling for convolutional neural network." Neurocomputing 224 (2017): 96-104.
15. Tuazon, Giorgette Louise H., Hazeline M. Duran, and Jocelyn F. Villaverde. "Portable Sigatoka Spot Disease Identifier on Banana Leaves Using Support Vector Machine." In 2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM), pp. 1-6. IEEE, 2021.
16. Vidhya, N. P., and R. Priya. "Detection and Classification of Banana Leaf diseases using Machine Learning and Deep Learning Algorithms." In 2022 IEEE 19th India Council International Conference (INDICON), pp. 1-6. IEEE, 2022.
17. Wang, Guan, Yu Sun, and Jianxin Wang. "Automatic image-based plant disease severity estimation using deep learning." Computational intelligence and neuroscience 2017 (2017).
18. Yu, Yong, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. "A review of recurrent neural networks: LSTM cells and network architectures." Neural computation 31, no. 7 (2019): 1235-1270.