HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

DEVELOPING IMAGE-BASED ENSET PLANT DISEASE IDENTIFICATION USING CONVOLUTIONAL NEURAL NETWORK

BY

UMER NURI MOHAMMED

A Thesis Submitted as a Partial Fulfillment to the Requirements for the Award of the Degree of Master of Science in Computer Science

to

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF INFORMATICS

August 11, 2020

APPROVAL

This is to verify that this thesis work is done by Mr. Umer Nuri Mohammed and submitted as partial fulfillment for the degree of Masters of Computer Science complies with the regulations of the University and meets the accepted standards concerning originality, content, and quality.

Singed by Examining Board

Main Advisor :	Signature,	Date
_____	_____	_____
Co-Advisor:	Signature,	Date
_____	_____	_____
External Examiner:	Signature,	Date
_____	_____	_____
Internal Examiner:	Signature,	Date
_____	_____	_____
Chairperson:	Signature,	Date
_____	_____	_____
Department Head:	Signature,	Date
_____	_____	_____
College Dean	Signature,	Date
_____	_____	_____

DECLARATION

I hereby declare that this thesis work was done by me, with the guidance of my advisors. The work contained herein is my own except where explicitly stated otherwise in the text, and that this thesis work has not been submitted, in whole or in part, for any other degree or professional qualification.

Submitted by

UMER NURI M.	_____	_____
Author	Signature	Date

Witnessed by

1.TESFAYE BAYU (PhD)	_____	_____
Advisor	Signature	Date

August 11, 2020

DEDICATION

TO KEDIR MOHAMMED

ACKNOWLEDGMENTS

*I would like first to thank **ALLAH** (s.w) fo giving me strength and ability to learn, understand and complete this work. All praises and thanks are due to **ALLAH** (s.w), the Lord of the Worlds. Without Allah's support, nothing would have been possible. After that. I would like to thank my main advisor, **Dr. Tesfaye Bayu** for his close supervision and constructive suggestion during my thesis work. He has been devoting his time and providing ideas to carry out the research. I thank you for your encouragement starting from the beginning to the end of this thesis work.*

*I would also like to express my gratitude to my co-advisor **Mr. Mekuannt Birara** from the computer science department at Hawassa University for his technical support and constructive guidance right from the implementation to the completion of my work. He has been devoting his time and providing ideas to carry out my research. I thank you for your encouragement starting from the beginning to the end of this thesis work.*

I am also very thankful to Mr. Shiferaw Mekonen and Tessema Tesfaye who are plant science experts at Southern Agricultural Research Institute at Hawassa for your help and support about the disease of enset and for helping me to get images of enset that are affected by different diseases.

Next, my deepest gratitude goes to my families for providing me with unfailing support and continuous encouragement throughout my life.

Finally, I would like to give my gratitude to people who are not mentioned in name but whose effort helped me much all along.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND ACRONYMS

DL	*Deep Learning*
CNN	*Convolutional Neural Network*
ADAM	*Adaptive Learning Rate Optimization*
ML	*Machine Learning*
ANN	*Artificial Neural Network*
BCE	*Binary Cross-Entropy*
CNN	*Convolutional Neural Network*
DBN	*Deep Belief Networks*
FC	*Fully Connected*
GPU	*Graphics Processing Units*
HOT	*Histogram Of Template*
ILSVRC	*ImageNet Large Scale Visual Recognition Challenge*
KNN	*K-Nearest Neighbor*
LSTM	*Long Short-Term Memory*
ML	*Machine Learning*
MSE	*Mean Squared Error*
SNNPR	*Southern Nations, Nationalities, and Peoples' Region*
ReLU	*Rectified Linear Unit*
RGB	*Red, Green, And Blue*
RNN	*Recurrent Neural Network*
SGD	*Stochastic Gradient Descent*
SVM	*Support Vector Machine*
ANN	*Artificial Neural Network*
BCE	*Binary Cross-Entropy*
CNN	*Convolutional Neural Network*

ABSTRACT

Nowadays, decline in food plant productivity is a major problem causing food insecurity to which plant disease is one of the factors. Early identification and accurate diagnosis of the health status of food plants is hence critical to limit the spread of plant diseases and it should be in a technological manner rather than by the labor force. Traditional observation methods by farmers or domain experts is perhaps time-consuming, expensive and sometimes inaccurate. Based on the literature, the literature suggests that deep learning approaches are the most accurate models for the detection of plant disease. Convolutional Neural network (CNN) is one of the popular approaches that allows computational models that are composed of multiple processing layers to learn representations of image data with multiple levels of abstraction. These models have dramatically improved the state-of-the-art in visual object recognition and image classification that makes it a good way for enset plant disease classification problems. For this purpose, we used an appropriate CNN based model for identifying and classifying the three most critical diseases of enset plants: - enset bacterial wilt, enset Leaf spot, and Root mealybug diseases. Enset is one of a major source of food in the South, Central and Southwestern parts of Ethiopia. A total of 14,992 images are used for conducting experiments including augmented images with four different categories; three diseased and a healthy class obtained from the different agricultural sectors stationed at Hawassa and Worabe Ethiopia, these images are provided as input to the proposed model. Under the 10-fold cross-validation strategy, the experimental results show that the proposed model can effectively detect and classify four classes of enset plant diseases with the best classification accuracy of 99.53%, which is higher than compared to other classical deep learning models such as MobileNet and Inception v3 deep learning models.

Keywords: *Convolution neural networks; Plant disease detection; Detection of enset plant diseases, bacterial wilt; leafspot; root mealybug.*

CHAPTER ONE
INTRODUCTION

1.1. Background

Enset plant is popular staple food for more than 20 million people [1, 5,6,40]. It is used as food (Bulla, Kocho, Amicho, etc.), clothes, beds, houses, medicine, cattle-feed, plates, fiber, and construction materials. Furthermore, in its wild form, it is native to the eastern edge of the Great African Plateau, extending northwards from South Africa through Mozambique, Zimbabwe, Malawi, Kenya, Uganda and Tanzania to Ethiopia, and west to the Congo, being found in high rainfall forests on mountains, and along forested ravines and streams [40]. However, this useful plant is attacked by different diseases that reduce the quantity and quality of production as well as the economy. The most three critical diseases of enset plants are enset bacterial wilt disease, the enset root mealybug disease, and enset leaf spot disease [2, 5, 6]. So that protecting enset plants from diseases is very essential to guarantee enset farm quality and quantity [2, 5]. The successful strategy of protection should start by early detection of the disease to take the proper treatment at the right time to prevent its spreading. Commonly, this detection is done by experts having academic knowledge through the practical experience of disease symptoms and causes [14, 15]. Moreover, these experts must monitor plants consistently to avoid disease spreading. This continuous monitoring represents a difficult and time-consuming task which lets to create the automation of enset plant disease detection and identification methods which is essential to protect enset plants [4, 12].

Previously several studies [4, 12, 14, 15] have proposed to detect and classify plant diseases using image processing and machine learning approaches. These approaches are trying to build plant disease classifiers using images taken from the fields. These classifiers are based on handcrafted features designed by feature engineering experts to extract the most important information for image classification. For this reason, these classifiers suffer from a lack of automation because of the hand-crafted features dependency [12,14]. Also, the classifier must be trained using images labeled by experts. Gathering these labeled images is very expensive because it is done manually. This difficulty of data collection has forced the earlier studies to use small datasets to train and test classifiers [4, 15, 14]. Using small-labeled datasets is a

limiting factor in machine learning and it can lead to overfitting. In recent years, convolutional neural networks(CNN) is adopted by the computer vision community that outperform the state-of-the-art in many fields. The main advantage of CNN in computer vision is the automatic feature extraction of images without any hand-crafted features. CNN classifiers are end-to-end systems that form features in a fully automated way without any intervention of human experts. In plant disease protection, many works have proposed that uses CNN architecture to detect and classify plant diseases. This new trend produced more accurate classifiers compared to traditional machine learning approaches. Despite these good results, CNN research in plant diseases remain immature and require more attention to produce practical systems. In the CNN plant disease recognition task using a contaminated image is divided into different phases such as image data collection for input, perform augmentation, feature extraction, classification, and giving probabilistic like output value.

Enset diseases are one of a major threat to food security in the South, Central and Southwestern parts of Ethiopia, but their rapid identification is difficult due to lack of the necessary modern techniques. The usual way of identification of disease in Ethiopia is observing the plant disease with the naked eye and further laboratory-based (microscopic) analysis. However, this method is time-consuming, expensive, and laborious. The enset disease early detection and management is one of the fundamental tasks for the Southern Agricultural Research Institute stationed at Hawassa. Therefore, the aim of this thesis is targeted to make its contribution by designing and developing CNN model that can detect and classify three most critical enset plant diseases appropriately based on images of infected leaves, stems, and roots that can be helpful for farmers, experts, and researchers to take an immediate measurement before the diseases bring total damage on the enset plant farm.

1.2. Statement of the problem

Plant diseases are major sources of famine and food insecurity. These cause serious losses in the quality, quantity, production, and economy in agriculture. Enset plant is the main cultivation in the central, south, and southwestern regions of Ethiopia [2,5,6]. Several challenges affect enset farms in the current area such as bacterias, viruses, and pests from these the most critical

disease are bacterial wilt, enset root mealybug and, leafspot disease. As researchers said that up to 80% of enset farms in Ethiopia are currently infected by this disease[5, 6].

Moreover, it is not possible for the pathologists to reach every farm and as even the crop pathologists rely on manual eye observation, the manual prediction method is not so accurate, it is time-consuming, and it takes a lot of effort. Therefore, developing an enset disease detection model that assists the farmers and domain area experts in early-stage with greater accuracy is a necessity. The Microscopic or lab-based solutions are effective to identify different kinds of diseases, but many of the domain area employees do not have access to use these diagnostics tools. Hence, several approaches have turned to computer vision approaches. The image-based deep learning models like CNN provide consistent, reasonably accurate, less time consuming, and cost-effective solutions for farmers to identify critical diseases of the plant. Therefore, the task identification of enset plant disease is a prerequisite necessity in the enset production system to fight or take measure the biotic factors at an earlier stage as soon as they occur. For that reason, this CNN model will be proposed to detect the most critical diseases specifically enset bacterial wilt disease, enset root mealybug disease, and enset leaf spot disease.

Several recent related studies have applied CNN models to identify plant disease problems using public datasets. However, even if several works have done so far, there is only one particular model is developed for enset disease identification using CNN that identifies only one disease namely enset bacterial wilt [48]. Additionally, this particular work needs classification accuracy results improvement. Moreover, there are no public datasets prepared before for three critical diseases of enset plant and Healy enset plant. Hence, this study is proposed to prepare a dataset to get more classification results that of related work as well as to design and develop a model that will identify and classify three critical diseases of enset plant from symptoms that appear on enset plant leave, root, and stem images. The research questions formulated for the proposed study are: Can CNN models be best to detect and classify proposed three diseases, what kind of strategies could be used to identify enset plant disease in current time, and how an appropriate dataset is collected in order to accomplish this thesis work.

1.3. The objective of the research

1.3.1. General objective

The general objective of this thesis work is to design and develop enset plant disease identification model using a convolutional neural network.

1.3.2. Specific objectives

The specific objectives of this research are to:

- Collect three critical disease of enset images as well as healthy enset plant images.
- Design and develop a model to identify and classify the three critical diseases of enset.
- Experiment and evaluate a model using the prepared enset dataset.
- Evaluate the predictive performance of the model.
- Undertake a comparative evaluation of the results to conclude the study.

1.4. Scope and Limitations of the research

As any thesis works, this thesis work is not free from limitations. Due to limited time, financial resources, and lack of sufficient dataset the scope of this thesis is up to designing and developing a CNN model with the capability of identifying and classifying only three most affecting diseases of enset plant namely Enset bacterial wilt disease, enset root mealybug disease, and enset leaf spot disease. In addition to this, it will not take any countermeasures for the disease. On the other hand, to make the study more manageable and to complete the study within the given time, this thesis is delimited to only designing and developing the CNN model.

1.5. Significance of the research

Problems mentioned in the statement of the problem will be addressed by developing a CNN model which has capabilities of detecting and classifying of enset disease. In general, at the successful completion of this thesis, the following expected outcomes are included: -

- In the existing work process, domain experts use different methods like Microscope or lab-based methods to detect enset plant disease. So that the model will replace these tools to detect Enset plant disease.
- The model will have capable of detecting enset plant disease at the earlier stage as soon as it occurs. Therefore, saving the lost and reducing the dependency on the expert to a

certain extent will possible. It will provide help for a person having less knowledge about the enset disease.

- The model will serve as a guideline tool and will suggest recommended treatment methods for possible signs and symptoms for domain users.
- The model will be used to fill the gap of lack of domain experts in a particular area.
- The model will be used as a teaching tool for domain area students on how to detect enset disease symptoms.
- The model will be applied to any artificial intelligence machines like drones and robots.
- Any government and private agricultural organizations will benefit from this work.
- Finally, this study will serve as reference material for other researchers who will want to conduct further research into the problems related to the domain area.

1.6. Organization of the thesis

The remaining part of the thesis work is organized as follows. Chapter two covers a review of general and related literature in which different concepts and approaches related to this thesis work are presented which are done by other researchers using machine learning and CNN to detect plant disease. Chapter three covers the methodologies that we are followed to conduct this thesis including data collection methods, tools that are used to implement the model, and the evaluation techniques are discussed. Chapter four deals with the design of the proposed model. It presents the general architecture of the proposed model with its basic components and the discussion of the components with their interaction in the model. Chapter five focuses on the detailed experimental results and discussions of the proposed model. Chapter six presents the conclusion and future work recommendations for the improvement of this work.

CHAPTER TWO

LITERATURES REVIEW

In this chapter, research works related to the objective of this thesis work written on plant disease detection and classification are reviewed to have a detailed understanding of the methods on domain area. It including a description of the enset plant, three critical diseases of enset plant, data collection and preparation methods, the architecture of the CNN, the summaries of related works, and their main gap which should be solved in this thesis work.

2.1. The disease of Enset plant

Enset diseases are one of the threats to food security in SNNPR of Ethiopia their rapid identification is difficult due to lack of the necessary modern techniques. The most affecting disease are enset bacterial wilt, enset root mealybug, and enset leafspot [2,5,6]. In the following sections, this critical disease of enset, as well as healthy enset, is discussed briefly.

2.1.1. Enset bacterial wilt disease

Enset bacterial wilt disease is the very destructive disease of enset plant whose scientific name is Xanthomonas campestris pv. Musacearum. It kills the enset plant at all stages of growth and currently found in all of the enset growing areas [2, 3]. The appearance of the bacterial wilt is taking place in the apical leaves of enset that will wilt then dry and finally lead to the killing of the whole plant [2,5]. First, an infected enset plant changed to a yellow orange color. Then the leaves will wilt, bend over, wither, and causes the death of the enset plant. Infected farm tools, infected planting material, repeated transplanting, animals feeding on infected plants, splashing rain, Birds, fruit flies, Wind-blown, and Lizards can spread bacterial wilts [6].

Figure 1.Sample enset bacterial wilt disease images

Figure 1 shows us a sample enset plant image infected by enset bacterial wilt. In this work for experimenting, a total of 3863 such images are used from different species of enset plants. 3476 images are chosen for training and rest 387 images are used for the validating model.

2.1.2. Enset root mealybug

The second critical disease that seriously attacks the enset plant is enset root mealybug disease whose scientific name is cataenococcus [2]. This disease is caused by pests that attack enset plants at any age but extremely serious on 2 to 4-years-old enset plants. The appearance of the enset root mealybug is taking place in the roots of enset that rotten root and finally lead to the killing of the whole plant [2,5]. Machinery, tools, equipment, and soil movement can distribute enset root mealybug disease during cultivation. Figure 2 shows us the sample enset root image infected by enset root mealybug disease. In this thesis work for experimenting, a total of 3979 enset root mealybug disease images are used from different species of enset. A total number of 3581 images are chosen for training and rest 398 images are used for validating the model.

Figure 2.Sample enset root mealybug disease image

2.1.3. Enset leafspot disease

The third critical disease that seriously attacks the enset plant is enset leafspot disease. Leaf spot diseases caused by the fungi whose scientific name is phyllosticta sp., Pyricularia sp., and drechslera sp. This disease commonly affects suckers, seedlings, and young enset plants. The following Figure 3 shows us sample enset leaf image that infected by enset leafspot disease. In this thesis work for experimenting, a total of 2808 enset root mealybug disease images are used from different species of enset. A total number of 2527 images are chosen for training and rest 281 images are used for validating the model.

Figure 3: Sample enset leafspot disease image

2.2. Healthy Enset plant

Enset is a miracle staple food plant of several millions of people; more than 20% of the population of Ethiopia depends upon this crop [1]. It is used as food (Bulla, Kocho, Amicho, etc.), clothes, beds, houses, medicine, cattle-feed, plates, fiber, and construction materials. Enset based farming system is an indigenous and sustainable agricultural system in Ethiopia. More than 300,000 hectares of land is assumed to be covered with enset cultivation [5,6]. It has a major economic and socio-cultural importance for a wide range of smallholder households in the South, Central, and Southwestern parts of Ethiopia. In addition to this, enset constitutes one of the cheap sources of carbohydrates in Ethiopia as well as providing more calcium and iron than most cereals, tubers, and root crops. Currently, Enset is also being used for textile, paper, and adhesive industries in saving the foreign exchange rate which could have been spent to import the raw material [6].

Figure 4:Sample healthy enset plant images

Figure 4 shows us the sample healthy enset plant images. In this thesis work for experimenting, a total of 4,342 healthy enset plant mages from different enset species are used which are

divided into two parts a total number of 3907 images are chosen for training and 435 images are chosen for the testing.

2.3. Convolutional Neural Network

A convolutional neural network is one of the most popular algorithms for deep learning mostly used in image recognition, image clustering (photo search), and classification (e.g. name what they see), object detection within scenes (real-time), that are specifically designed to process pixel data. Based on various improvements (like parameters optimizations, regularization, structural reformulation, etc.) CNN can be broadly categorized into seven different classes namely: spatial exploitation, depth, multi-path, width, channel boosting, feature map exploitation, and attention-based CNN. The taxonomy of CNN architectures is shown in Figure 5[9]. CNN has an inbuilt automatic multi-stage feature learning process that learns rich hierarchical representations (i.e. features). CNN detects image pixels, edges, textures, motif, parts, and objects of features in the image and converts them into a map of numbers. These a map of numbers are then processed and fed into an artificial neural network that can learn from them and make predictions. Unlike other machine learning approaches, CNN learns image features directly from raw image data, using patterns to classify images and eliminating the need for manual feature extraction.

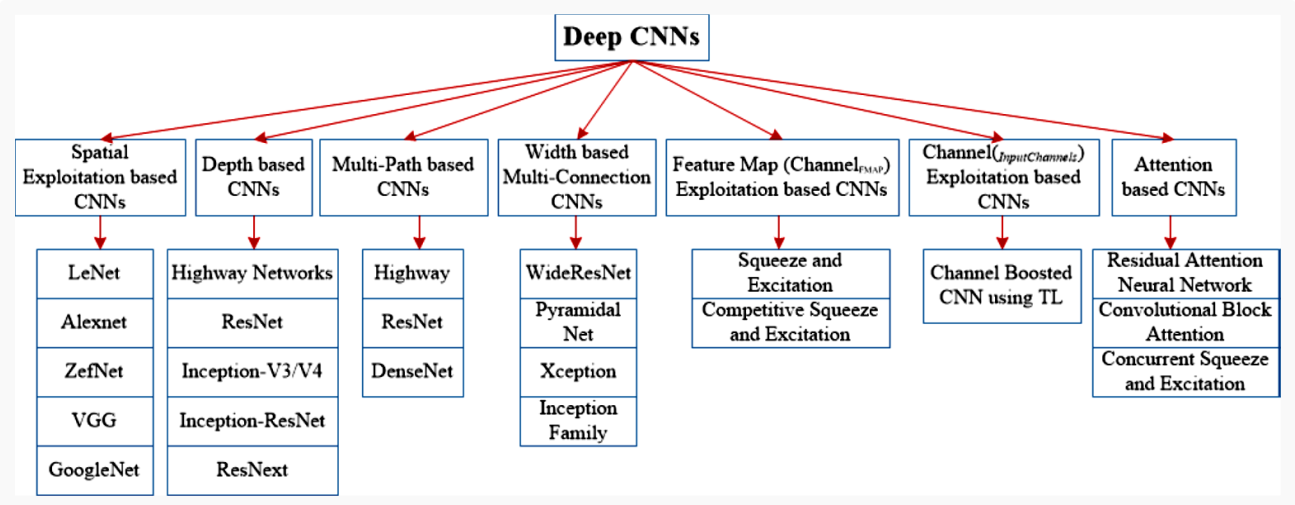


Figure 5: Taxonomy of CNN architectures

The availability of a large amount of data and improvements in the hardware processing units have accelerated the research on CNN. Figure 6 shows us the basic layout of a typical CNN image recognition task [9,10,11].

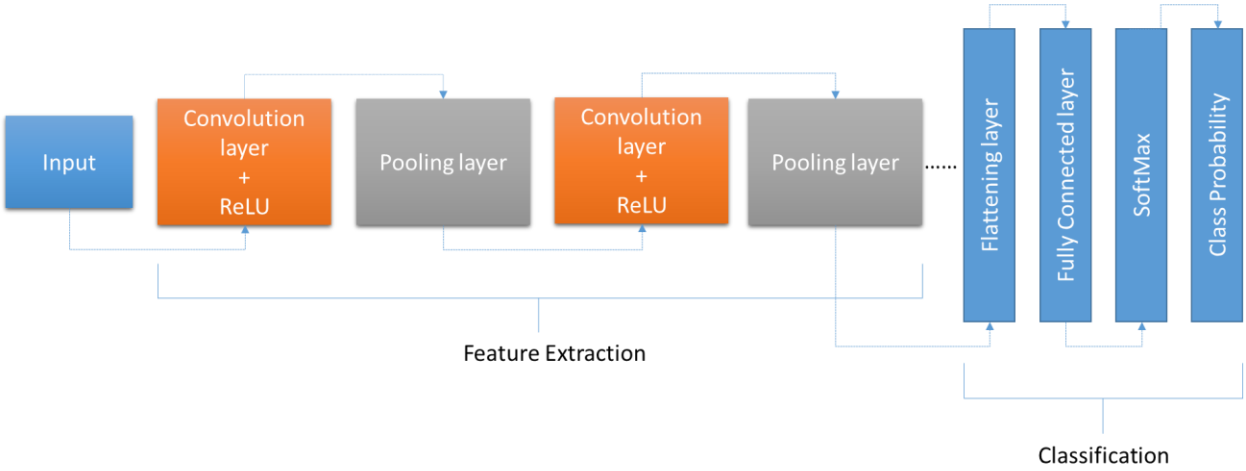


Figure 6.Typical CNN image recognition task

Using CNN for deep learning has become increasingly popular due to three important factors: firstly, CNN eliminates the need for manual feature extraction, the features are learned directly by the CNN from raw image data, secondly, CNN produce state-of-the-art recognition results, and thirdly, CNN can be retrained for new recognition tasks, enables to build on pre-existing networks. Depending on the application type, researchers can build a CNN from scratch, or use a pre-trained CNN model with an existing dataset. Similar to all deep learning techniques, CNN is very dependent on the size and quality of the training data. With a given well-prepared dataset, CNN is capable of surpassing humans at image recognition tasks. CNN uses relatively less preprocessing when compared with other algorithms of image processing. The connectivity pattern of the CNN looks like the structure of the human visual cortex. CNN consists of different layers. They are the input layers and output layers, and between these layers, there are multiple hidden layers. There is no limitation for hidden layers present in this network. Generally, the CNN image recognition task is divided into four phases: phase one: is related to dataset gathering for input, phase two: is related to performing augmentation, phase three: is related to feature extraction whereas phase four: classification which is related to giving probabilistic like output value. In the following sections, the basic layers of the CNN architecture are presented briefly.

2.3.1. Input layer

The input layer of a neural network is composed of artificial input neurons and it accepts the initial pixel image dataset in the form of arrays into the hidden layers for further processing [9]. Before starting the convolution operation, the input layer contains images as pixel values for all CNN based methods. When the input image is grayscale, the input shape will be $P \times P$ dimensions. Considering the color images, the shape will be $P \times P \times (N = 3)$ where N defined as total channel numbers [39].

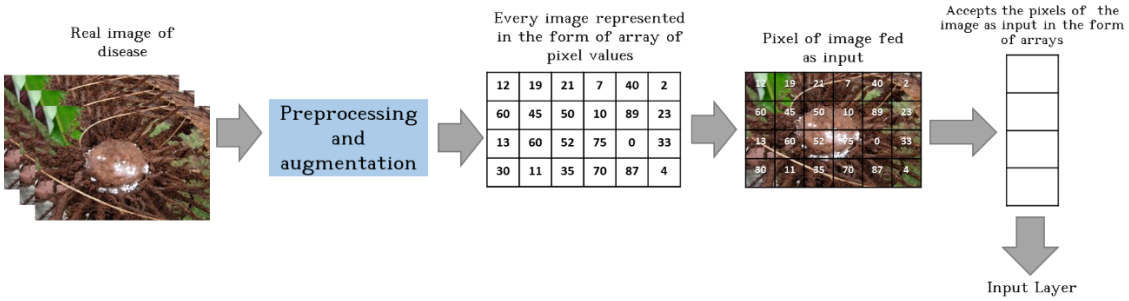


Figure 7:Sample input layer of CNN

2.3.2. Convolution Layer

The convolutional layer includes a set of convolutional kernels, which are associated with a small area of the image known as a receptive field. It is used to extract useful features from the input image. The output of convolution operation is the multiplication of weights and the corresponding inputs in the sliding window [9,10,11]. First, an image will be pushed to the network this is known as an input image. Then, the input image will go through (sliding) an infinite number of steps this is known as the convolutional part of the network. Finally, the neural network will predict the digit (pattern) on the image [10].

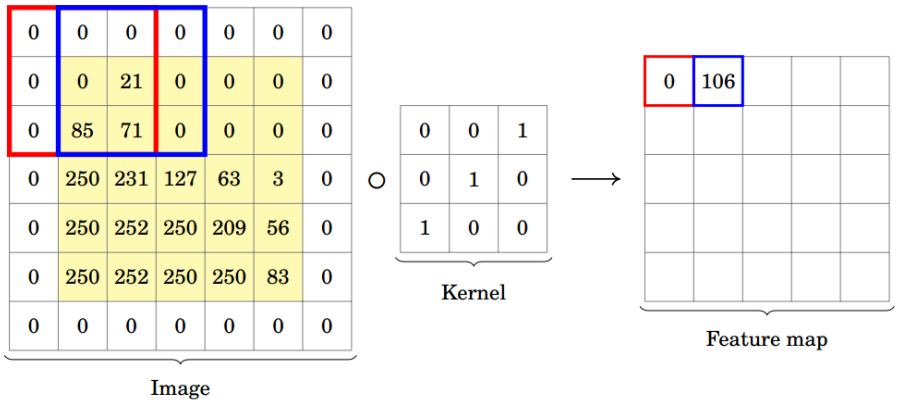


Figure 8. A typical convolution Example

2.3.3. Pooling Layer

The pooling layer is used to reduce the dimension of the representatives in the image dataset from the convolution layer and creates a smaller sample to speed up calculations. There are different types of pooling layers such as max-pooling that keeps the maximum values from the particular shape of the filter, average pooling that deals with an average value, and min pooling which takes the minimum value of this filter [9,10,11,48]. Figure 9 shows us the example of a max-pooling operation that reduces 4 by 4 image to 2 by 2 image.

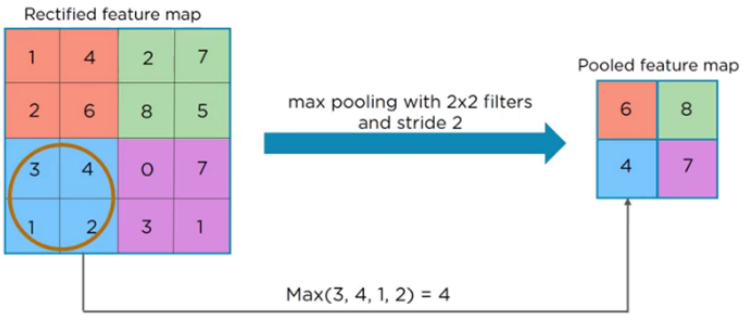


Figure 9.Example of max pooling

2.3.4. Flattening layer

Flattening is the process of converting all the results of a 2-dimensional array from pooled feature map into a single long continuous linear vector to create fully connected layers. In other words, it is the process of putting all the pixel data in one line and makes connections with the final layer so that this layer accomplishes this task [9,10,11].

Figure 10:Sample Flattening Layer

2.3.5. Fully Connected Layer

The fully connected layer is found at the end of the neural network which is used for classification purposes. It will take an input dataset from the previous layers and analyze the output of all previous layers globally. Also, it makes a non-linear combination of selected features, which are used for the classification of data. unlike pooling and convolution layers, it is a global operation that uses activation functions like SoftMax activation and Sigmoid activation function to classify the number on the input image [9,10,11]. Most of the time the SoftMax activation function is used for categorical classification [29,31,32] whereas Sigmoid Sigmoid activation function used for binary classification to compute the class's scores. The input of the SoftMax classifier is a vector of features resulting from the learning process and the output is a probability that an image belongs to a given class [9,10,11,48]. In a fully connected layer, every neuron in the previous layer is connected to every neuron in the next layer. This layer accepts the output of the convolution or pooling layer which is high-level features of the input volume.

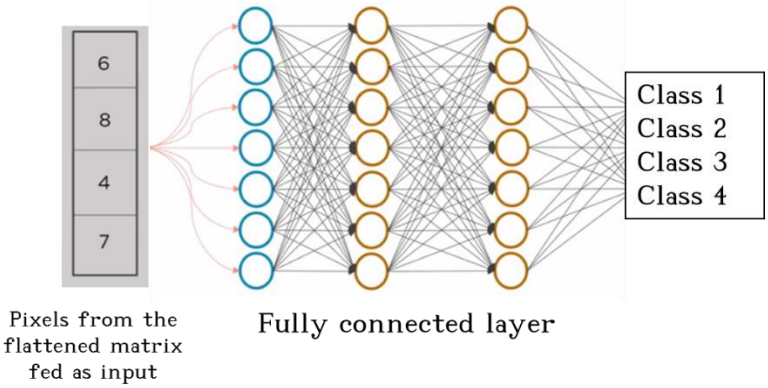


Figure 11. Example of a fully connected layer

2.3.6. Hyper Parameters

Hyperparameters are values that are decided outside of the model training process that determine the network structure and behavior of a deep learning algorithm which is very specific to each model such as activation functions, learning rate, epochs, optimization algorithm, batch size, and loss function [37,38,39,48].

2.3.6.1. Activation Functions

The activation functions control how the signal flows from one layer to the next, emulating how neurons are fired. Output signals which are strongly associated with past references would activate more neurons, enabling signals to be propagated more efficiently for identification. CNN is compatible with a wide variety of complex activation functions to model signal propagation, there are three most common activation functions in CNN such as ReLU, SoftMax, and Sigmoid which are applied in both convolution layers and output layers [9,10,11,48.

2.3.6.1.1. ReLU

To make up a convolution layer, activation functions like ReLU will be added to replaces the entire negative pixel value with Zero (0) that is performed after every convolution to introduce nonlinearity. The ReLU is a very popular activation function which is defined as f (x) = max (0, x) where x is the input to a neuron [9,10,11,48].

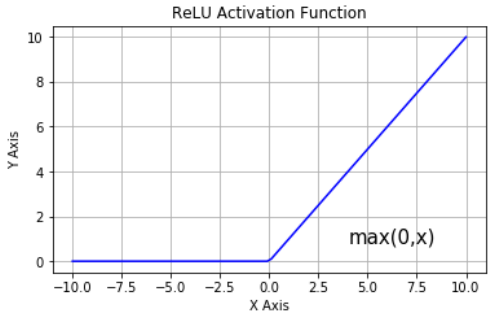


Figure 12: ReLU activation function

2.3.6.1.2. SoftMax

In deep learning models, the SoftMax function is the last layer which is used to compute the class's scores. The input of the SoftMax classifier is a vector of features resulting from the learning process and the output is a probability that an image belongs to a given class [9,10,11].

The SoftMax activation function uses to get the resulting input from the process of learning and gives the probability output which is needed to classify images more accurately which is mostly used for binary. The softmax activation function has the following formula.

$$\sigma(z)_j = \frac{e_j^z}{\sum_{k=1}^{K} e_k^z}$$

Figure 13:SoftMax activation function

2.3.6.1.3. Sigmoid

The sigmoid function is an activation function in terms of underlying gate structured in correlation to Neurons firing, in Neural Networks which is mostly used for binary classification. The derivative also acts to be an activation function in terms of handling Neuron activation in terms of NN's. By using the sigmoid activation function the last layers (output layer) of the fully connected layer perform classification (probabilities of inputs being in a particular class) based on the training data. The Sigmoid activation function has the following mathematical formula.

$$f(x) = \frac{1}{(1+exp^{-x}}$$

Figure 14:Sigmoid activation function

The learning rate is the rate at which a function moves through the search space. A small learning rate leads to more precise results but it requires more training time. The optimization algorithm is used in CNN models to learn the best set of weights and biases of the neural network that minimize the loss function by randomly choosing a small number of training inputs. The Epochs are the number of iterations the entire dataset passes forewarned and backward through the model or the network. Batch size is the number of input data we pass into the network at once. It is too hard to give all the data to the computer in a single epoch so we need to divide the input into several smaller batches. It is preferred in model training to minimize the computational time of the machine[9,10,11,48].

2.3.6.2. Evaluation metrics

Accuracy:- an accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. If we have high accuracy, then our model is best [49].

Accuracy = TP+TN/TP+FP+FN+TN

Loss:-It is a summation of the errors made for each example in training or validation sets. In this thesis work, we have used Categorical cross-entropy[48]. It's defined as:

$$\text{Categorical cross entropy} = \sum_{j=0}^{M} * \sum_{i=0}^{N} (yij * \log(yij))$$

Precision: - Precision is the ratio of correctly predicted positive observations of the total predicted positive observations[51]. High precision relates to the low false-positive rate.

Precision = TP/TP+FP

Where TP is True Positives - These are the correctly predicted positive values which mean that the value of the actual class is yes and the value of the predicted class is also yes. FP is the False Positives- When the actual class is no and the predicted class is yes.

Recall:- Recall is the ratio of correctly predicted positive observations to all observations in actual class – yes[51].

Recall = TP/TP+FN

Where TP is the number of true positives and FN is the number of false negatives.TN is the True Negatives -These are the correctly predicted negative values which means that the value of the actual class is no and the value of the predicted class is also no.

F1 Score:-F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account[51]. Accuracy works best if false positives and false negatives have a similar cost. If the cost of false positives and false negatives are very different, it's better to look at both Precision and Recall.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

Confusion matrix:- a confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. It shows how the classification model is confused when it makes predictions[50].

2.3.7. Popular CNN models

In recent years, several CNN models are developed based on from scratch and as transfer learning strategies. AlexNet model [30,33,34,36,39], ResNet model [19,32], GoogLeNet model

[21,23], VGGNet model [29,31,36,16,19], LeNet model [17,35], Cifar-10 model [37,38], and Inception3 model [28] are most popular models; most of the plant disease detection models developed based on these architectures. These architectures are used previously with success in computer vision challenges like ImageNet [22,25,26,27]. They are trained more than 1.3 million high-resolution images to recognize 1000 different objects which are composed of a depthwise convolutional layer, Max-pooling layer, and a fully connected layer having a rectifier activation function and a SoftMax activation function at the output layer to turn the outputs into probability-like values and allow one class to be selected as the model's output prediction with loss function and an adaptive learning rate optimization like adam to learn the weights.

2.3.8. Related Works

In this chapter, different previous deep learning approaches particularly CNN based models are reviewed that are more related to this thesis work. In recent years, several CNN models are developed for different plant disease detection and classification which serve as an expert assistance tool. Such tools could contribute to more sustainable agricultural practices and greater food production security. In the following, the literature which is more related to this thesis work is reviewed.

In [17] a model for soybean plant disease identification have proposed using the convolutional neural network, in this study, the new model is designed based on the LeNet architecture to perform the soybean disease classification. PlantVillage database with 12,673 sample images containing leaf images of four classes, including the healthy leaf images were used. As researchers say, experiments implemented on their model achieve 99.32% classification accuracy which shows that CNN can extract important features and classify automatically plant diseases from images.

In [19] a model for corn disease identification from leaf images have proposed using convolutional neural networks, in this paper, a model is developed followed by ResNet50 and the InceptionV3 architectures to classify images of corn leaves into four categories namely common rust disease, northern leaf blight disease, gray leaf spot disease, or healthy. The dataset taken from plantvillage.org consists of 8506 images of corn leaves from 4 categories. The proposed model evaluated with 5-fold cross-validation achieved an accuracy of 97.09% on

average as researches said. The result of this model was slightly worse than the performance of other famous CNN architectures, but the complexity of this architecture was much lower. Data augmentation was done using KerasImageDataGenerator; the proposed network architecture was based on the bottleneck architecture and the MobileNet architecture.

In [28] a model for diagnosing plant disease have proposed using Convolutional Neural Networks in 2019. In this paper, 54,306 public images are classified into 38 classes from the PlantVillage dataset with 26 diseases, and 14 crop species are used. Images are split into training, validation, and test datasets. The model is developed based on InceptionV3 architecture. For the testing phase network weights with the lowest validation loss are used. A Python library called Keras with TensorFlow backend is used as a framework to Train CNN. The pixel values of input images are divided by 255 so that they range within [0.0,1.0]. The network is initialized with random weights. Using a categorical cross-entropy loss metric network weights are optimized using the Adam optimization algorithm with a learning rate of 0.05. In this work, a set of 128 images are fed to the network as a batch per Fold.

In [29] a model for mango leaf disease Identification has proposed using a convolutional neural network. In this work, researchers are used three hidden layers to extract features of mango leaf disease inspired by VGG-16, VGG-19, AlexNet. Five different mango leaf diseases database consisting of 1200 images of diseased and healthy mango leaves is created by acquiring images under challenging conditions. The proposed model has achieved higher accuracy compared with available popular models.

In [30] a model for rice plant disease classification has proposed using transfer learning of deep convolution neural network. In this work, a total of 619 rice plant diseased images from the real field belong to four classes such as Rice Blast, Bacterial Leaf Blight, Sheath Blight, and Healthy Leave are collected and a pre-trained deep convolutional neural network (CNN) like AlexNet Architecture as a feature extractor and Support Vector Machine (SVM) as a classifier are used. The proposed model has achieved higher accuracy compared with available popular models.

In [31] a model for the classification of tobacco leaf pests have proposed using VGG16 transfer learning. In this work, a total of 1500 tobacco plant diseased images from the real field are

collected and a pre-trained VGG16 is used. In this study, the same feature extraction section is used for extracting new dataset features. The classifier section is replaced with a new FC that adjusts to the number of classes on a new dataset so that the number of outputs is 3. The proposed model able to classify the types of pest attacks well with very high accuracy. A greater number of learnable parameters have improved reliability performance. Adam optimizer is showed better results than SGD with a few and many learnable parameters.

In [32] a model for rice disease detection has proposed using a deep convolutional neural network. In this work, a dataset of 500 images of healthy and diseased samples are collected and the model is trained to identify the three common diseases of rice. A percentage of all the images are randomly chosen for each class and put them into the training sets. Similarly, another 15% of the images of each class are put into the validation set and the rest of the images are put into the test set with a Python script divider. In this manner, several experiments are conducted to improve the accuracy of the identification of rice diseases. Three CNN architectures are trained with three different training strategies namely baseline training, fine tuning, and transfer learning with the same training parameters. Among these training strategies, fine tuning is the most successful learning strategy in detecting rice plant diseases for three chosen CNN architectures. The ResNet50 achieved the highest accuracy. Furthermore, VGG16 achieved the lowest accuracy.

In [33] a model for apple leaf disease identification has proposed using deep convolutional neural networks. In this work, a dataset of 13,689 images of four common apple leaf diseases and healthy apple leaves are used. This study proposed sufficient pathological images and designed a novel architecture based on AlexNet architecture by removing partial fully connected layers, adding pooling layers, introducing the GoogLeNet Inception structure into the proposed model, and NAG algorithm is applied to optimize network parameters to accurately identify the apple leaf diseases. The proposed model is trained to identify the four common apple leaf diseases and tested under the hold-out test set. Compared with the original AlexNet model, the proposed model reduces the number of parameters greatly has better accuracy. Furthermore, other learning models such as SVM and BP neural networks, standard AlexNet, GoogLeNet, ResNet-20, and VGGNet-16 are trained on the expanded dataset to compare results of them with the proposed model.

In [34] a model for tea leaf disease recognition has proposed using a convolutional neural network. In this study, a CNN model named LeafNet is developed with different sized feature extractor filters that automatically extract the features of tea plant diseases from images. Additionally, the performance of two feature extraction methods and three classifiers are individually evaluated in their abilities to identify seven tea leaf diseases. The proposed leafNet model achieved the highest accuracies compared to SVM and MLP classification algorithms. The proposed model is an improvement of the classical AlexNet model, its network constructed by reducing the number of convolutional layer filters and the number of nodes in the fully connected layer, thereby effectively reducing the number of network parameters requiring training and reducing the overfitting problem as well as reduce the computational complexity. Particularly, the proposed model contains five convolutional layers, two fully connected layers, and a final layer as the classification layer. Also, the number of filters in the first, second, and fifth convolutional layers is designed to equal half of those used in AlexNet's filters. Further, the number of neurons in the fully connected layer is 500, 100, and seven, respectively, which differs from the original AlexNet architecture.

In [35] a model for tomato leaf disease detection has proposed using convolutional neural networks. The implementation of the proposed model consists of around 18160 images belonging to 10 different classes of tomato leaf diseases. Out of the 18160 images,4800 images are set aside for testing and 13360 images are used for training. The architecture used is a simple convolutional neural network with the minimum number of layers with a slight variation of LeNet architecture to classify the tomato leaf diseases into 10 different classes. In this study, several experiments have conducted with architectures like AlexNet, GoogleNet and the best results could be seen with the use of a variation of the LeNet architecture. This proposed model has achieved good accuracy compared with these models even under unfavorable conditions. The proposed architecture consists of an additional block of convolutional, activation, and pooling layers in comparison to the original LeNet architecture. Each block consists of a convolutional, activation, and max-pooling layer. Three such blocks followed by fully connected layers and Softmax activations are used in this architecture. Convolutional and pooling layers are used for feature extraction whereas the fully connected layers are used for classification. Activation layers are used for introducing non-linearity into the network.

In [36] a model for tomato crop disease classification has proposed using a pre-trained deep learning algorithm. In this study, 13,262 images are used for common 6 classes of diseases, and a healthy image of tomato leaves obtained from the PlantVillage dataset is provided as input to AlexNet and VGG16 net. The last layer of AlexNet and VGG16 has replaced with the output layer which is equal to the number of classes. In this case, it is 7 where it includes 6 diseases and a healthy class. Also, the Softmax layer and a fully connected layer is added to the architecture. Therefore, the last three layers of both models have been modified. Moreover, the classification accuracy of AlexNet using 13,262 images is better than that of the VGG16 net. In addition to this, the performance of both models is evaluated by modifying the number of images, setting various minibatch sizes, fine tuning, varying the weight and bias learning rate.

In [37] a model for the identification of tea leaf diseases has proposed using an improved deep convolutional neural network. In this study, a multiscale feature extraction module is added to the CIFAR10-quick deep learning model to improve the ability to automatically extract image features of different tea leaf diseases. The standard convolution in the multi-scale feature extraction module is changed to depthwise separable convolution to reduce the number of model parameters. Four models with different structures are obtained by replacing the first, second, third, and all three convolutional layers of the CIFAR10-quick model with the multiscale feature extraction module. These four models are named as Alter-first, Alter-second, Alter-third, and after-all, respectively. The most effective model is selected through experiments. Several experimental comparisons have conducted, results show that the average identification accuracies between the proposed improved model are higher than that of the original CIFAR10-quick model, traditional machine learning models such as BP neural network, Bayesian classifier, SVM, KNN, and classical CNN models such as LeNet-5, AlexNet, and VGG16.

In [38] a model for the Identification of maize leaf diseases has proposed using improved deep convolutional neural networks. In this study to improve the identification accuracy and to reduce the number of network parameters of 9 kinds of maize leaf diseases two improved models namely GoogLeNet and Cifar10 have used by adjusting their parameters, changing the pooling combinations, adding dropout operations and rectified linear unit (ReLU) functions, and reducing the number of classifiers. For the GoogleNet model, only the first classifier is used to train and test, which reduces the number of model parameters and the time required for

convergence without affecting the recognition accuracy. For the Cifar10 model, the structure is optimized by adding dropout and ReLU between the two fully connected layers. A dropout operation with an appropriate probability value can prevent overfitting of CNNs. During the recognition time improved GoogLeNet model achieves higher identification accuracy compared with the improved Cifar10 model.

In [39] a new model for automatic recognition of guava leaf diseases has proposed using a deep convolution neural network. In this study, eleven layers based D-CNN model which contains 4 convolution layers, 4 max-pooling operations with rectified linear unit function, and 3 fully connected layers which are developed followed by the AlexNet model to identify major diseases of guava leaf such as Algal Leaf spot, Whitefly, and Rust. The stochastic gradient descent (SGD) optimization algorithm with some hyperparameters which randomly select the training inputs, learning rate, weights, and bias values are selected. To reduce the overfitting problems, the data augmentation techniques are used with python scripts. Therefore, the proposed model gives the highest accuracy of both testing and training phases comparing to some popular models such as AlexNet and LeNet-5 and some popular feature descriptors namely Local Binary Pattern (LBP), CENTRIST, Local Gradient Pattern (LGP) and Noise Adaptive Binary Pattern (NABP).

In [45] a model is developed for the detection of black Sigatoka on the banana trees using image processing techniques. The paper proposes a system for pest detection on the infected images of Banana leaf. Images of the infected leaf are captured by a digital camera and processed using image growing, image segmentation techniques to detect infected parts of the particular plants. Then the detected part processed for further feature extraction which gives a general idea about pests. This paper proposes automatic detection and calculating the area of infection on leaves of a black Sigatoka at a mature stage on a banana tree.

In [46] a deep learning-based model is developed for banana leaf disease classification. This research paper identifies and classifies the disease caused by fungi and the disease is known as banana Sigatoka and banana Speckle. In this study, a globally available dataset from the PlantVillage project is used to collect the infected and healthy leaf images of banana. The leaves

infected by the disease are determined based on the color difference between the healthy and the infected leaves. The trained model gives motivating classification with better accuracy.

In [48] a model for identification and classification of enset bacterial wilt disease has proposed using a convolution neural network. In this work, a total of 4896 healthy and the infected images for experimenting is collected from the field. The eight layers based CNN model is used which contains 5 convolution layers, and 3 fully connected layers that are developed followed by the VGG16 model to identify and classify enset bacterial wilt disease from a healthy one. To get a similar size of all images and to decrease the computational time of training the size normalization preprocessing strategy is performed. In addition to this, data augmentation techniques are used to reduce overfitting problems. As researchers said that, the model gives the higher accuracy in both testing and training phases with classification accuracy 985%. comparing to some popular models such as inception v3 and VGG16.

2.3.9. Summary

As we can see from the above sections, plant disease detection and classification using image processing techniques with deep learning approaches like CNN is one of the active research areas which gives us promising results. In recent years, several plant disease detection researches have been conducted using machine learning and deep learning approaches like CNN models. Enset is one of the staple food plants in Ethiopia which is affected by a different disease. Among those affecting disease, the most critical disease of enset is Rootmealybug disease, Leafspot disease, and Bacterial Wilt disease. Even if several works have been developed for different plant disease identification and classification, there is only one work has proposed for the enset plant that only identifies the enset bacterial wilt disease. On the other hand, there is no machine learning and deep learning models have proposed to detect and classifies the three most critical diseases of enset. Hence, an efficient CNN-based model is required to detect and classify healthy enset plants from enset root mealybug disease, Leafspot disease, and bacterial wilt disease using contaminated and healthy enset plant images.

CHAPTER THREE

RESEARCH METHODOLOGIES

In this chapter, the experimental methodology is applied that is used to complete this thesis work. Therefore, to design and develop the proposed model this study has passed through the following activities including data collection and preparation, data preprocessing, data augmentation, dataset partitioning, tools used to implement and test the model, and the evaluation techniques to measure the predictive performance of the model.

3.1. Dataset collection

The image dataset samples used in this work are collected the different fields of research centers and University agriculture campuses namely Southern Agricultural Research Institute in Hawassa, Hawassa University Agricultural Campus in Hawassa, and a Worabe Agricultural Research Center in Worabe by capturing images of enset plants infected with diseases and healy in real life scenario and these images are captured using Huawei Smart Phone with 12MP multiresolution camera.While collecting images, we have tried to capture images in various capture conditions, different shapes, sizes and uncontrolled illumination. This has helped with training the model in a way that it can do well in real life scenario in any possible weather. Each of the captured images belongs to the RGB color space by default and were stored in the uncompressed PNG format. The total number of originally collected image datasets befor imade augumentation technique is 5,940. In other words, we have a total number of 4 classes.The three classes namely enset bacterial wilt diseases with number of 1,465 images, enset mealybug pest disease with number of 1,679 images, enset leafspot disease with number of 928 images,and one class for healthy enset plant with number of 1,778 images. Then after, all of the images are checked and annotated by the plant science experts of the Southern Agricultural Research Institute, Hawassa, SNNPR Ethiopia.

3.1.1. Data pre-processing

In this work, before images are feeding to the network two pre-processing steps are applied. First, the images are resized into 150 x 150 x 3 to match the size of the input layer of the CNN that takes color images. This size is decided after several experiments that give us optimum results. Secondly, the images are rescaled from values between 0 and 255 to values between 0 and 1, as neural networks prefer to deal with small input values as well as they will give an as

good result with the good color distribution of image pixel values since the collected images are captured with different sizes, orientations, poses, backgrounds, and illumination.

3.1.2. Data augmentation

Most of the previously studied CNN image classification models are used very large datasets that are collected from different places and some of the datasets are generated from that of the collected dataset using data augmentation technique using python[12,13,17,18,20,29,31,32]. An image data augmentation is a technique that can be used to artificially expand the size of a training dataset by generating more data from the existing training sample. Transformations like resizing, crop, rotation, horizontal flipping, and intensity transformations such as contrast and brightness enhancement, color, noise are popular techniques of augmentation.

Figure 15: Sample healthy enset plant augmented images

In this work, since the originally collected enset disease dataset is not sufficient, some of the datasets are augmented to create an additional version of images using augmentation methods to distinguish the different disease categories, an additional version of images is created by zooming and rotating the images into different degrees. The augmented dataset also helps to reduce overfitting during the training stage. The augmented sample images are shown in Figure 15, Figure 16, Figure 17, and Figure 18 below.

Figure 16: Sample augmented enset bacterial wilt disease images

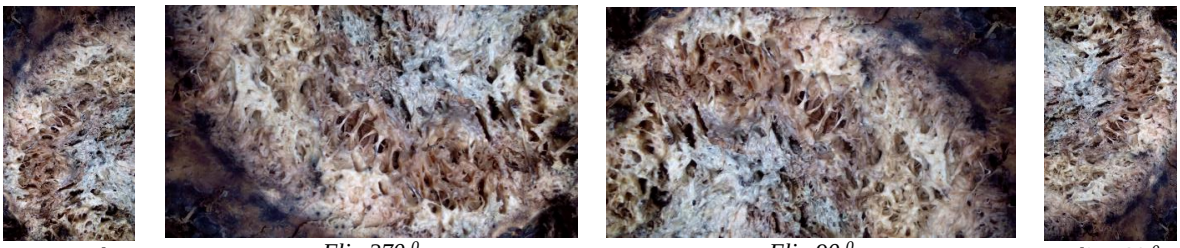


Figure 17: Sample augmented enset root mealybug disease images

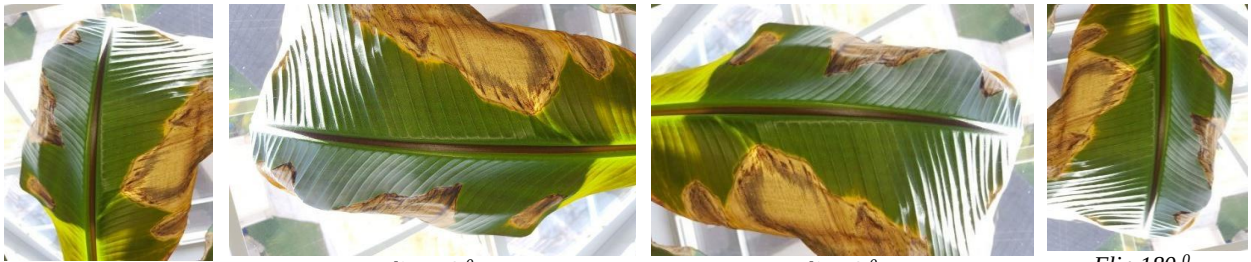


Figure 18: Sample augmented enset leafspot disease images

3.1.3. Dataset Partitioning

In this work, the total number of originally collected images is divided into two sets which are training dataset and test dataset. Then after, the training dataset is split into two splits the training split is used to train the model and the validation split is used to assess the performance of the model during the training and it is also used to fine-tune model parameters to select the best performing model. The test dataset is used to test the model after training of the model. Splitting the dataset into training and test sets have a great impact on model evaluation, for this reason, this thesis work has used a 10-fold cross-validation strategy to train-validate-test-split tasks. This technique is set apart some fraction of the dataset for the train, some fraction of the dataset for validation, and the remaining one is for testing. Table 1 shows us how the 10-fold cross-validation splits the enset training dataset into a training split and validation split.

Table 1:10 fold cross-validation strategy

Split 1	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Fold_6	Fold_7	Fold_8	Fold_9	Fold_10
Split 2	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Fold_6	Fold_7	Fold_8	Fold_9	Fold_10
Split 3	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Fold_6	Fold_7	Fold_8	Fold_9	Fold_10
Split 4	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Fold_6	Fold_7	Fold_8	Fold_9	Fold_10
Split 5	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Fold_6	Fold_7	Fold_8	Fold_9	Fold_10
Split 6	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Fold_6	Fold_7	Fold_8	Fold_9	Fold_10
Split 7	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Fold_6	Fold_7	Fold_8	Fold_9	Fold_10
Split 8	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Fold_6	Fold_7	Fold_8	Fold_9	Fold_10
Split 9	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Fold_6	Fold_7	Fold_8	Fold_9	Fold_10
Split 10	Fold_1	Fold_2	Fold_3	Fold_4	Fold_5	Fold_6	Fold_7	Fold_8	Fold_9	Fold_10

3.2. Tools used to implement and test the proposed model

In this work, the experiments are implemented on Windows 10, 64 bits operating system containing core i3 with 16GB RAM, and 3.6GHz Central Processing Unit (CPU) with 1TB hard disk. The model is implemented using open-source software such as the TensorFlow framework with Keras neural network library and python programming language. Table 2 shows us information about tools used to implement this thesis work.

Table 2: Detail information about tools used for this work

Name of the environment	Parameters
Operating system	Windows 10 enterprise 64 bit
CPU	Intel core(TM) i7-7700 CPU with 3.6GHz
Hard disk	2 TB
Environment	Anaconda navigator, TensorFlow, Keras
Language	Python
Notebook	Jupyter
RAM	8GB
Office 2016	Documentation

3.2. Evaluation

In this work, to evaluate the performance of the proposed model, the dataset is split into a different range of training and validation where the blue color is for validation and the remaining one is for training using a 10-fold cross-validation strategy. During the training process, we have used a combination of accuracy, precision, recall, F1-score, support, and confusion matrix to evaluate the performance Model. After that, a model is tested using a test dataset that is not used during training or validation phases.

CHAPTER FOUR

MODEL DESIGN AND EXPERIMENT

This chapter presents the design and experimental procedures of the proposed model that is designed based on the most studied CNN plant disease detection models. The general design architecture of the model is divided into several necessary phases as shown in Figure 19 including the input phase, data preprocessing phase, data augmentation phase, dataset partitioning phase, feature extraction phase, classification phase, the output phase, the evaluation phase, and the predictive phase. These phases are discussed briefly in the following subsections.

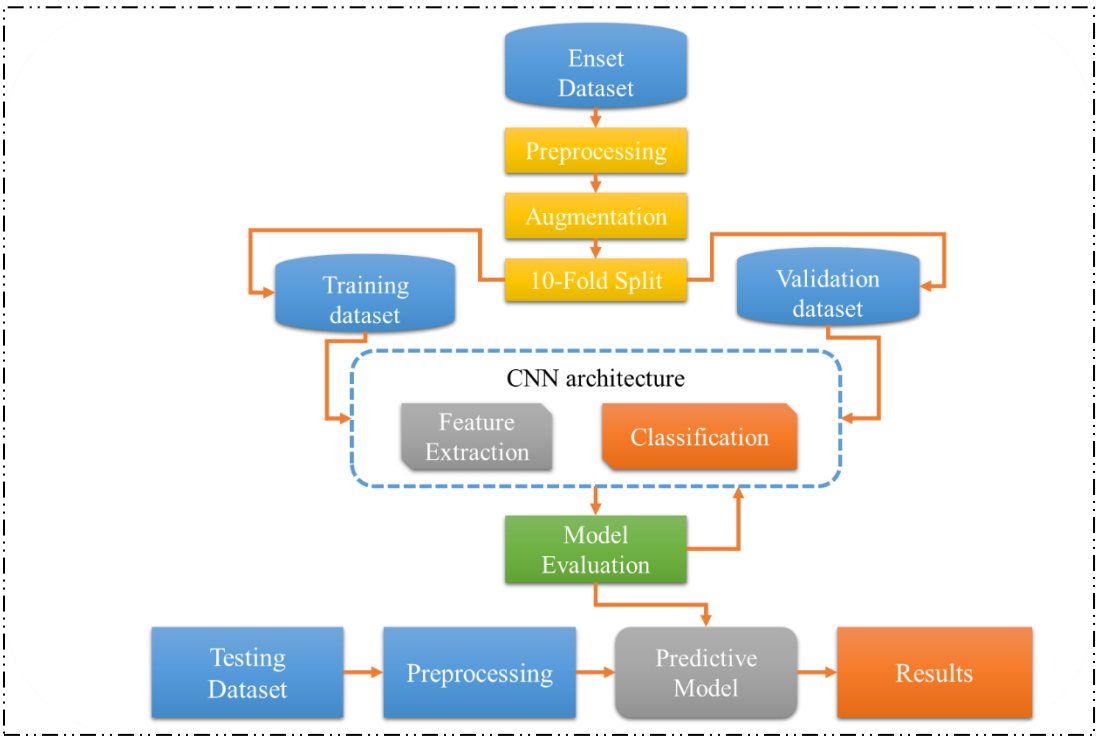


Figure 19. The general architecture of the proposed model

4.1. Experimental Setup

The experiential procedure of the proposed model is based on the following steps: First, the experimental execution environments (tools) are prepared followed by previously studied models. Second, the whole prepared enset dataset is divided into a training dataset and test dataset. Third, the whole dataset is preprocessed means the dataset is resized to 150x150x3 and normalized to *1/255 to convert decimal values of images between 0 and 1.Fourth, to increase

the size of the dataset, the data augmentation techniques are applied by rotating the images into different degrees, horizontal flipping, vertical and horizontal shifting. Fifth, the training dataset is split into 10 training sets and 10 validation set using a 10 fold cross-validation strategy and it passes to the feature extraction part of CNN. Sixth, the training of the proposed model to extract the most important feature that is the process of running a training dataset through the model from the input layer to the output layer simultaneously to make a classification and figuring out the results or errors. Seventh, we have used a combination of accuracy, precision, recall, F1-score, support, and confusion matrix to evaluate the performance model. Finally, we have checked the predictive performance of our model using a single image from each class.

4.1.1. Hyper Parameters configuration

The hyperparameters that are used for the proposed model are defined in the following Table 3 including activation functions, learning rate, epochs, optimization algorithm, batch size, and loss function. The activation functions that are used to conduct experiments are ReLU, SoftMax, and Sigmoid which are applied in both convolution layers and output layers. The number of epochs that are used in the proposed model was an epoch of 30 to get the optimum results. The batch size that is used for the proposed model is a batch size of 90 to reduce the computational time of the machine. The optimization algorithm that is used for the proposed model is an Adaptive Moment Estimation (Adam) optimization algorithm. The learning rate that we have used for the proposed model is 0.001. The Loss function that is used for the proposed model is Categorical Cross-Entropy (CCE) since the type of problem that going to solve is not binary classification.

Table 3:Hyperparameters configuration

Parameters	Configuration values
Learning rate	0.001
Batch size	90
Epochs	30
Optimization algorithm	Adam
Activation function	ReLU and SoftMax
Loss Function	Categorical Cross-Entropy

4.1.1. Input Layer Setup

During the implementation process of the model, before starting the convolution operation, the input layer is first configured properly to accept pixel of image dataset in the form of arrays, since the input images are the color images, the shape of the input layer is configured as 150 ×

150 × (N = 3) where N defined as a total number of channels. This layer only passes the input to the first convolution layer without any computation. Hence, there are no learnable features in this layer and the number of parameters in this layer is 0.

4.1.2. Feature Extraction Layers Setup

The proposed model involves four convolution layers with four filters(16,32,64,32 respectively), four max-pooling operations with rectified linear unit function (ReLU) to add nonlinearity during the training of the network having padding of 0 stride of 1, two fully connected layers, and a final layer as the classification layer that has 4 output nodes and two dropouts are included after the fourth max-pooling layers to prevent the problem of overfitting. The general features extraction and classification phases of the proposed model are shown in Figure 20 and Figure 21.

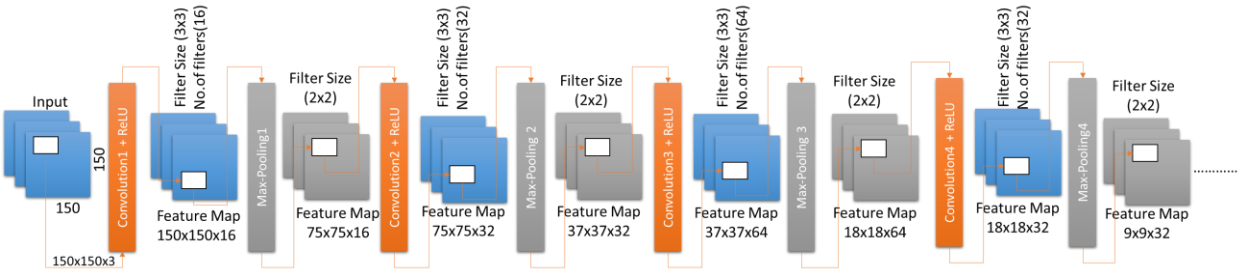


Figure 20. General Features Extraction Phase of the proposed model

The model extracts the most important features from both healthy and diseased enset images before the classification stage is started. During convolution operation, an image pixels are pushed to the convolution layer, this is known as the values of the input image. Then, the values of the input image are sliding an infinite number of steps this is known as the convolutional part of the network, and values are multiplied and summed up with the weights of the filter and output values are called feature map. This feature map is the most important feature of the input images that contain patterns that are used to classify the input enset images to different diseases and healthy classes. The output or the feature map of the proposed model is computed as follows in equation 1[48].

$$Mi = \sum_{k} wik * xk + b \qquad (1)$$

Where: w_{ik} is a filter of the input, x_k is the k[th] number of channel, and b is the bias term.

Then, the values of the feature map are passed to the ReLU activation function to introduce nonlinearity to the network. Afterward, it is passed into the pooling layer to reduce the dimension of the feature map that also reduces the computational time of the model. In addition to this, the three-dimensional size of the output volume is computed as equation 2 in each layer [48].

$$output\ size(w2) = \frac{(w1 - F + 2P)}{S} + 1 \qquad (2)$$

Where: w1 is the size of the input volume, F is filter size, P is the paddings, and S is the stride.

4.1.2.1. Convolution layer Setup

The proposed model is a sequential model that has a series of layers to convert the standard size of images into a feature set for further processing. There are four convolutional layers, each four convolution layers are connected to the four max-pooling layers to reduce the dimension and speed up calculations of the features in the enset dataset. The first convolutional layer of the model takes a size of 150×150×3 input image using 16 filters with a size of 3×3×16, a stride of 1 pixel, and a padding value of 0. This process is followed by a rectified linear unit (ReLU) activation function to replace the entire negative pixel value to 0 to introduce nonlinearity to the networks. The second convolutional layer takes the output of the first convolutional layer as input and filters it by using 32 kernels with a size of 3×3×32. The third convolutional layer takes the output of the second convolutional layer as input and filters it by using 64 kernels with a size of 3×3×64. The fourth convolutional layer takes the output of the third convolutional layer as input and filters it by using 32 kernels with a size of 3×3×32. All the convolutional layers of the proposed model use ReLU nonlinearity as activation functions. ReLU is chosen because it is faster than other non-linearities such as tanh to train deep CNNs with gradient descent [9,10,48].

4.1.2.2. Pooling layer Setup

There are four max-pooling layers with a kernel size of 2 × 2 × 3. All four max-pooling layers reduce the output of all four convolutional layers with a filter size of 2×2 with a stride value of

1.All pooling layers have no learnable features and they only perform down-sampling operation along the spatial dimension of the input volume, hence the number of parameters in these layers is 0.

Table 4. Configuration of the Proposed feature extraction phase

Layers	Filter size	Depth	Stride/ Padding	Output Shape	# Param
Input	-	-	…	(150 ×150 × 3)	0
Conv1 + ReLU	3 × 3	16	1/same	(150,150,16)	448
maxPool1	2 × 2	-	…	(75,75,16)	0
conv2 + ReLU	3 × 3	32	1/same	(37,37,32)	4640
maxPool2	2 × 2	-	…	(37,37,32)	0
Conv3 + ReLU	3 × 3	64	1/same	(37,37,64)	18,496
maxPool3	2 × 2	-	…	(18,18,64)	0
Conv4 + ReLU	3 × 3	32	1/same	(18,18,32)	18464
maxPool4	2 × 2	-	…	(9,9,32)	0

4.1.2.3. Flattening Layer Setup

The flattening is the process of converting all the results of a 2-dimensional array from pooled feature map into a single long continuous linear vector to create fully connected layers. In other words, it is the process of putting all the pixel data in one line and makes connections with the final layer.

4.2. Classification Layers Setup

In the proposed model, there are a total of three fully connected layers including the output layer. The main function of these layers is to classify the input images based on the most important extracted features which are done by convolution layers. The first fully connected layer accepts the last output of the convolution layer four and pooling layer four in the form of a flattened layer before feature maps fed into the second fully connected layer. Figure 21 shows the configuration of the proposed classification phase.

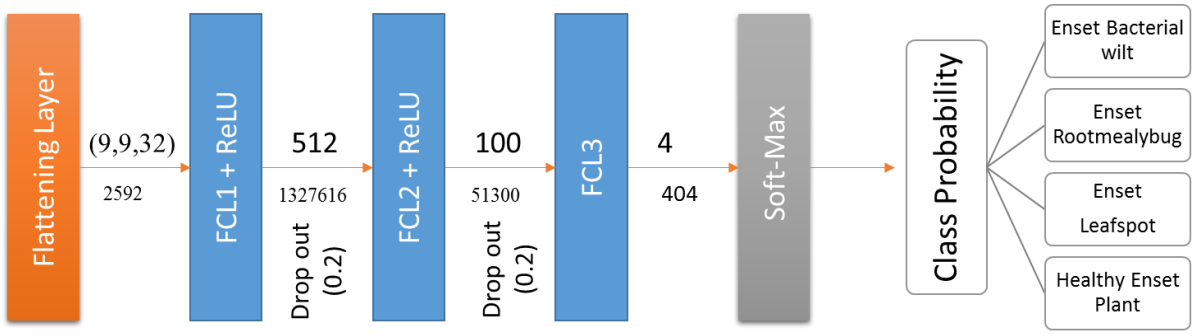


Figure 21: Configuration of the proposed Classification phase

4.2.1.1. Fully Connected Layer Setup

In the proposed model there are a total of three fully connected layers including the output layer. The importance of these layers is to classify and predict class probabilities of the input extracted features by the convolution layers. The first fully connected layer accepts the output of the fourth max-pooling layer in the form of a flattening layer. It contains a total of 512 neurons and is followed by a ReLU activation function and a dropout operation with a value of 0.2. The second fully connected layer contains 100 neurons that are connected to all the neurons of the first fully connected layer as well as the last fully connected layer and are followed by ReLU and dropout operations as above. In each layer, the dot product is applied to multiply the values most important features by weights and bias are added then after the results are passes forward to the third fully connected layer in which every neuron represents a classification label to produce a single value. The output of each fully connected layer is computed as follows in equation 3 [48].

$$Class\ output = w * x + b \quad (3)$$

Where w is weighted and x is the values of the features extracted by the convolutional layer and b is bias. For example, the first fully connected layer performs a dot product to compute the class of output. It takes the output of the flattening layer (2592) as input and multiplied by weights (512) and the bias 512 is added.

class output =2592 * 512 + 512

=1327616

The third fully connected layer is the last layer of the proposed model which contains four neurons with a Softmax activation function that represents the number classes in the proposed model. The output of this layer is then transferred to the SoftMax function to determine the classification of the input healthy and diseased image. The SoftMax activation function is implemented, to sum up, the output values equal to 1.0 and limits individual outputs to values between 0 and 1.The configuration of the proposed classification phase shown in Table 5.

Table 5. Configuration of the proposed classification phase

Layers	Output Shape	# Param
Flatten	2592	0
FC1 + ReLU	512	1327616
Dropout	512	0
FC2 + ReLU	100	51300
Dropout	100	0
FC3+ SoftMax	4	404

4.3. Training of Proposed Model

Training of the model is the process of extracting the most important features or representative pixels from healthy and contaminated training enset image datasets and teaching this pixel to the networks directly without handcraft, then after classifying these types of disease based on learning categories of the specific class. After several sets of training processes using different hyperparameters and parameters the proposed model has completed the detection and classification of enset plant disease. During the training process, the performance of the model is measured using the validation dataset. After measuring the performance of the trained model, the model with higher validation accuracy is saved at each split (from split 1 to split 10) and used as a proposed classification model. Then the testing phase is performed by giving an unseen enset image dataset to the proposed trained model. The model finally predicts the class of enset dataset in the form of probabilistic value which is a value between 0 and 1 and the name of the disease with its image that belongs to one of the given classes during the training.

4.4. Experiments with other pre-trained Models

The pre-trained models are kind of CNN models that are trained with a million previously developed by someone else to solve a similar problem. The basic premise of these models is simple: just take a model trained on a large dataset and transfer its knowledge to a smaller dataset. Their ability to generalize an image object is higher since these models are trained by millions of image features and thousands of classes. Therefore, these extracted features and classes learned by pre-trained models are used to solve many other real-world problems even though the problems are different than those of the original task [17,19,29,31,34]. The technique of using these models to solve real-world problems is called transfer learning. Training these pre-trained models from scratch is computationally expensive and sometimes impossible. Most of the pre-trained models have two important parts, the first main part is the convolution part (feature extraction part) which includes convolution layer and pooling layers to extract the

important features from the given input image and the second part is a classification part which is used to classify the input image based on the extracted features of the input images. For this work, two pre-trained models are used namely InceptionV3 and MobileNet by downloading the weights of them and training using the enset dataset, and finally, the results are compared with the proposed model. During the training of these models, the convolutional bases are taken as it is or frozen and only the classification base is trained. In the transfer learning process, three common strategies[19] are there, the first strategy is to train all the convolution base and only change the fully connected layers based on specified class, the second strategy is to train some part of the convolution base by freezing the weights of the pre-trained model and change the fully connected layer. The third strategy is to freeze all convolution base and train classification base. The process of training some parts of the convolution base is called fine-tuning. Figure 22 shows us the three strategies of transfer learning.

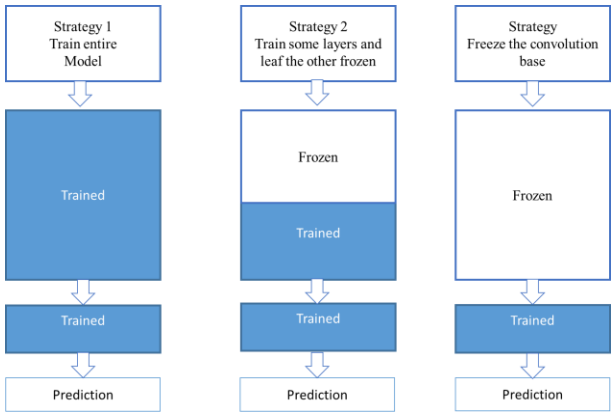


Figure 22.Sample image for Proposed Pre-trained Models classification [1]

4.5. Summary

The proposed model is designed and implemented based on mostly studied CNN plant disease detection models. In this work, the total number of 14,992 datasets are used for experimenting. The proposed enset dataset contains four different classes where three classes are represented infected enset plant diseases and the remaining class represents a healthy enset plant image dataset. since the originally collected enset disease dataset is not sufficient, the dataset is expanded to create an additional version of images using different augmentation methods to distinguish the different disease categories and to overcome overfitting. The proposed model is

[1] https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751

a sequential model having a series of layers to convert the standard size image into a feature set for further processing. The first layer of this model is the convolutional layer with 16 filters with a size of (3,3) and ReLU is used as the activation function. The second layer of this model is the max-pooling layer that will decrease the size of the convoluted image by (2, 2). The third layer is the convolutional layer with 32 filters with a size of (3,3) and ReLU as the activation function having the same value of padding. The fourth layer is the max-pooling layer that will decrease the size of the convoluted image by (2, 2). The fifth layer of this model is the convolutional layer with 64 filters with a size of (3,3) and ReLU as the activation function having the same value of padding. The sixth layer is the max-pooling layer with a pool size of (2, 2). The seventh layer of this model is the convolutional layer with 32 filters with a size of (3,3) and ReLU as the activation function. The eighth layer is the max-pooling layer with a pool size of (2, 2) followed by a dropout rate of 0.2. Then one more layer is added to flatten the output of the above designed convolutional neural network model. The model has two fully connected layers that will be used for the classification of images based on the generated feature set the first one has 512 hidden neurons and the activation function is ReLU followed by dropout value of 0.2 and the second one has 100 hidden neurons followed by dropout value of 0.2 and the activation function ReLU is used. It has one more fully connected layer which acts as the output layer of the network having 4 output nodes having SoftMax activation function.

CHAPTER FIVE

RESULTS AND DISCUSSIONS

This chapter focuses on the experimental evaluation of the proposed model. The evaluation is conducted towards the performance of the proposed model both independently and in comparison with the performance from previously developed classical pretrained models. The experimental results of the proposed model are discussed based on the following tasks: analyzing results of both proposed and pre-trained models, identifying the class of each enset disease, and measuring the accuracy of both the training process and the testing process of the proposed and pre-trained models. After that, a comparing is made with the pre-trained existing models. All experimental details and results are presented in detail using different graphs and tables.

5.1. Experimental results

In this section different experiments are conducted to investigate the classification performance of the proposed model. The first four experiments are performed on the proposed model and the last two experiments are conducted on pre-trained CNN models. The experimental process of this work has two main phases like other CNN classification tasks. The first part is the training phase and the second part is the testing phase. The training phase learns different features of an image that is extracted by the convolutional base afterward it sends its output to the classifier to obtain the desired class. The testing phase is used to analyze the trained model using a test or unseen dataset to test the performance of the classification model. The experimental results are discussed in detail in the following sections. Moreover, during the training of the proposed model in each fold(split), the model is tested against a validation set, confusion matrics, and different classification accuracy metrics such as precision, recall, F1-score, and support, and results are displayed in the form of tables and figures.

5.1.1. Experimental Results of the proposed Model

The proposed model is designed based on mostly studied plant disease detection CNN models which can run in a minimum hardware resource and can give promising results. This model has a total of seven layers, four convolution layers, and three fully-connected layers. This model accepts 150 × 150 × 3 sizes of color images. To train and get a good result in this model a set

of experiments are conducted using a 10-fold cross-validation strategy and train/test split strategies using different learning rates, different activation functions, different batch sizes, and different epochs. The model is trained with a total number of 14,992 images including augmented images. The following sections show us a set of experiments done in the proposed model to get good results.

5.1.1.1. Experiment 1: Training using 10-fold Cross-validation strategy

In this section, ten experiments are conducted for each fold(split) using a 10-fold Cross-validation strategy. This strategy is set apart some fraction of the dataset for the training, and the remaining fraction of the dataset for validation using python scripts. Table 26 shows the summary results of the proposed model using 10-fold Cross-validation with the different train and validate split choices while the number of folds is varied. During this experiment, the total training dataset is divided into ten(10) different folds. Then after 10% of the dataset is used for validation and the remaining 90% of the dataset is used for training as discussed in Table 1. As we can see from this table all folds(in each split) with blue colors (from fold_1 to fold_10 diagonals) are used for validation set which represents 10% of whole dataset and the remaining folds (in each split) with other color are used for training set which represents 90% of the whole dataset. In addition to this, for each split (from split 1 to split 10 with the red color) we have trained with the epoch of 30 (thirty). So that we have trained our proposed model with a total of 10 x 30 (300) epochs. Moreover, in each split, we have created 1 model with the best validation accuracy to select one best model among 10 models rather than creating 1 model. So that we have created a total of 10 models (from split 1 to split 10). In other words, every 10 models are created after training the model with epoch values of 30. In the following sections, all 10 split experimental results are presented in detail.

Split 1: The experimental results obtained from split 1(the blue,fold_1 as the validation set and the remaining one as the training set) is presented in Table 6 concerning Training, Validation, and classification accuracy additionally Training, Validation, and classification loss in the form of a percentage.

Table 6. Training, Validation, and Testing accuracy and loss of the Fold one

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.93%	99.03%	99.07%	0.31%	4.54%	0.93%

The graphs in Figure 23 show us the results of the proposed model for split one (fold_1 as the validation set and the remaining one is as training set) concerning training versus validation accuracy and training versus validation loss.

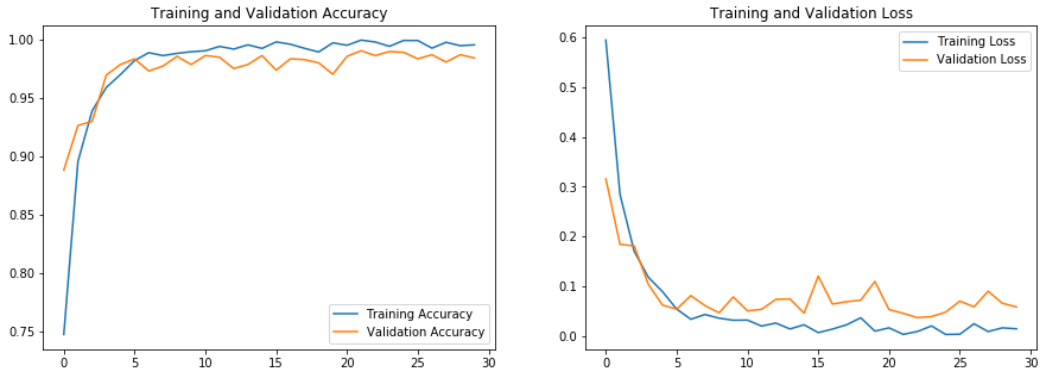
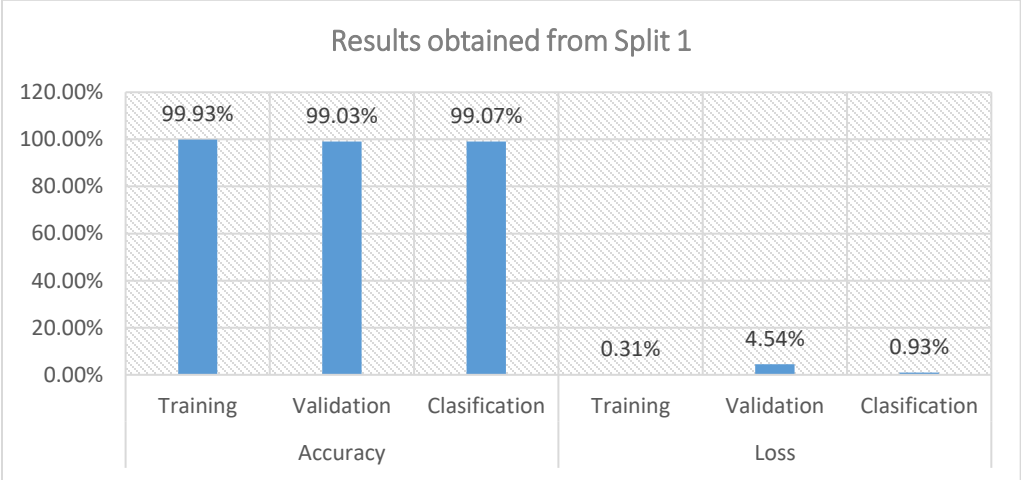


Figure 23. Training vs validation accuracy and Training vs validation loss for split one



The detailed experimental result of the proposed model is illustrated in the form of the confusion matrix, precision, recall, f1-scores, support, and confusion matrix for split one for each class is illustrated in Table 7 and Figure 24.

Table 7: Precision, Recall, F1-Score, and Support test data for split one

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.98	0.99	0.98	387
Health Enset	0.99	0.97	0.98	435
Leaf Spot	0.98	1.00	0.99	279
Rootmealybug	0.99	0.99	0.99	398

Figure 24 illustrates the classification accuracy as well as classification loss for split one of the proposed Model. As we can see from the plotted table graph in the first class from a total of 387 images 384 images are accurately classified as bacterial wilt,3 images are wrongly classified as healthy enset.

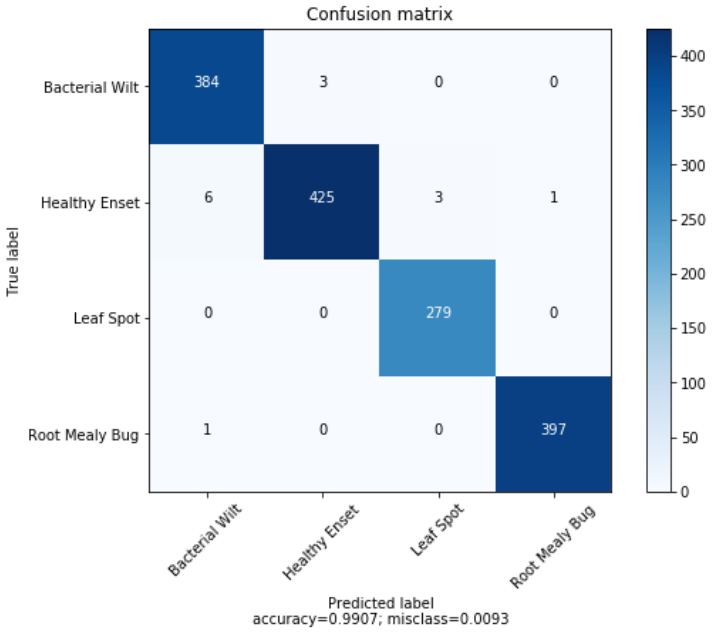


Figure 24.Confusion matrix on the Testing dataset for split one

In the second class from a total of 435 images, 425 images are accurately classified as healthy enset, 6 images are classified wrongly as bacterial wilt,3 images are wrongly classified as leafspot, and 1 image is classified wrongly as root mealybug. In the third class, all of the 279 images are accurately classified as leafspot. In the fourth class, from a total of 398 images,397 images are accurately classified as root mealybug, and the remaining 1 image is wrongly classified as bacterial wilt.

Split 2: The overall experimental results obtained from the proposed model for split two(the blue,fold_2 as the validation set and the remaining one is as training set) are presented in Table 8 concerning Training, Validation, and Test accuracy as well as Training, Validation, and Test loss in the form of a percentage.

Table 8. Training, Validation, and Testing accuracy and loss of the split two

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.96%	99.37%	99.33%	0.13%	2.65%	0.67%

The graphs in Figure 25 show us the results of the proposed model for split two during training concerning training versus validation accuracy and training versus validation loss.

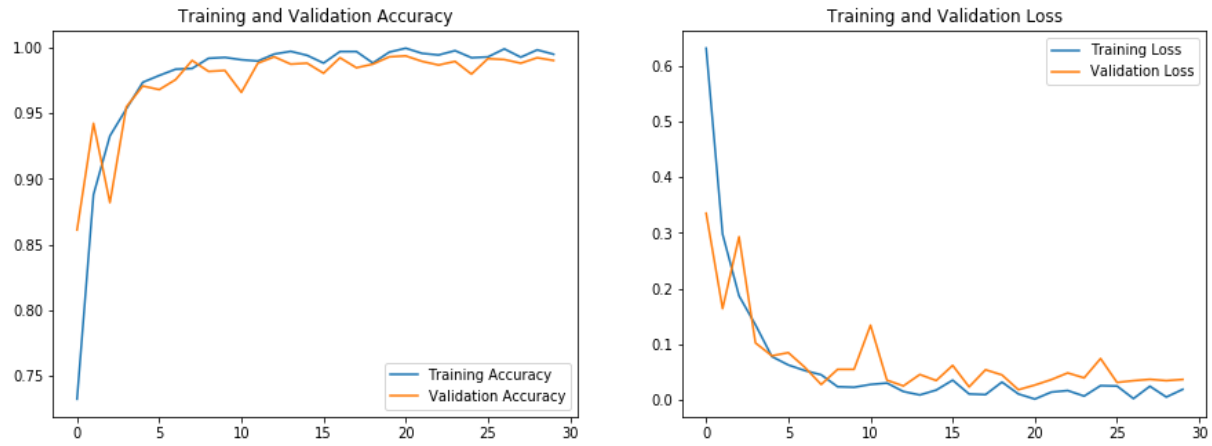
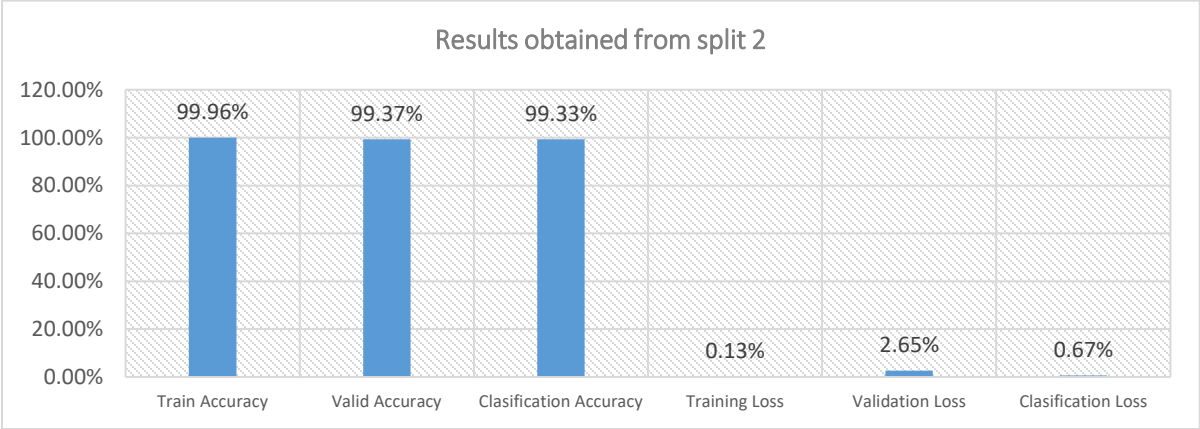


Figure 25.Training vs validation accuracy and Training vs validation loss of the split two



The detailed experimental result of the proposed model is presented in the form of the confusion matrix, precision, recall, f1-scores, support, and confusion matrix for split two for each class that is illustrated in Table 9 and Figure 26.

Table 9.Precision, Recall, F1-Score, and Support test data of the split two

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.99	0.99	0.99	387
Health Enset	0.99	0.98	0.98	435
Leaf Spot	0.98	1.00	0.99	279
Rootmealybug	0.99	1.00	0.99	398

Figure 26 illustrates that the classification accuracy as well as classification loss for split two of the proposed Model. As we can see from the plotted table graph in the first class from a total of 387 images 384 images are accurately classified as bacterial wilt,3 images are wrongly classified as healthy enset.

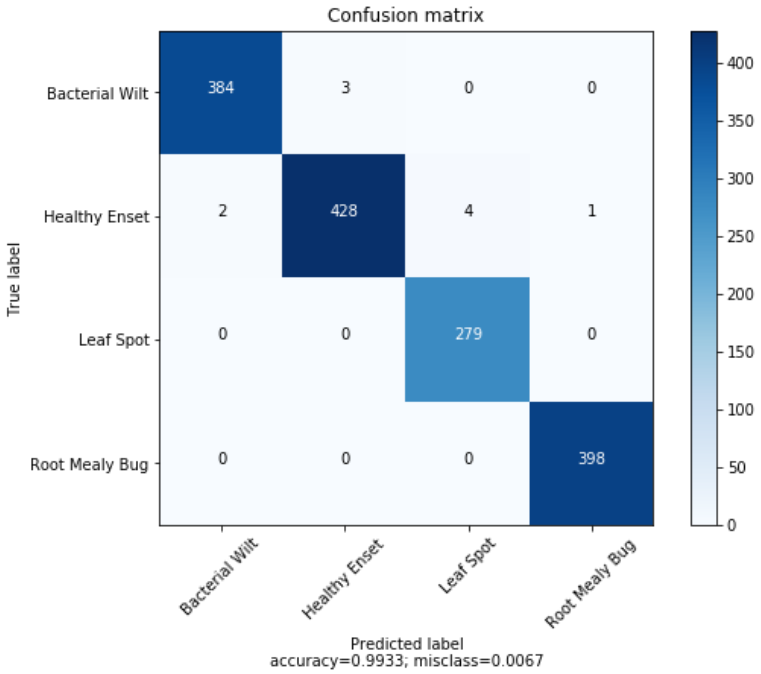


Figure 26. Confusion matrix on the Testing dataset for split two

In the second class from a total of 435 images, 428 images are accurately classified as healthy enset, 2 images are classified wrongly as bacterial wilt,4 images are wrongly classified as leafspot, and 1 image is wrongly classified as root mealybug. In the third class, all of the 279

images are accurately classified as leafspot. In the fourth class also all of the 398 images are accurately classified as root mealybug.

Split 3: The overall experimental results obtained from the proposed model for split three (the blue, fold_3 as the validation set and the remaining one is as training set) are presented in the following Table 10 concerning Training, Validation, and Test accuracy as well as Training, Validation, and Test loss in the form of a percentage.

Table 10. Training, Validation, and Testing accuracy and loss of the split three

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.48%	99.10%	99.07%	1.84%	3.29%	0.93%

The graphs in Figure 27 show us the results of the proposed model for split three during training concerning training versus validation accuracy and training versus validation loss.

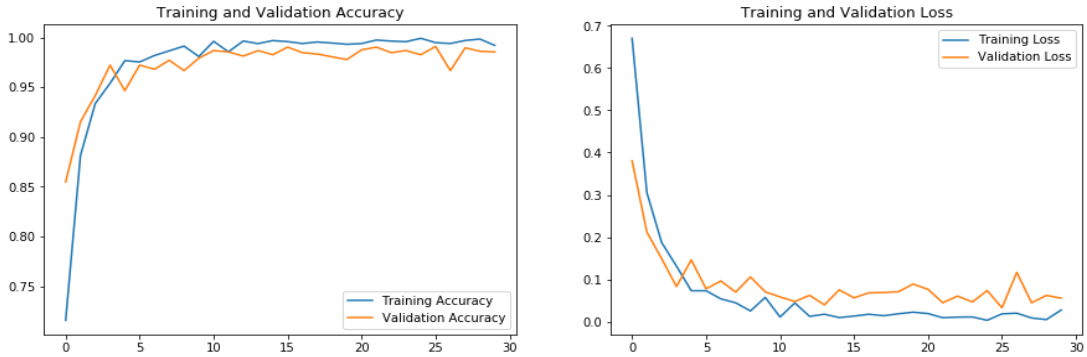
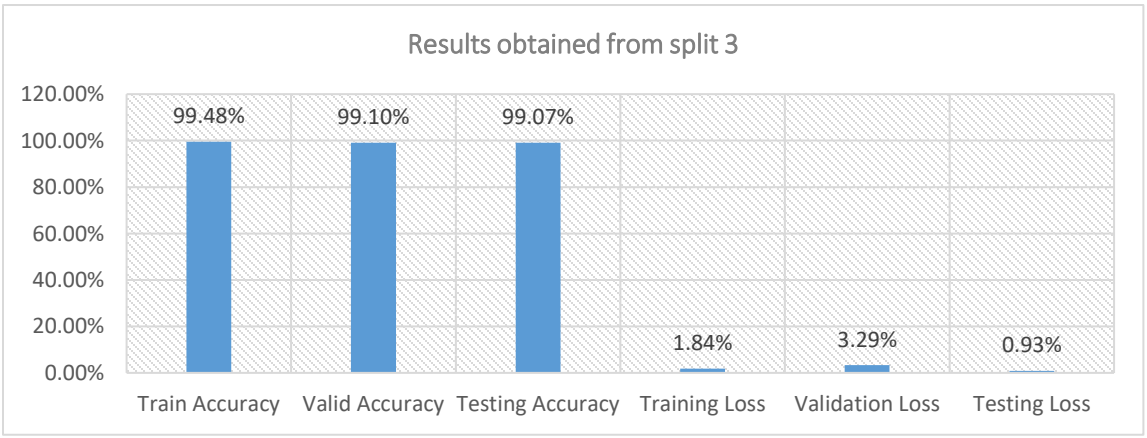


Figure 27. Training vs validation accuracy and Training vs validation loss of the split three



The detailed experimental result of the proposed model is presented in the form of the confusion matrix, precision, recall, f1-scores, support, and confusion matrix for split three for each class which is illustrated in Table 11 and Figure 28.

Table 11. Precision, Recall, F1-Score, and Support test data of the split three

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.98	0.98	0.98	387
Health Enset	0.98	0.98	0.98	435
Leaf Spot	0.99	1.00	0.99	281
Rootmealybug	0.99	1.00	0.99	398

Figure 28 illustrates that the classification accuracy as well as classification loss for splitting thee of the proposed Model. As we can see from the plotted table graph in the first class from a total of 387 images 380 images are accurately classified as bacterial wilt,5 images are wrongly classified as healthy enset,1 image is wrongly classified as leaf spot, and 1 image also wrongly classified as root mealybug.

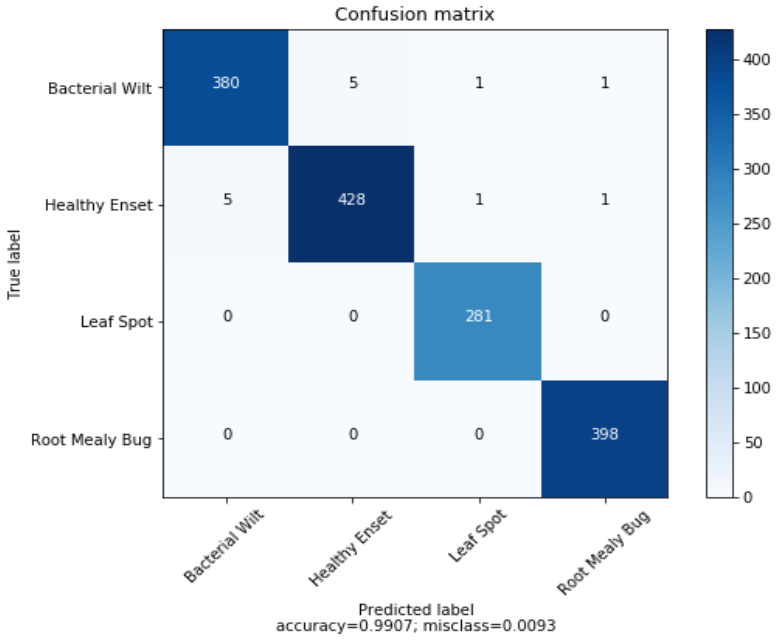


Figure 28. Confusion matrix on the Testing dataset for split three

In the second class from a total of 435 images, 428 images are accurately classified as healthy enset, 5 images are classified wrongly as bacterial wilt,1 image is wrongly classified as leafspot, and 1 image is wrongly classified as root mealybug. In the third class, all of the 281 images are

accurately classified as leafspot. In the fourth class also all of the 398 images are accurately classified as root mealybug.

Split 4: The overall experimental results obtained from the proposed model for split four (the blue, fold_4 as the validation set and the remaining one is as training set)are presented in the following Table 12 concerning Training, Validation, and Test accuracy as well as Training, Validation, and Test loss in the form of a percentage.

Table 12. Training, Validation, and Testing accuracy and loss of the split four

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.58%	98.75%	98.73%	1.15%	5.21%	1.27%

The graphs in Figure 29 show us the results of the proposed model for split four during training concerning training versus validation accuracy and training versus validation loss.

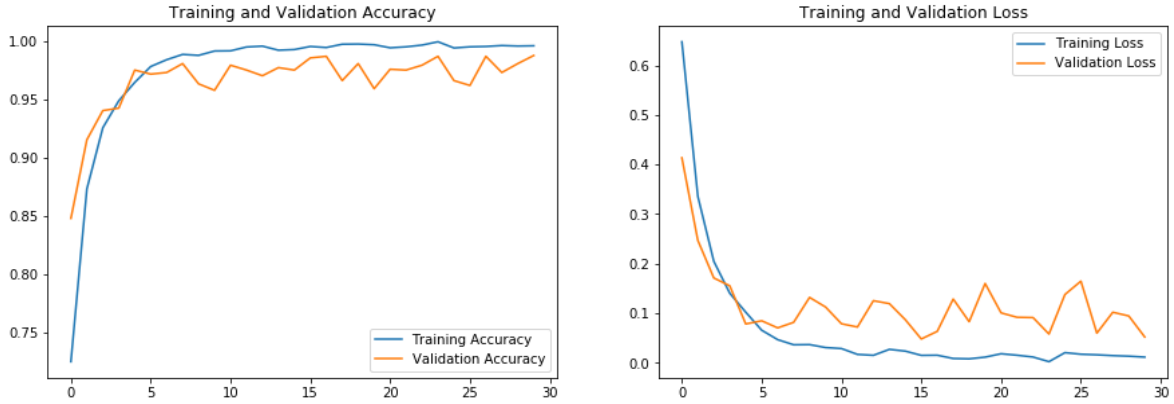
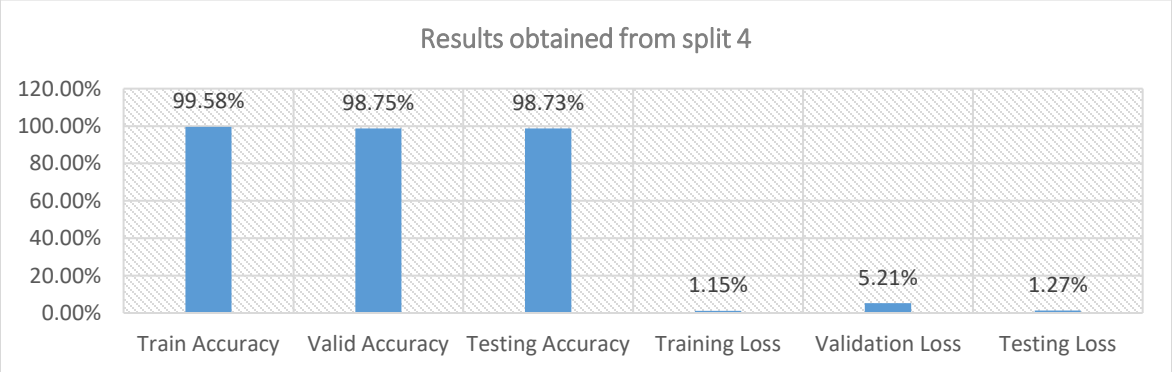


Figure 29.Training vs validation accuracy and Training vs validation loss of the split four



The detailed experimental result of the proposed model is presented in the form of the confusion matrix, precision, recall, f1-scores, support, and confusion matrix for split four for each class which is illustrated in Table 13 and Figure 30.

Table 13.Precision, Recall, F1-Score, and Support Fold four for test data

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.98	0.97	0.98	387
Health Enset	0.98	0.98	0.98	435
Leaf Spot	0.99	1.00	0.99	281
Rootmealybug	0.98	0.98	0.99	398

Figure 30 illustrates the classification accuracy as well as classification loss for split four of the proposed Model. As we can see from the plotted table graph in the first class from a total of 387 images 378 images are accurately classified as bacterial wilt,7 images are wrongly classified as healthy enset,1 image is wrongly classified as leaf spot, and 1 image also wrongly classified as root mealybug.

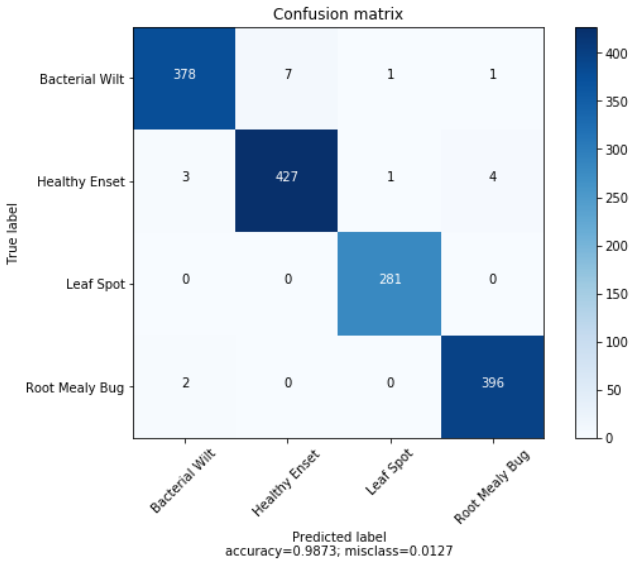


Figure 30. Confusion matrix on the Testing dataset for split four

In the second class from a total of 435 images, 427 images are accurately classified as healthy enset, 3 images are classified wrongly as bacterial wilt,1 image is wrongly classified as leafspot, and 4 images are wrongly classified as root mealybug. In the third class, all of the 281 images are accurately classified as leafspot. In the fourth class from a total of 398 images 396 images are accurately classified as root mealybug,2 images are wrongly classified as bacterial wilt.

Split 5: The overall experimental results obtained from the proposed model for split five(the blue, fold_5 as the validation set and the remaining one is as training set) are presented in the following Table 14 concerning Training, Validation, and Test accuracy as well as Training, Validation, and Test loss in the form of a percentage.

Table 14. Training, Validation, and Testing accuracy and loss of the split five

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.62%	98.96%	98.93%	1.40%	4.52%	1.07%

The graphs in Figure 31 show us the results of the proposed model for split five during training concerning training versus validation accuracy and training versus validation loss.

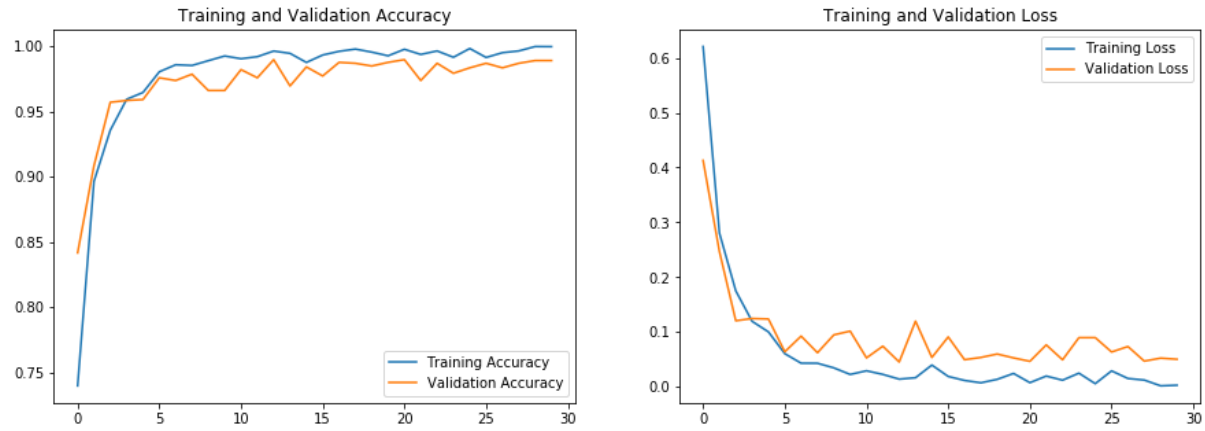
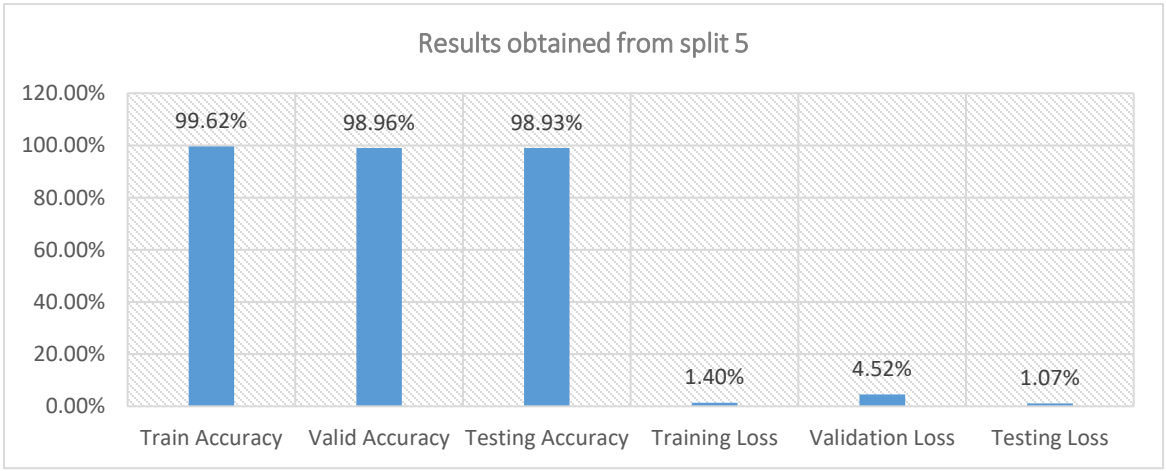


Figure 31. Training vs validation accuracy and Training vs validation loss of split five



The detailed experimental result of the proposed model is presented in the form of the confusion matrix, precision, recall, f1-scores, support, and confusion matrix for split five for each class which is illustrated in Table 15 and Figure 32.

Table 15. Precision, Recall, F1-Score, and Support split five for test data

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.98	0.98	0.98	387
Health Enset	0.98	0.98	0.98	435
Leaf Spot	0.98	1.00	0.99	278
Rootmealybug	0.99	0.99	0.99	398

Figure 32 illustrates that the classification accuracy as well as classification loss for split five of the proposed Model. As we can see from the plotted table graph in the first class from a total of 387 images 381 images are accurately classified as bacterial wilt,5 images are wrongly classified as healthy enset,1 image is wrongly classified as root mealybug.

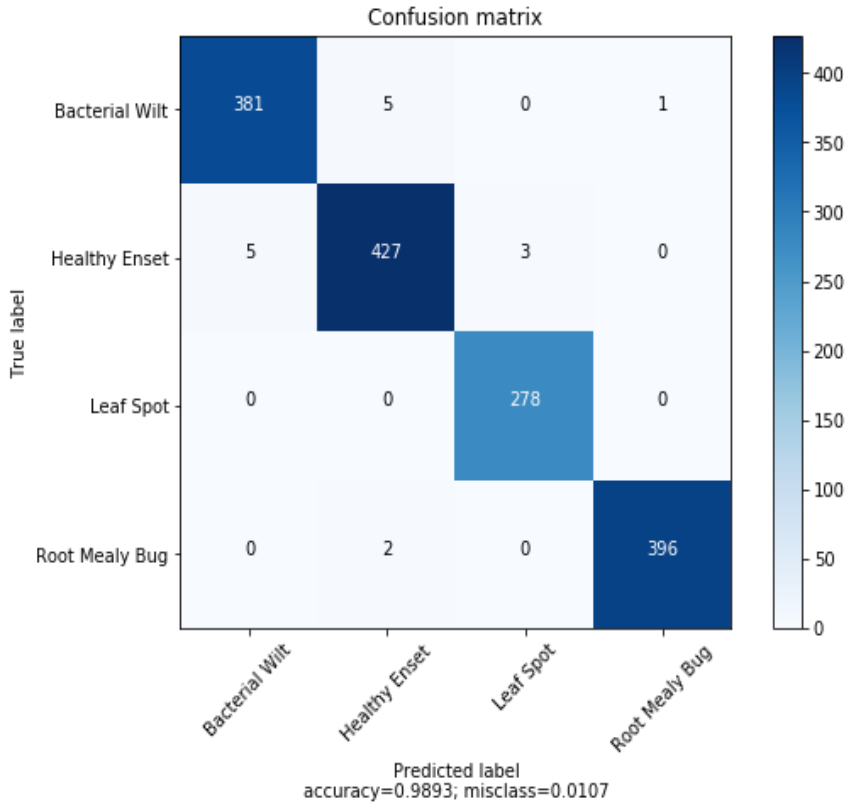


Figure 32. Confusion matrix on the Testing dataset for split five

In the second class from a total of 435 images, 427 images are accurately classified as healthy enset, 5 images are classified wrongly as bacterial wilt,3 images are wrongly classified as

leafspot. In the third class, all of the 281 images are accurately classified as leafspot. In the fourth class from a total of 398 images 396 images are accurately classified as root mealybug,2 images are wrongly classified as healthy enset.

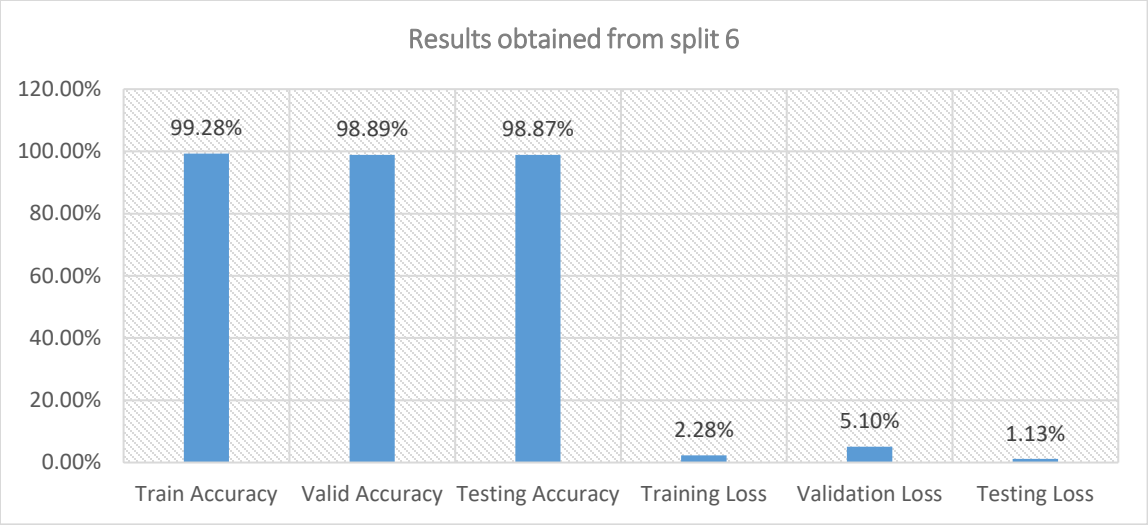
Split 6: The overall experimental results obtained from the proposed model for split six (the blue, fold_6 as the validation set and the remaining one is as training set) is presented in the following Table 16 concerning Training, Validation, and Test accuracy as well as Training, Validation, and Test loss in the form of a percentage.

Table 16. Training, Validation, and Testing accuracy and loss of the split six

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.28%	98.89%	98.87%	2.28%	5.10%	1.13%

The graphs in Figure 33 show us the results of the proposed model for split six during training concerning training versus validation accuracy and training versus validation loss.

Figure 33. Training vs validation accuracy and Training vs validation loss of split six



The detailed experimental results of the proposed model are illustrated in the form of the confusion matrix, precision, recall, f1-scores, support, and confusion matrix for Fold six for each class which is illustrated in Table 15 and Figure 34.

Table 17. Precision, Recall, F1-Score, and Support split six for test data

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.98	0.98	0.98	387
Health Enset	0.98	0.97	0.98	435
Leaf Spot	0.99	1.00	0.99	280
Rootmealybug	0.98	0.99	0.99	398

Figure 34 illustrates the classification accuracy as well as classification loss for split six of the proposed Model. As we can see from the plotted table graph in the first class from a total of 387 images 382 images are accurately classified as bacterial wilt,5 images are wrongly classified as healthy enset. In the second class from a total of 435 images, 424 images are accurately classified as healthy enset, 4 images are classified wrongly as bacterial wilt,2 images are wrongly classified as leafspot, and also 5 images are wrongly classified root mealybug. In the third class, all of the 280 images are accurately classified as leafspot. In the fourth class from a total of 398 images 397 images are accurately classified as root mealybug,1 image is wrongly classified as bacterial wilt.

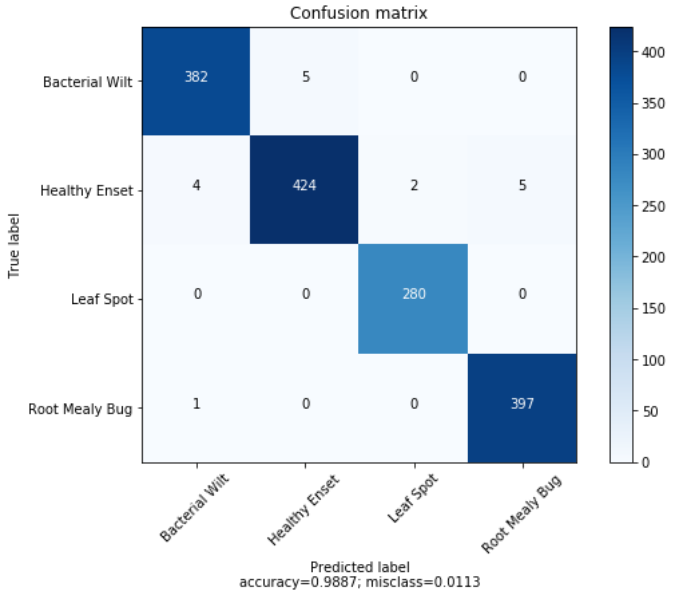


Figure 34. Confusion matrix on the Testing dataset for split six

Split 7: The overall experimental results obtained from the proposed model for split seven(the blue, fold_7 as the validation set and the remaining one is as training set) are presented in the following Table 18 concerning Training, Validation, and Test accuracy as well as Training, Validation, and Test loss in the form of a percentage.

Table 18. Training, Validation, and Testing accuracy and loss of the split seven

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.38%	99.31%	99.33%	2.04%	4.76%	0.67%

The graphs in Figure 35 show us the results of the proposed model for split seven during training concerning training versus validation accuracy and training versus validation loss.

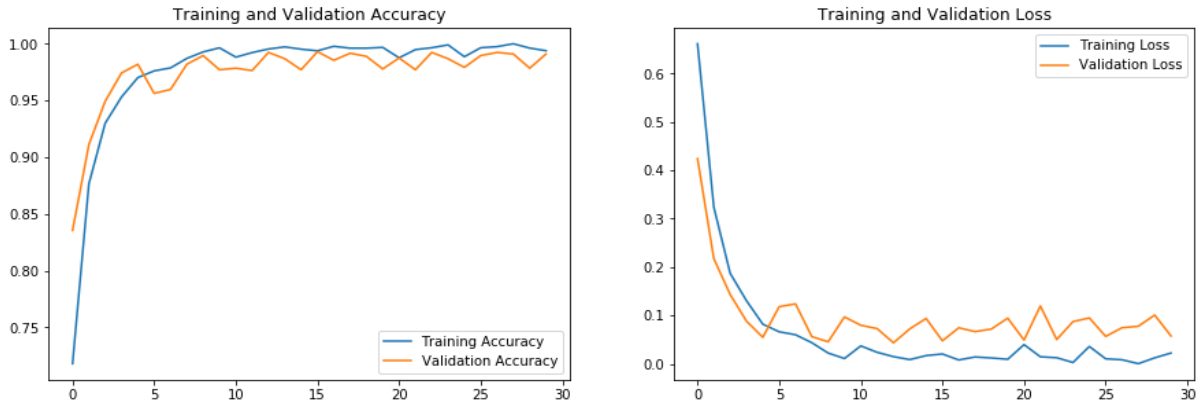
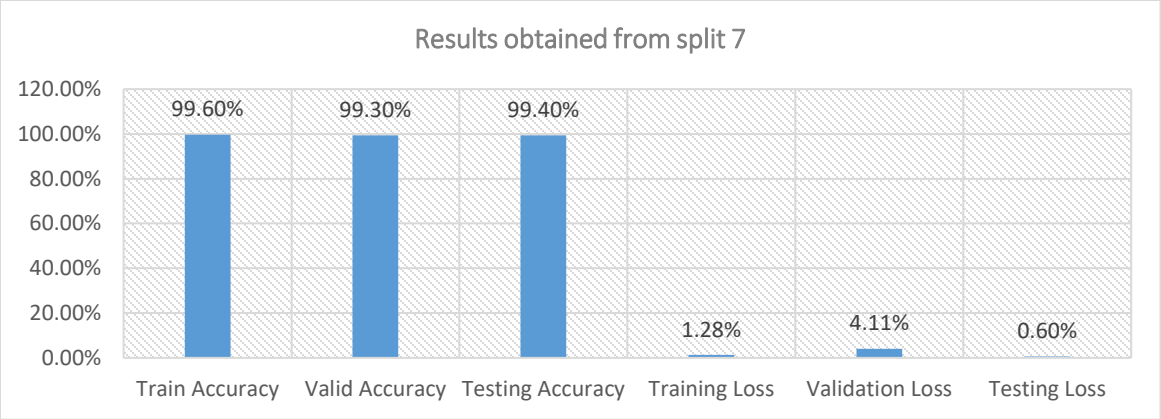


Figure 35. Training vs validation accuracy and Training vs validation loss of split seven



The detailed experimental result of the proposed model is illustrated in the form of the confusion matrix, precision, recall, f1-scores, support, and confusion matrix for split seven for each class illustrated in Table 19 and Figure 36.

Table 19. Precision, Recall, F1-Score, and Support split seven for test data

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.99	0.99	0.99	387
Health Enset	0.99	0.98	0.99	435
Leaf Spot	0.99	1.00	0.99	279
Rootmealybug	0.99	0.99	0.99	398

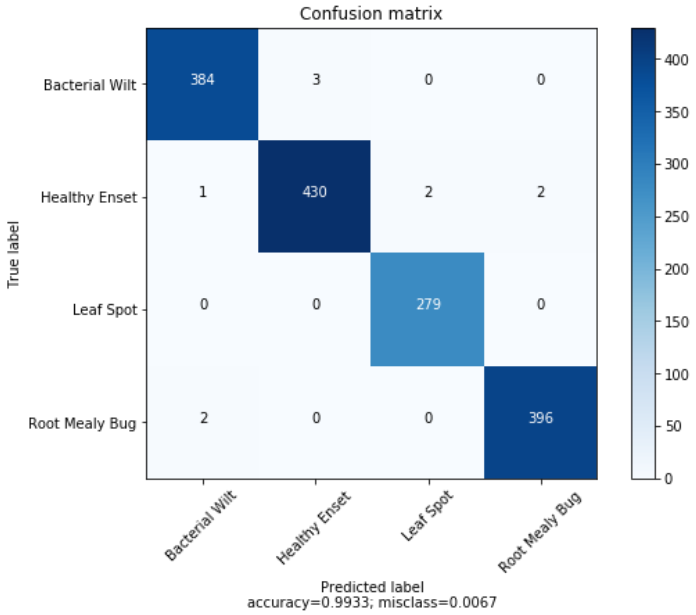


Figure 36. Confusion matrix on the Testing dataset for split seven

Figure 36 illustrates the classification accuracy as well as classification loss for Fold seven of the proposed Model. As we can see from the plotted table graph in the first class from a total of 387 images 384 images are accurately classified as bacterial wilt,3 images are wrongly classified as healthy enset. In the second class from a total of 435 images, 430 images are accurately classified as healthy enset, 1 image is classified wrongly as bacterial wilt,2 images are wrongly classified as leafspot, and also 2 images are wrongly classified root mealybug. In the third class, all of the 279 images are accurately classified as leafspot. In the fourth class from a total of 398 images 396 images are accurately classified as root mealybug,2 images are wrongly classified as bacterial wilt.

Split 8: The overall experimental results obtained from the proposed model for split eight (the blue, fold_8 as the validation set and the remaining one is as training set) are presented in the following Table 20 concerning Training, Validation, and Test accuracy as well as Training, Validation, and Test loss in the form of a percentage.

Table 20. Training, Validation, and Testing accuracy and loss of the split eight

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.91%	99.24%	99.27%	0.42%	6.20%	0.73%

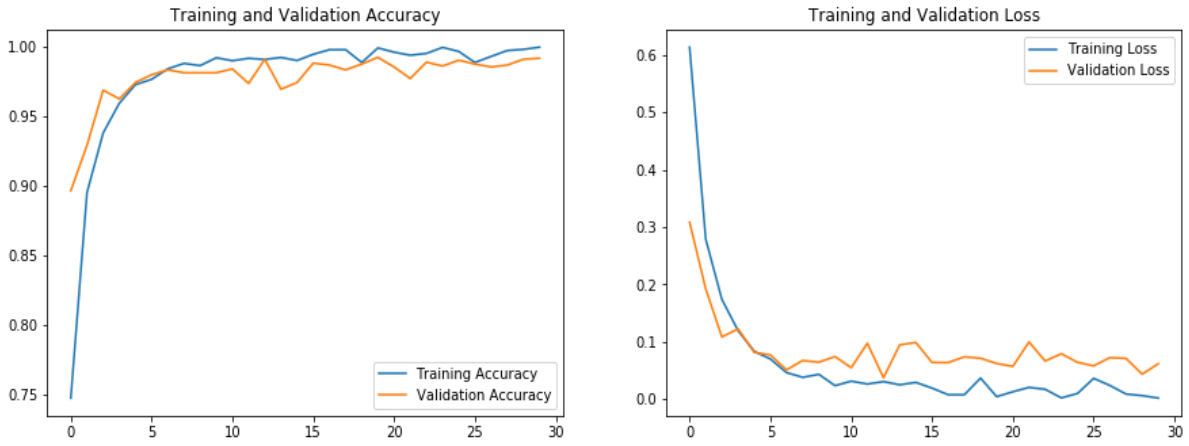
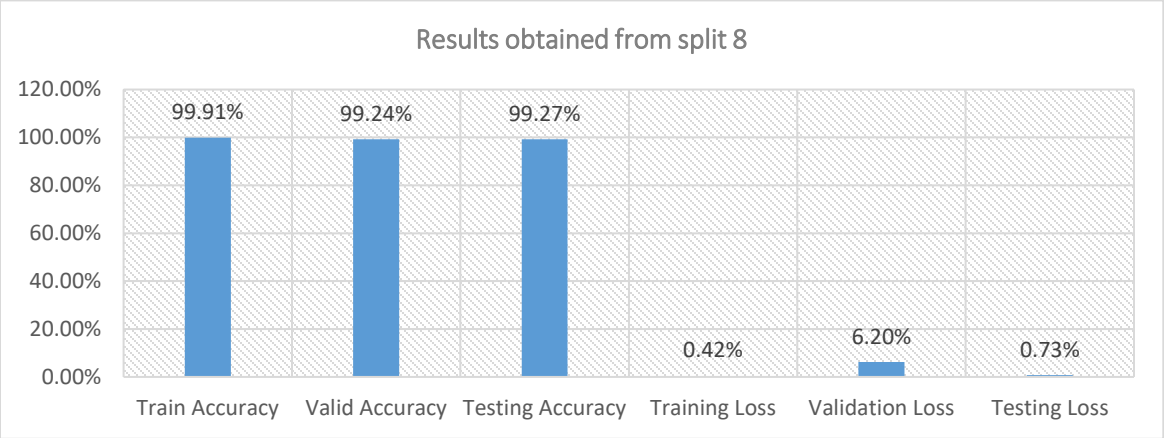


Figure 37. Training vs validation accuracy and Training vs validation loss of split eight

The graphs in Figure 37 show us the results of the proposed model for split eight during training concerning training versus validation accuracy and training versus validation loss.



The detailed experimental result of the proposed model is presented in the form of the confusion matrix, precision, recall, f1-scores, support, and confusion matrix for split eight for each class illustrated in Table 21 and Figure 38.

Table 21. Precision, Recall, F1-Score, and Support split eight for test data

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.99	0.99	0.99	387
Health Enset	0.99	0.98	0.98	435
Leaf Spot	0.97	1.00	0.99	277
Rootmealybug	0.99	0.99	0.99	398

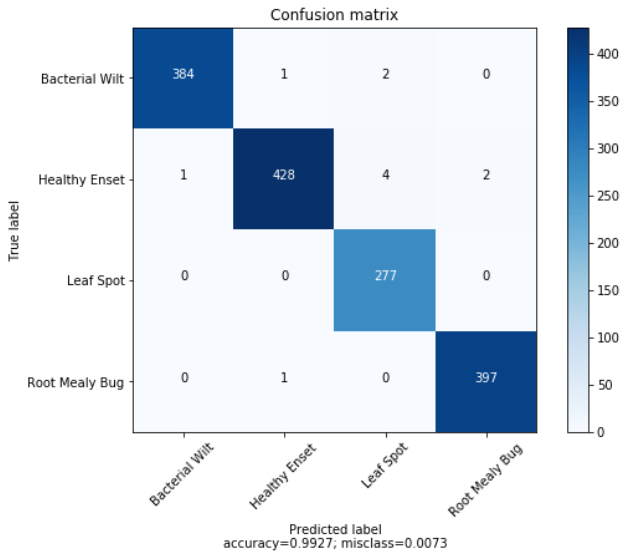


Figure 38. Confusion matrix on the Testing dataset for split eight

Figure 38 illustrates the classification accuracy as well as classification loss for split eight of the proposed Model. As we can see from the plotted table graph in the first class from a total of 387 images 384 images are accurately classified as bacterial wilt,1 image is wrongly classified as healthy enset, and also 2 images are wrongly classified as leafspot. In the second class from a total of 435 images, 428 images are accurately classified as healthy enset, 1 image is classified wrongly as bacterial wilt,4 images are wrongly classified as leafspot, and 2 images are wrongly classified as root mealybug. In the third class, all of the 277 images are accurately classified as leafspot. In the fourth class from a total of 398 images 397 images are accurately classified as root mealybug,1 image is wrongly classified as healthy enset.

Split 9: The overall experimental results obtained from the proposed model split nine (the blue, fold_9 as the validation set and the remaining one is as training set) is presented in the following Table 22 concerning Training, Validation, and Test accuracy as well as Training, Validation, and Test loss in the form of a percentage.

Table 22. Training, Validation, and Testing accuracy and loss of the split nine

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.90%	98.54%	98.53%	0.37%	8.76%	1.47%

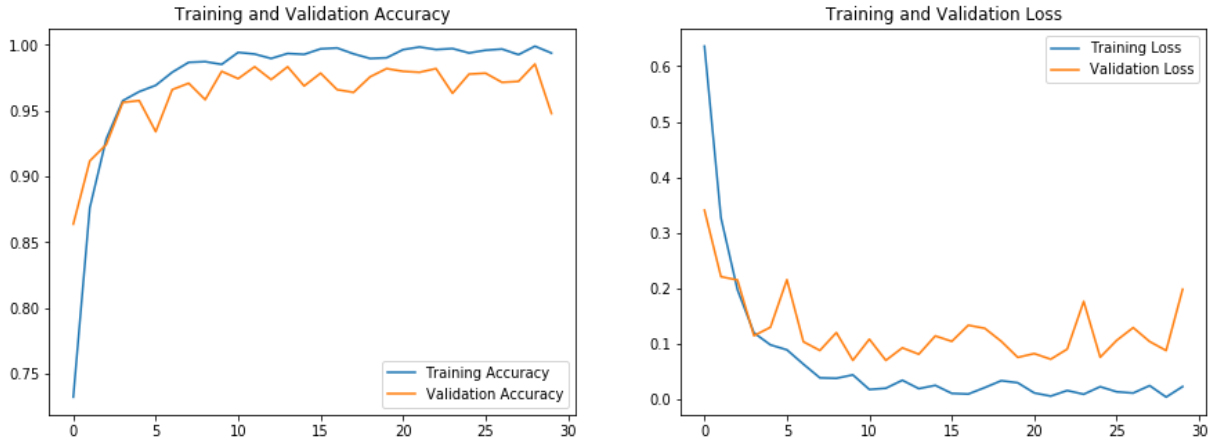
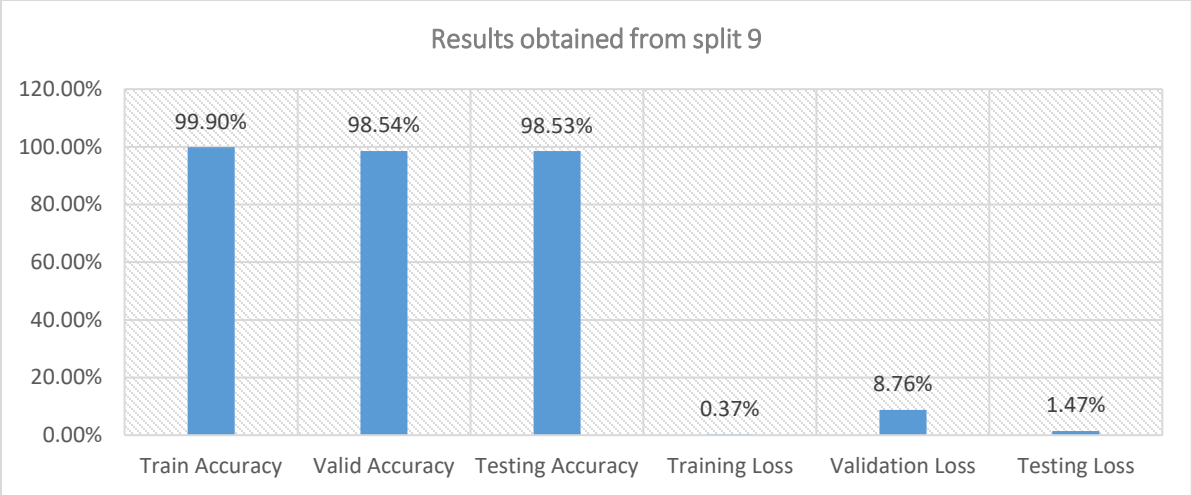


Figure 39. Training vs validation accuracy and Training vs validation loss of split nine

The graphs in Figure 39 show us the results of the proposed model for split nine during training concerning training versus validation accuracy and training versus validation loss.



The detail experimental result information of the proposed model is presented in the form of the confusion matrix, precision, recall, f1-scores, support, and confusion matrix for split nine for each class illustrated inTable 23 and Figure 40.

Table 23. Precision, Recall, F1-Score, and Support split nine for test data

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.98	0.96	0.97	387
Health Enset	0.97	0.98	0.97	435
Leaf Spot	0.99	1.00	0.99	279
Rootmealybug	0.99	0.98	0.99	398

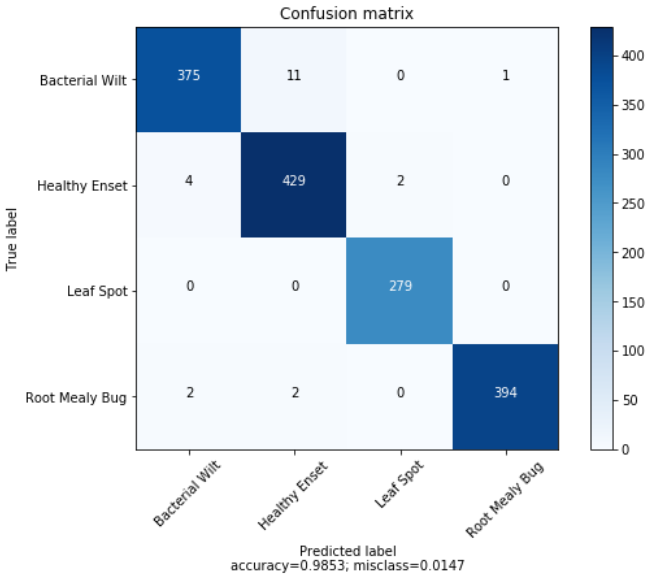


Figure 40. Confusion matrix on the Testing dataset for split nine

Figure 40 illustrates that the classification accuracy as well as classification loss for Fold nine of the proposed Model. As we can see from the plotted table graph in the first class from a total of

387 images 375 images are accurately classified as bacterial wilt, and 11 images are wrongly classified as healthy enset, and also 1 image is wrongly classified as root mealybug. In the second class from a total of 435 images, 429 images are accurately classified as healthy enset, 4 images are classified wrongly as bacterial wilt, and also 2 images are wrongly classified as leafspot. In the third class, all of the 279 images are accurately classified as leafspot. In the fourth class from a total of 398 images 394 images are accurately classified as root mealybug,2 images are wrongly classified as bacterial wilt, and also 2 images are wrongly classified as healthy enset.

Split 10: The overall experimental results obtained from the proposed model for split ten (the blue, fold_10 as the validation set and the remaining one is as training set)are presented in the following Table 24 concerning Training, Validation, and Test accuracy as well as Training, Validation, and Test loss in the form of a percentage.

Table 24. Training, Validation, and Testing accuracy and loss of the split ten

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.50%	99.51%	99.53%	1.51%	2.37%	0.47%

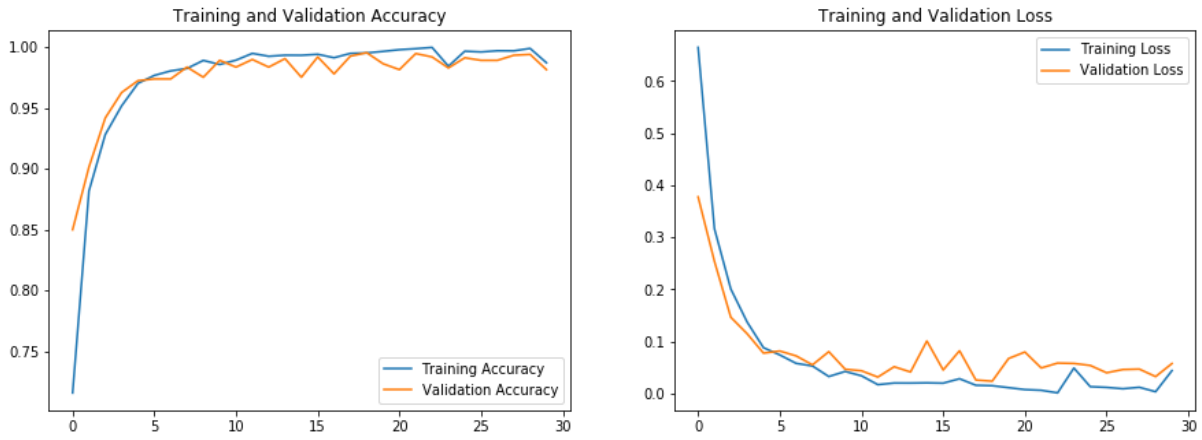
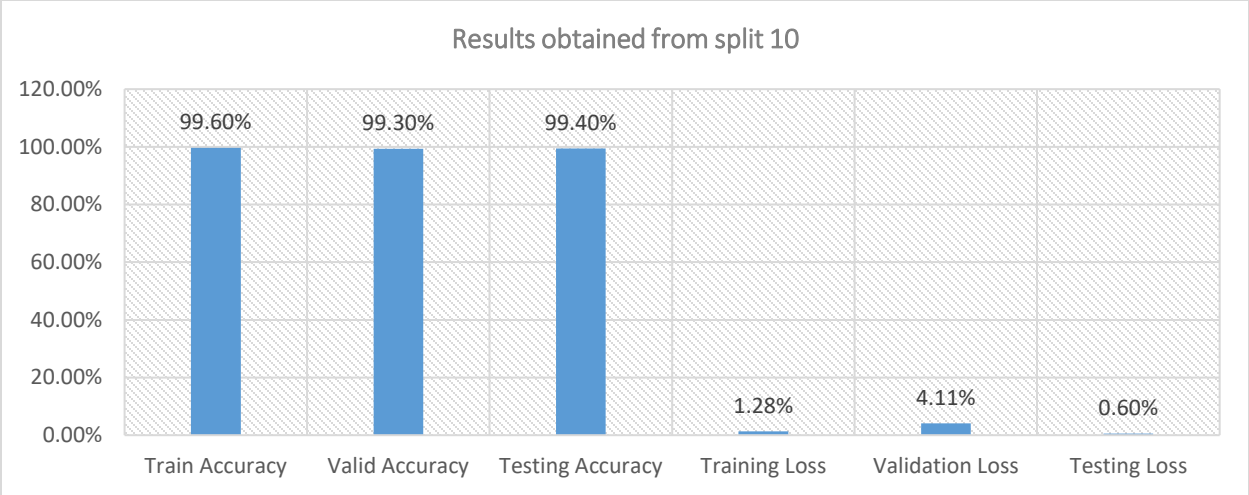


Figure 41. Training vs validation accuracy and Training vs validation loss of split ten

The graphs in Figure 41 show us the results of the proposed model for split nine during training concerning training versus validation accuracy and training versus validation loss.



The detail experimental result information of the proposed model is presented in the form of the confusion matrix, precision, recall, f1-scores, support, and confusion matrix for Fold ten for each class which is illustrated in Table 25 and Figure 42.

Table 25. Precision, Recall, F1-Score, and Support split ten for test data

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.99	0.99	0.99	380
Health Enset	0.99	0.98	0.99	427
Leaf Spot	0.99	1.00	0.99	278
Rootmealybug	0.99	1.00	0.99	397

Figure 42 illustrates that the classification accuracy as well as classification loss for split ten of the proposed Model. As we can see from the plotted table graph in the first class from a total of 380 images 378 images are accurately classified as bacterial wilt, 1 image is wrongly classified as healthy enset,1 and also 1 image is wrongly classified as root mealybug. In the second class from a total of 427 images, 422 images are accurately classified as healthy enset, 3 images are classified wrongly as bacterial wilt,1 image is wrongly classified as leafspot, and also 1 image is wrongly classified as root mealybug. In the third class, all of the 278 images are accurately classified as leafspot. In the fourth class, all of the 397 images are accurately classified as root mealybugs.

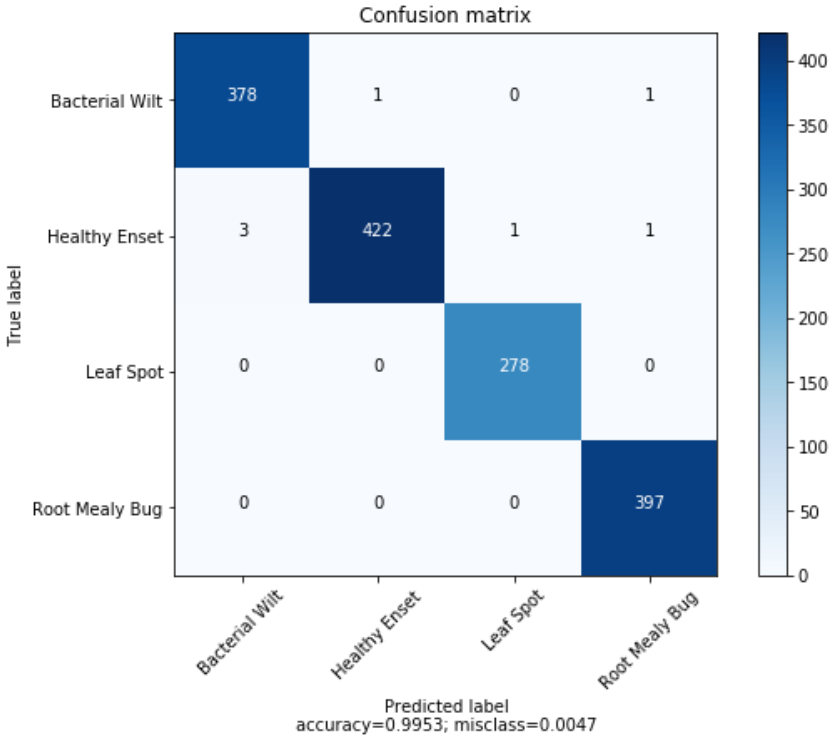


Figure 42. Confusion matrix on the Testing dataset for split ten

Table 26. Summary results of the proposed model using 10-fold Cross-validation

Splits	Accuracy			Loss		
	Training	Validation	Classification	Training	Validation	Classification
Split 1	99.93%	99.03%	99.07%	0.31%	4.54%	0.93%
Split 2	99.96%	99.37%	99.33%	0.13%	2.65%	0.67%
Split 3	99.48%	99.10%	99.07%	1.84%	3.29%	0.93%
Split 4	99.58%	98.75%	98.73%	1.15%	5.21%	1.27%
Split 5	99.62%	98.96%	98.93%	1.40%	4.52%	1.07%
Split 6	99.28%	98.89%	98.87%	2.28%	5.10%	1.13%
Split 7	99.38%	99.31%	99.33%	2.04%	4.76%	0.67%
Split 8	99.38%	99.07%	98.93%	1.87%	4.33%	1.07%
Split 9	99.90%	98.54%	98.53%	0.37%	8.76%	1.47%
Split 10	99.50%	99.51%	99.53%	1.51%	2.37%	0.47%

Table 26 shows us the summary results of experiments using 10-fold cross-validation strategies. From those experiments, split ten gives us the best result and split nine gives us less result among the other ten splits.

5.1.1.2. Experiment 2: Training using train test split strategy

In this section, three experiments are conducted with the proposed model using a different learning rate by applying the train test split strategy. Table 27 below shows us different results concerning accuracy and loss metrics such as training, validation, and testing in the form of percentages separately. Results show that giving higher learning rates has less accuracy than that of smaller learning rates. Therefore, among the three different learning rates, the learning rate of 0.001 gives us a good result in the proposed model.

Table 27. Experimental results of the proposed model using different learning rate

Learning Rate	Epoch	Accuracy			Loss		
		Training	Validation	Classification	Training	Validation	Classification
0.001	30	97.56%	98.44%	98.36%	7.20%	4.76%	1.64%
0.01	30	95.44%	97.36%	97.91%	1.261%	6.88%	2.09%
0.1	30	94.38%	97.06%	97.12%	16.06%	11.34%	2.88%

5.1.1.3. Experiment 3: Training using train test split strategy

In this section, two experiments are conducted with the proposed model using the different activation functions in the output layer. Table 28 below shows us different results concerning accuracy and loss metrics such as training, validation, and testing in the form of percentage separately. The experimental results show that using the SoftMax activation function is more preferred for proposed multiclass classification problems and using the sigmoid activation function is better for binary classification. Therefore, using the SoftMax activation function gives us an optimal result in the proposed model.

Table 28. Experimental results of the proposed model using different activation functions

Activation function	Epoch	Accuracy			Loss		
		Training	Validation	Classification	Training	Validation	Classification
SoftMax	30	96.41%	98.14%	97.97%	10.00%	5.53%	2.03%
Sigmoid	30	97.54%	97.06%	97.12%	7.11%	6.31%	2.88%

5.1.1.4. Experiment 4: Training using train test split strategy

In this section, several experiments are conducted with the proposed model using different epochs. Table 29 below shows us some of the results obtained concerning accuracy and loss metrics such as training, validation, and testing in the form of percentage separately. Results show that giving a higher epoch value during the training process gives better accuracy than

that of smaller epoch values. Therefore, in this experiment, the epoch value of 30 gives us an optimal result in the proposed model.

Table 29. Experimental results of the proposed model using different epochs

Epoch Value	Accuracy			Loss		
	Training	Validation	Classification	Training	Validation	Classification
20	98.31%	98.48%	97.65%	4.3%	5.7%	6.23%
25	94.6%	95.2%	97.10%	15.5%	13.2%	8.6%
30	98.31%	98.48%	97.65%	4.3%	5.7%	6.23%

5.2. Comparison with other Pre-Trained Models

In this section, two experiments are conducted using two mostly studied classical pre-trained models namely MobileNet and InceptionV3 by applying transfer learning to compare the classification accuracy of these models with the proposed model using the proposed dataset. In this implementation, a similar dataset and similar hyperparameter configuration are used except the convolution base (feature extractor base). The MobileNet architecture is selected because it is an efficient model for mobile and embedded vision applications, and the Inceptionv3 model is selected because of its complicated features.

5.2.1. Experiment 1: Experimental results of InceptionV3 Model

In this experiment to retrain the InceptionV3 model with the proposed dataset one of the transfer learning strategy is applied by freezing convolution base and retrain fully collected layers. To make clear, the networks of the InceptionV3 model are retrained using proposed configured hyperparameters illustrated in and

Table 3 without making any fine-tuning method in the convolution base and only the output layers are changed to proposed four classes. The overall accuracy and loss results obtained from the experiment of the inceptionv3 model are illustrated in Table 30 concerning classification accuracy and loss metrics for the training dataset, validation dataset, and test dataset respectively.

Table 30. Overall classification accuracy and loss of InceptionV3 Model

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.46%	97.71%	97.69%	1.70%	6.71%	2.31%

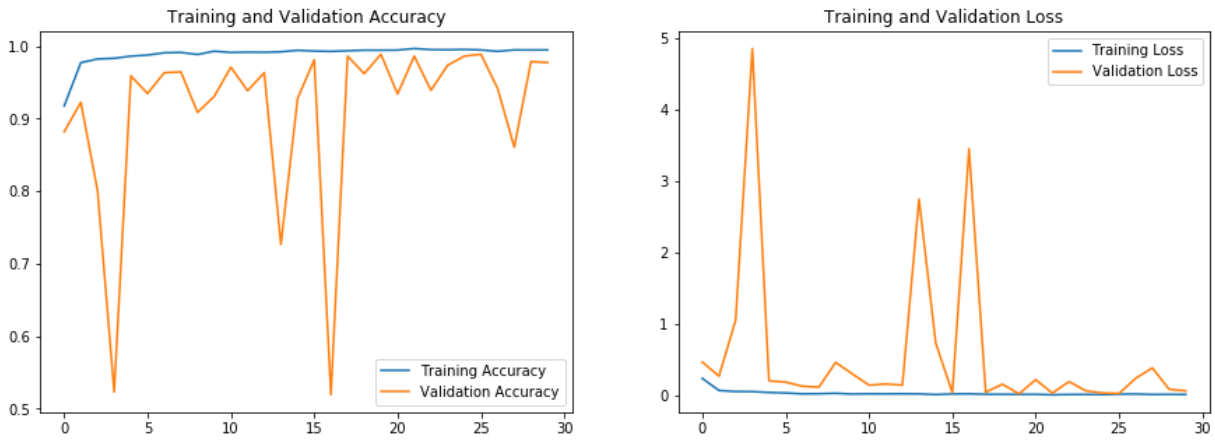


Figure 43.Training vs validation accuracy and Training vs validation loss inceptionv3 Model

The graphs in Figure 43 show us the results of the inceptionv3 model during training concerning training versus validation accuracy and training versus validation loss.

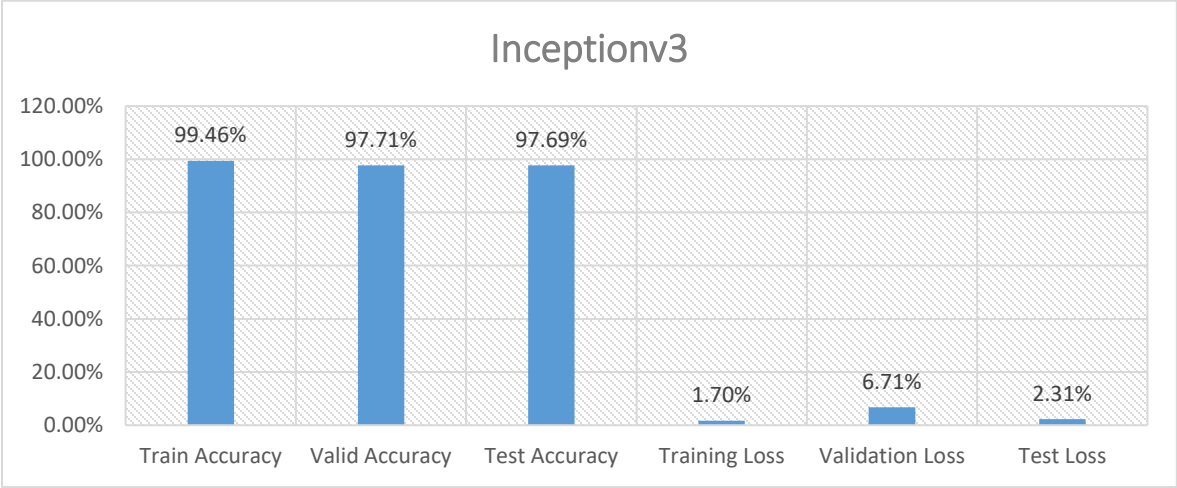


Table 31 and Figure 44 illustrates the detail information of the precision, recall, F1-scores, and support for each class and their confusion matrix is plotted.

Table 31.Precision, Recall, F1-Score, and Support of InceptionV3 Model

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.926	1.00	0.967	428
Health Enset	1.00	0.930	0.96	591
Leaf Spot	0.978	1.00	0.98	313
Rootmealybug	1.00	1.00	1.00	442

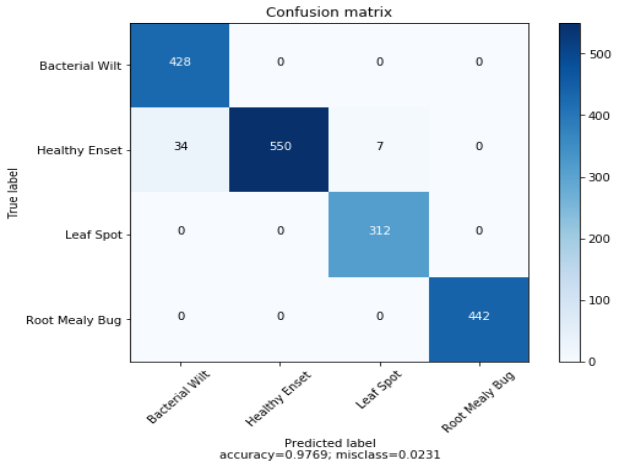


Figure 44.Confusion matrix of inceptionv3 Model

Figure 44 above illustrates that the classification accuracy as well as classification loss of the inceptionv3 Model. As we can see from the plotted table graph in the first class all of the 428 images are accurately classified as bacterial wilt without any mistake. In the second class from a total of 591 images, 550 images are accurately classified as healthy enset,34 images are classified wrongly as bacterial wilt as wee as 7 images are wrongly classified as leafspot.in the third class all of the 312 images are accurately classified as leafspot.in the fourth class also all of the 442 images are classified as root mealybug without any error.

5.2.2. Experiment 2: Experimental results of MobileNet Model

In this experiment to retrain the MobileNet model with the proposed dataset one of the transfer learning strategy is applied by freezing convolution base and retrain fully collected layers. To make clear, the networks of the MobileNet model are retrained using proposed configured hyperparameters illustrated in

Table 3 above without making any fine-tuning method in the convolution base and only the output layers are trained to propose four classes.

5.2.2.1. Results and Evaluation of MobileNet Model

The detail experimental result of the MobileNet model is explained in the form of the confusion matrix, precision, recall, f1-scores, and support concerning top experiment values for each class below. Additionally, the plotted graphs below show us the results of the MobileNet model during training concerning training versus validation accuracy and training versus validation

loss. In general, in between some epochs, the validation accuracy is decreased; there is a gap between training and validation accuracy the reason is the model is overfitted in these epochs. The overall accuracy and loss results obtained from the experiment of the MobileNet model are illustrated in Table 32 concerning classification accuracy and loss metrics for the training dataset, validation dataset, and test dataset respectively.

Table 32: Overall classification accuracy and loss of MobileNet Model

Accuracy			Loss		
Training	Validation	Classification	Training	Validation	Classification
99.71%	99.66%	79.81%	0.87%	0.91%	20.19%

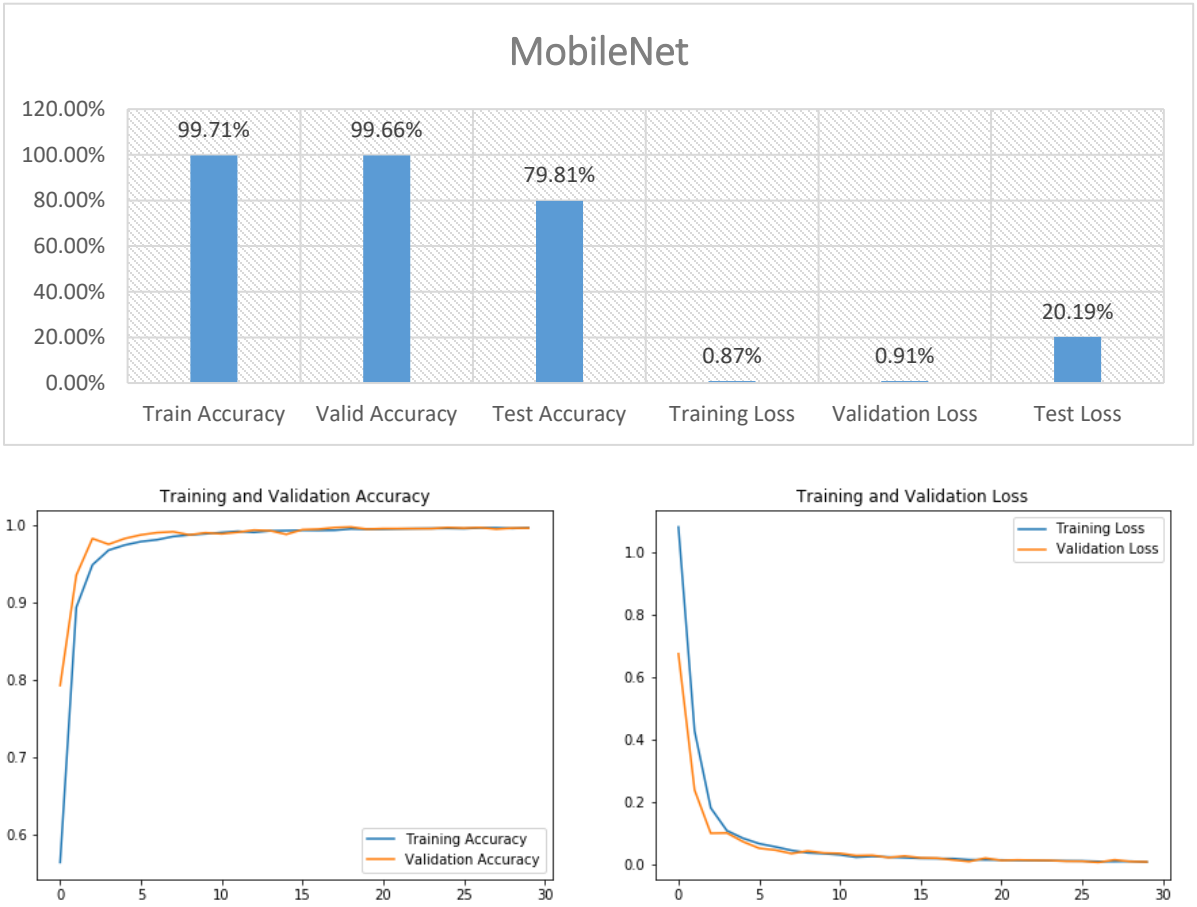


Figure 45.Training vs validation accuracy and Training vs validation loss MobileNet Model

The graphs in Figure 45 show us the results of the MobileNet model during training concerning training versus validation accuracy and training versus validation loss. Table 33 and Figure 46

illustrates the detail of the precision, recall, F1-scores, and support for each class and their confusion matrix is plotted.

Table 33.Precision, Recall, F1-Score, and Support of MobileNet model

Class	Precision	Recall	F1-Score	Support
Bacterial Wilt	0.54	1.00	0.70	428
Health Enset	1.00	0.66	0.80	591
Leaf Spot	1.00	0.61	0.79	312
Rootmealybug	1.00	90	0.95	442

Figure 46 illustrates that the classification accuracy as well as the classification loss MobileNet Model. As we can see from the plotted table graph in the first class, all of the 428 images are classified accurately without any mistake as bacterial wilt. In the second class from a total of 591 images, 394 images are accurately classified as healthy enset,197 images are classified wrongly as bacterial wilt.

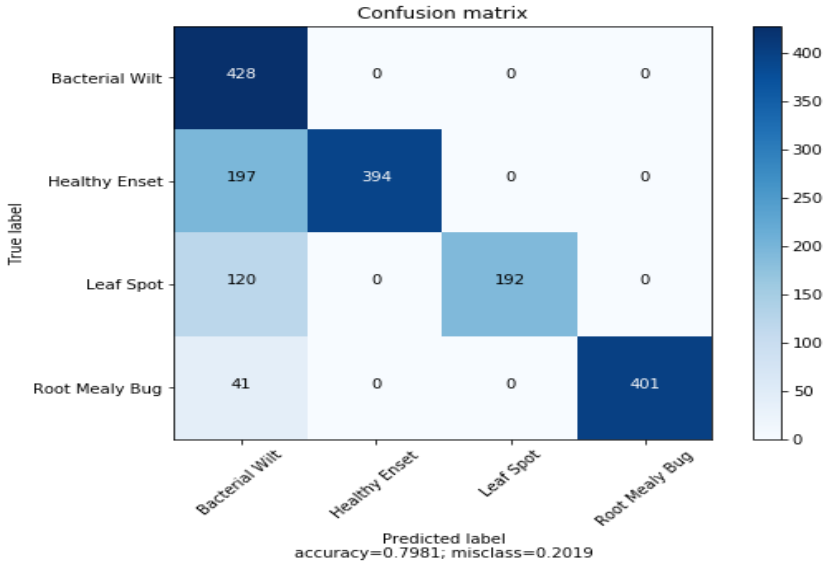


Figure 46.Confusion matrix for MobileNet Model

In the third class, from a total of 312 images, 192 images are accurately classified as leafspot and the remaining 120 images are wrongly classified bacterial wilt. In the fourth class, from a total of 442 images,401 images are accurately classified as root mealybug, and the remaining 41 images are wrongly classified as bacterial wilt.

5.3. Summary

As we discussed in the previous sections, the experiments are conducted using the proposed model and other two classical CNN pre-trained models namely InceptionV3 and MobileNet. All experiments are conducted with the same dataset, software, hardware, and hyperparameters configurations. All models are tested using a test dataset which is unseen during the training of the model. To measure and compare the performance of the pretrained classical models with the proposed model, the classification accuracy metrics are used and the proposed model gives us higher classification accuracy results. The following plotted graphs show us the summary results of training, validation, and test accuracy of three top experiments.

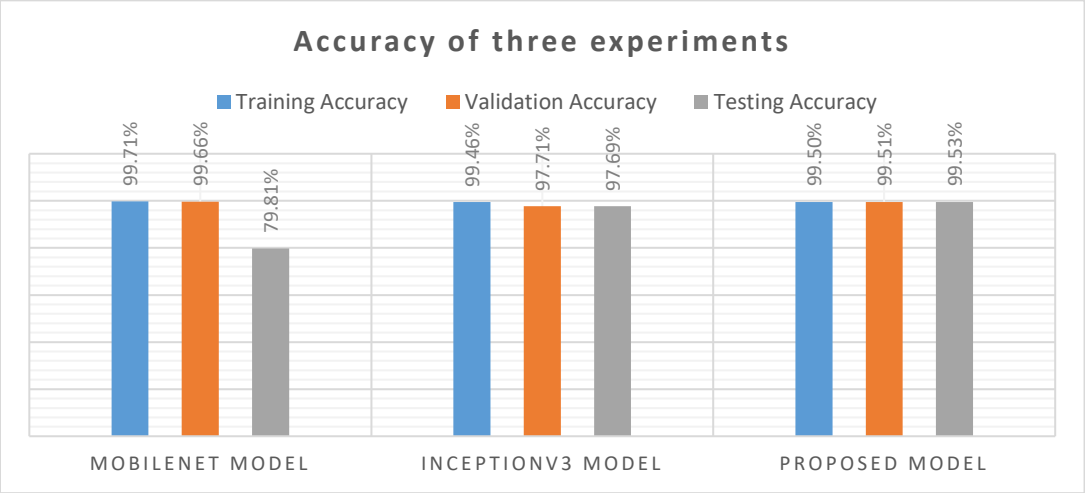


Figure 47.Training, validation and testing accuracy of three experiments

As we can see from the above graph the percentage of the best training, validation and testing accuracy of the proposed model are 99.50%,99.51%, and 99.53% respectively. The percentage of training, validation and testing accuracy of the MobileNet model are 99.71%,99.66%, and 79.81% respectively. The percentage of training, validation and testing accuracy of the InceptionV3 model are 99.45%,97.71%, and 97.69% respectively. When we compare the training, validation, and testing accuracy between each of the three experiments, the proposed model performs higher results. These show that the detection and classification ability of the proposed model is high compared to other models. To measure and compare the inconsistency of the pretrained classical models with the proposed model, the classification loss metrics are used. The following plotted graphs show us the summary results of the training, validation, and test loss result values.

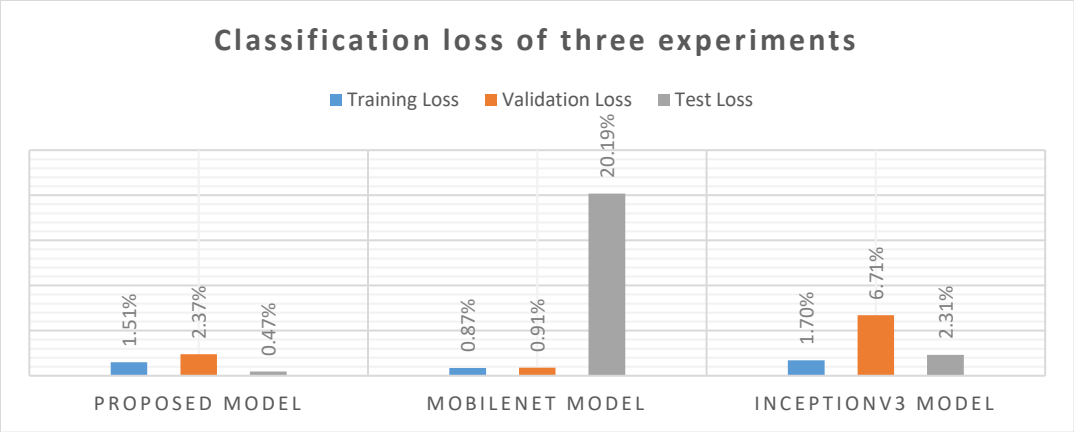


Figure 48.Training, validation and testing loss of three experiments

As we can see from the above graph, the percentage of the training, validation, and testing loss of the proposed model are 1.51%,2.37%, and 0.47% respectively. The percentage of the training, validation and testing loss of MobileNet model are 0.87%,0.91%, and 20.19% respectively. The percentage of the training, validation, and testing loss of the InceptionV3 model are 1.70%,6.71%, and 2.31% respectively. When we compare these results, the proposed model gives less loss classification results. Therefore, the proposed model is doing well both in the training and dataset. So we can say that the ability of the proposed model to detect and classify enset disease is high compared to other models.

CHAPTER SIX

CONCLUSION AND FUTURE WORK

6.1. Conclusion

In Ethiopia, the agricultural sector is still one of the most important sectors over which the majority of the Ethiopian population depends on. This sector suffers from several critical problems like plant diseases that decrease the production and quality of yield. Enset is one of the most important staples and co-stable food plants for 20 million people in Ethiopia. However, its production has been threatened by several diseases, among those, bacterial wilt, root mealybug, and leafspot are the most destructive ones. This necessitates the development of diagnostics tools as an urgent task to detect the plant's diseases at their early stages. This study aimed to build and test a model to detect and identify the three most critical diseases of the enset plant. A CNN based model is proposed and implemented as part of the thesis work as a decision-making tool for the automatic detection system. The CNN based deep learning approaches are selected for the thesis work because computer vision with deep learning methodologies has an outperformed in solving a total number of plant disease problems by extracting features without human intervention. The work of Yidnekachew [48] has also applied the same approach with a focus on Enset bacterial wilt. Experimental results of this study show that the proposed model has higher performance, computationally efficient, simple, and has a minimum classification accuracy of 98.40%, maximum classification accuracy of 99.53%, and average classification of 99.065%. This result is comparable in performance to the closely related work of Yidnekachew [48] to the work and is higher in comparison to the existing classical CNN state-of-the-art models, such as MobileNet and inception V3 as it is discussed in Section 5.2. This identification accuracy indicates that the proposed model is more effective and gives a better solution to identify enset plant diseases. Moreover, the proposed model can serve as a decision support tool to help farmers and experts around the area to identify the disease in the enset plant.

6.2. Contributions

The three critical enset plant diseases (enset bacterial wilt, enset Leaf spot, and Root mealybug diseases) identification model is proposed for classification among infected and healthy enset plants. This study extended the work of Yidnekachew [48] on enset bacterial wilt one-step ahead by adding two more common Enset diseases. The high accuracy of classification results that the

model performed in this study together with the similar result by Yidnekachew [48] demonstrates that the CNN based technology can help in the automatic detection of the three common Enset diseases through the application of deep learning approach of computer vision. Hence, with further fine-tuning and technological development, the result shows that farmers and experts can take an image of the enset plant with the symptoms, and then the model will help them identify the type of the disease automatically. Also, this well organized healthy and diseased enset plant dataset can be used in the future for related studies. Moreover, unlike pre-trained CNN models, the proposed model can be trained and get better results using small-sized networks with fewer parameters, taking less time, and fewer hardware resources and fewer data. This is demonstrated by the comparative evaluation of the results of the proposed model with other classical pretrained CNN models.

6.3. Future work

The protection of crops from disease-causing agents and curing infected crops is essential to boost food security in Ethiopia. This is also an active research area in many Agricultural Research Institutions of the country. Adding computer vision approaches with image analysis techniques to such initiatives is indispensable as many related studies showed that the techniques have outperformed the human-expert based identification and classification of crop diseases particularly at earlier stages of the diseases. Together with Yidenkachew's [48] work, this study showed that image analysis techniques with CNN models can be used to detect and classify the three most critical diseases of enset plants. We propose the following as relevant areas of future works of this study.

- The image dataset developed was collected from Hawassa and Worabe Research Institutes. Further validation of the model with Enset diseased and normal images from different agro-ecological regions would help to verify the performance of the model in all Enset types as well as the behaviors of the disease-causing bacteria across the agro-ecological zones.
- The recent development in the internet of things (IoT) has an important role in real-time surveillance of crop health in the field. A further test of the model with (IoT) enabled real-time data also boost confidence in the performance of the model.

- In addition to this, the more accurate Mobile and online-based GUI application for image-taking and real-time interfacing with a central image analysis system is a prerequisite for usage of the models by farmers and agricultural domain experts.
- The CNN-based model is tested with the three most common types of Enset diseases. A further study is essential to test the model with additional enset diseases as well as yield maturity stages.
- As a staple food for large population size in Ethiopia, the creation of large size high-quality natural images of enset plant diseases and making them available for researchers and practitioners is highly required. We advise universities and research institutes to work in collaboration to this end.

REFERENCE

[1]. Addis, T., Alemu, T., Lemawork, S., Tadesse, E., Gemu, M., & Blomme, G. (2010). Biology, Geographical Distribution, Prevention and Control of the Enset Root Mealybug, Cataenococcus enset (Homoptera: Pseudococcidae) in Ethiopia.

[2]. Reddy, P. P. (2015). Plant Protection in Tropical Root and Tuber Crops, Enset, Ensete ventricosum. Plant Protection in Tropical Root and Tuber Crops. Springer, New Delhi

[3]. Blomme, G., Dita, M., Jacobsen, K. S., Pérez Vicente, L., Molina, A., Ocimati, W., Poussier, S. … Prior, P. (2017). Bacterial Diseases of Bananas and Enset: Current State of Knowledge and Integrated Approaches Toward Sustainable Management. Frontiers in plant science, 8, 1290. doi:10.3389/fpls.2017.01290

[4]. Owomugisha, Godliver & Quinn, John & Mwebaze, Ernest & Lwasa, James. (2019). Automated Vision-Based Diagnosis of Banana Bacterial Wilt Disease and Black Sigatoka Disease.

[5]. Baya, B. B., Nzeadibe, T. C., Nwosu, E. O., & Uzomah, N. L. (2019). Climate change, food insecurity and household adaptation mechanisms in Amaro Ward, Southern Region of Ethiopia. Journal of Agricultural Extension and Rural Development, 11(5), 106-113.

[6]. Walle, T. M. (June 2014). Farmer's indigenous knowledge and assessment of enset (Ensete ventricosum Welw. Cheesman) cultivars for major insect pests in Ojojia water shade Kembata-tembaro zone, South Ethiopia. Sky Journal of Agricultural Research Vol. 3(6), pp., 082 - 088.

[7]. Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using Deep Learning for Image-Based Plant Disease Detection. *Frontiers in Plant Science*, *7*, 1–12. https://doi.org/10.3389/fpls.2016.01419

[8]. P.Ferentinos, K. (February 2018). Deep learning models for plant disease detection and diagnosis. Computers and Electronics in Agriculture, 311-318.

[9]. Asifullah Khan, A. S. (January 2019). A Survey of the Recent Architectures of Deep Convolutional Neural Networks. Research gate.

[10]. Zebin, Tahmina. (2017). Training Deep Neural Networks in Python Keras Framework(Tensorflow Backend) with Inertial Sensor Data for Human Activity Classification.

[11]. Pembelajaran, Belajar, and Mesin Indonesia. 2019. “Belajar Pembelajaran Mesin Indonesia Student Notes: Convolutional Neural Networks (CNN) Introduction Convolution Operation Basic Convolution Operation.” 1–16.

[12]. Tichkule, Shivani K., and Dhanashri H. Gawali. 2017. “Plant Diseases Detection Using Image Processing Techniques.” Proceedings of 2016 Online International Conference on Green Engineering and Technologies, IC-GET 2016: 1–6.

[13]. Wang, Haiguang, Guanlin Li, Zhanhong Ma, and Xiaodong Li. 2012. “Image Recognition of Plant Diseases Based on Backpropagation Networks.” 2012 5th International Congress on Image and Signal Processing, CISP 2012 (Crisp): 894–900.

[14]. Hossain, Md Selim, et al. 2018. “Recognition and Detection of Tea Leaf’s Diseases Using Support Vector Machine.” Proceedings - 2018 IEEE 14th International Colloquium on Signal Processing and its Application, CSPA 2018 (March): 150–54.

[15]. Vipinadas, M. J., & Thamizharasi, A. International Journal Of Scientific & Engineering Research, Volume 7, Issue 7, July-2016 Detection and Grading of diseases in Banana leaves using Machine Learning. 7(7), 916–924.

[16]. Zhang, Xihai, et al. 2018. “Identification of Maize Leaf Diseases Using Improved Deep Convolutional Neural Networks.” IEEE Access 6(c): 30370–77.

[17]. Wallelign, S. (2017). The Thirty-First International Florida Artificial Intelligence Research Society Conference (FLAIRS-31) “Soybean *Soybean Plant Disease Identification Using Convolutional Neural Network”*. 146–151.

[18]. Sladojevic, Srdjan, et al. 2016. “Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification.” Computational Intelligence and Neuroscience.

[19]. Ouppaphan, Pichayoot. 2018. “Corn Disease Identification from Leaf Images Using Convolutional Neural Networks.” ICSEC 2017 - 21st International Computer Science and Engineering Conference 2017, Proceeding 6: 233–38.

[20]. Mohanty, Sharada P., David P. Hughes, and Marcel Salathé. 2016. “Using Deep Learning for Image-Based Plant Disease Detection.” Frontiers in Plant Science 7(September).

[21]. Jeon, Wang-Su, and Sang-Yong Rhee. 2017. “Plant Leaf Recognition Using a Convolution Neural Network.” The International Journal of Fuzzy Logic and Intelligent Systems 17(1.

[22]. Szegedy, Christian, et al. 2015. “Rethinking the Inception Architecture for Computer Vision.” http://arxiv.org/abs/1512.00567.

[23]. Adi, Aaditya Prakash. 2017. “One by One [1 x 1] Convolution - Counter- Intuitively Useful Whenever I Discuss or Show GoogleNet Architecture, One Question Always.”; [online; accessed; April/21/2019]; https://iamaaditya.github.io/2016/03/one-by-one-convolution/

[24]. Karen Simonyan and Andrew Zisserman; "Very Deep Convolutional Networks for Large-Scale Image Recognition”; arxiv; pdf/1409.1556,2015; [online; accessed;2019]; https://arxiv.org/abs/1409.1556v6.

[25]. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. CoRR, abs/1704.04861.

[26]. Follow, Matt Harvey. 2019. “Creating Insanely Fast Image Classic Ers with MobileNet in TensorFlow What Are MobileNets?”: 1–10.[online accesed;2019;]; https: // medium.com /hackernoon /creating-insanely-fast-image-classifiers-with-mobilenet-in-tensorflow-f030ce0a2991.

[27]. Simonyan, K., & Zisserman, A. A conference paper at ICLR in 2015” Very Deep Convolutional Networks for Large-Scale Image Recognition.” *CoRR, abs/1409.1556*.

[28]. Toda, Yosuke & Okura, Fumio. (2019). How Convolutional Neural Networks Diagnose Plant Disease. Plant Phenomics. 2019. 10.1155/2019/9237136.

[29]. Arivazhagan, S., & Ligi, S. V. International Journal of Pure and Applied Mathematics in (2018). “*Mango Leaf Diseases Identification Using Convolutional Neural Network*.” *120*(6), 11067–11079.

[30]. Shrivastava, V. K., Pradhan, M. K., Minz, S., & Thakur, M. P. (2019). Rice Plant Disease Classification Using Transfer Learning of Deep Convolution Neural Network. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-3/W6(February), 631–635. https://doi.org/10.5194/isprs-archives-xlii-3-w6-631-2019

[31]. Swasono, D. I., Tjandrasa, H., & Fathicah, C. (2019). Classification of Tobacco Leaf Pests Using VGG16 Transfer Learning. 2019 12th International Conference on Information & Communication Technology and System (ICTS), 176–181. https:// doi.org/ 10.1109/ icts. 2019.8850946

[32]. Shrivastava, V. K., Pradhan, M. K., Minz, S., & Thakur, M. P. (2019). Rice Plant Disease Classification Using Transfer Learning of Deep Convolution Neural Network. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information*

Sciences, *XLII-3/W6*(February), 631–635. https://doi.org/10.5194/isprs-archives-xlii-3-w6-631-2019.

[33]. Liu, B., Zhang, Y., He, D. J., & Li, Y. (2018). Identification of apple leaf diseases based on deep convolutional neural networks. Symmetry,10(1). https:// doi.org/ 10.3390/ sym10010011

[34]. Abdullah, N.E.; Rahim, A.A.; Hashim, H.; Kamal, M.M. Classification of Rubber Tree Leaf Diseases Using Multilayer Perceptron Neural Network. In Proceedings of the 5tn Student Conference on Research and Development-SCOReD, Shah Alam, Malaysia, 11–12 December 2007.

[35]. Tm, Prajwala & Pranathi, Alla & Kandiraju, Sai Ashritha & B Chittaragi, Nagaratna & Koolagudi, Shashidhar. (2018). Tomato Leaf Disease Detection Using Convolutional Neural Networks. 1-5. 10.1109/IC3.2018.8530532.

[36]. Rangarajan, A. K., Purushothaman, R., & Ramesh, A. (2018). Tomato crop disease classification using a pre-trained deep learning algorithm. Procedia Computer Science, 133, 1040–1047. https://doi.org/10.1016/j.procs.2018.07.070

[37]. Gensheng, H., Xiaowei, Y., Yan, Z., & Mingzhu, W. (2019). Sustainable Computing: Informatics and Systems Identification of tea leaf diseases by using an improved deep convolutional neural network. Sustainable Computing: Informatics and Systems, 24, 100353. https://doi.org/10.1016/j.suscom.2019.100353

[38]. Zhang, X., Qiao, Y., Meng, F., Fan, C., & Zhang, M. (2018). Identification of maize leaf diseases using improved deep convolutional neural networks. IEEE Access, 6(c), 30370–30377. https://doi.org/10.1109/ACCESS.2018.2844405

[39]. Howlader, M. R., Habiba, U., Faisal, R. H., & Rahman, M. M. (2019). Automatic Recognition of Guava Leaf Diseases using Deep Convolution Neural Network. Second International Conference on Electrical, Computer, and Communication Engineering, ECCE 2019, 1–5. https://doi.org/10.1109/ECACE.2019.8679421

[40]. Wikipedia contributors. (2019, November 15). Ensete ventricosum. In Wikipedia, the Free Encyclopedia. Retrieved 14:23, November 25, 2019, from https://en.wikipedia.org/w/index.php?title=Ensete_ventricosum&oldid=926306076

[41]. Xavier, F., & Boldú, P. (2018). A review of the use of convolutional neural networks in agriculture Social Internet of Things View project P-SPHERE project View project. *Article in The Journal of Agricultural Science*. https://doi.org/10.1017/S0021859618000436

[42]. Mohanty SP, Hughes DP and Salathé M (2016) Using Deep Learning for Image-Based Plant Disease Detection. Article.Front. Plant Sci. 7:1419. https://doi:10.3389/fpls.2016.01419

[43]. Owomugisha, G., Quinn, J. A., Mwebaze, E., & Lwasa, J. (2014). Automated Vision-Based Diagnosis of Banana Bacterial Wilt Disease and Black Sigatoka Disease. International Conference on the Use of Mobile ICT in Africa 2014, (June), 5.

[44]. Tigadi, B., & Sharma, B. (2016). Banana Plant Disease Detection and Grading Using Image Processing. International Journal of Engineering Science and Computing, 6(6), 6512–6516. https://doi.org/10.4010/2016.1565

[45]. Vipinadas, M. J., & Thamizharasi, A. International Journal Of Scientific & Engineering Research, Volume 7, Issue 7, July-2016 "*Detection and Grading of diseases in Banana leaves using Machine Learning*." *7*(7), 916–924.

[46]. Amara, J., Bouaziz, B. & Algergawy, A., (2017). A Deep Learning-based Approach for Banana Leaf Diseases Classification. In: Mitschang, B., Nicklas, D., Leymann, F., Schöning, H., Herschel, M., Teubner, J., Härder, T., Kopp, O. & Wieland, M. (Hrsg.), Datenbanksysteme für Business, Technologie und Web (BTW 2017) - Workshopband. Bonn: Gesellschaft für Informatik e.V.. (S. 79-88).

[47]. Bhamare, S. P., & Kulkarni, S. C. (2009). Detection of black Sigatoka on the banana tree using image-processing techniques. IOSR Journal of Electronics and Communication Engineering, 60–65. https://doi.org/10.1113/jphysiol.2012.244145

[48]. Yidnekachew, k. (2019). Developing a bacterial wilt detection model on enset crops using a deep learning approach. A thesis submitted to Addis Ababa science and Technology University.

[49]. Herdiyeni, Y., Bakhtiar, T. S., Dewanto, V., Tjahjono, B., Siregar, B. A., & Oliveira, L. S. S. (2017). Automatic identification of acacia leaf diseases in plantation forests using wavelet energy and Shannon entropy. *International Conference on Information and Communication Technology Convergence: ICT Convergence Technologies Leading the*

Fourth Industrial Revolution, ICTC 2017, *2017-Decem*, 570–575. https://doi.org/10.1109/ICTC.2017.8191043

[50]. Toda, Y., & Okura, F. (2019). *How Convolutional Neural Networks Diagnose Plant Disease*. *2019*.

[51]. Wang, Q., Qi, F., Sun, M., Qu, J., & Xue, J. (2019). *Identification of Tomato Disease Types and Detection of Infected Areas Based on Deep Convolutional Neural Networks and Object Detection Techniques*. *2019*.

APPENDIX A: INPLIMENTATION OF PROPOSED MODEL

```
import os
import numpy as np
from PIL import Image
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D,lr
from __future__ import absolute_import, division, print_function, unicode_literals
```

```
#10_fold split to create 10 folders
import os, os.path, shutil
import random
Dataset = "F:/eNsetDataSet/Training"
classes = os.listdir(Dataset)
if not os.path.exists("F:/eNsetDataSet/K_fold"):
       os.mkdir("F:/eNsetDataSet/K_fold")
for folder_path in classes:
    fold = 10
    images = os.listdir(os.path.join(Dataset, folder_path))
    random.shuffle(images)
    print(len(images))
    number_of_images = int(len(images)/fold)+1
    fold_counter = 0
    i = 0
    curr_subdir = None
    if os.path.exists("F:/eNsetDataSet/K_fold/" + folder_path):
        shutil.rmtree("F:/eNsetDataSet/K_fold/" + folder_path)
        os.mkdir("F:/eNsetDataSet/K_fold/" + folder_path)
    else: os.mkdir("F:/eNsetDataSet/K_fold/" + folder_path)
    for f in images:
# To create new subdir if necessary
        if i % number_of_images == 0:
            print("yes")
            fold_counter += 1
            subdir_name = os.path.join("F:/eNsetDataSet/K_fold/"+folder_path,
"fold_" + str(fold_counter))
            if os.path.exists(subdir_name):
                shutil.rmtree(subdir_name)
                os.mkdir(subdir_name)
            else: os.mkdir(subdir_name)
```

```
            curr_subdir = subdir_name
# move file to current dir
        f_base = os.path.basename(f)
        shutil.copy(os.path.join(Dataset,folder_path,f),
os.path.join(subdir_name,f))
        i += 1
```

```
#To plot_confusion_matrix with its classification accuracy and Classification loss
def plot_confusion_matrix(cm,
                          target_names,
                          title='Confusion matrix',
                          cmap=None,
                          normalize=False):
    import matplotlib.pyplot as plt
    import numpy as np
    import itertools
    accuracy = np.trace(cm) / np.sum(cm).astype('float')
    misclass = 1 - accuracy
    if cmap is None:
        cmap = plt.get_cmap('Blues')
    plt.figure(figsize=(8, 6))
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    if target_names is not None:
        tick_marks = np.arange(len(target_names))
        plt.xticks(tick_marks, target_names, rotation=45)
        plt.yticks(tick_marks, target_names)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
    thresh = cm.max() / 1.5 if normalize else cm.max() / 2
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if normalize:
            plt.text(j, i, "{:0.2g}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")
        else:
            plt.text(j, i, "{:,}".format(cm[i, j]),
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")
    plt.tight_layout()
    plt.ylabel('True label')
```

```
    plt.xlabel('Predicted label\naccuracy={:0.4f};
misclass={:0.4f}'.format(accuracy, misclass))
    plt.show()
```

```
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from sklearn.metrics import confusion_matrix, precision_recall_fscore_support
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
import tensorflow as tf
import os, os.path
import shutil
import random
import keras

for i in range(1, 11):
    train_dir = 'F:/eNsetDataSet/K_fold/train'
    validation_dir = 'F:/eNsetDataSet/K_fold/validation'

#To create sub/Class directories inside training directory
    if os.path.exists("F:/eNsetDataSet/K_fold/train/HealthyEnsetTree"):
                shutil.rmtree("F:/eNsetDataSet/K_fold/train/HealthyEnsetTree")
                os.mkdir("F:/eNsetDataSet/K_fold/train/HealthyEnsetTree")
    else: os.mkdir("F:/eNsetDataSet/K_fold//train/HealthyEnsetTree")

    if os.path.exists("F:/eNsetDataSet/K_fold/train/BacterialWilt"):
                shutil.rmtree("F:/eNsetDataSet/K_fold/train/BacterialWilt")
                os.mkdir("F:/eNsetDataSet/K_fold/train/BacterialWilt")
    else: os.mkdir("F:/eNsetDataSet/K_fold/train/BacterialWilt")

    if os.path.exists("F:/eNsetDataSet/K_fold/train/LeafSpot"):
                shutil.rmtree("F:/eNsetDataSet/K_fold/train/LeafSpot")
                os.mkdir("F:/eNsetDataSet/K_fold/train/LeafSpot")
    else: os.mkdir("F:/eNsetDataSet/K_fold/train/LeafSpot")

    if os.path.exists("F:/eNsetDataSet/K_fold/train/RootMealybug"):
                shutil.rmtree("F:/eNsetDataSet/K_fold/train/RootMealybug")
                os.mkdir("F:/eNsetDataSet/K_fold/train/RootMealybug")
    else: os.mkdir("F:/eNsetDataSet/K_fold/train/RootMealybug")

#To create sub/Class directories inside validation directory
```

```
    if os.path.exists("F:/eNsetDataSet/K_fold/validation/HealthyEnsetTree"):
                shutil.rmtree("F:/eNsetDataSet/K_fold/validation/HealthyEnsetTree")
                os.mkdir("F:/eNsetDataSet/K_fold/validation/HealthyEnsetTree")
    else: os.mkdir("F:/eNsetDataSet/K_fold/validation/HealthyEnsetTree")

    if os.path.exists("F:/eNsetDataSet/K_fold/validation/BacterialWilt"):
                shutil.rmtree("F:/eNsetDataSet/K_fold/validation/BacterialWilt")
                os.mkdir("F:/eNsetDataSet/K_fold/validation/BacterialWilt")
    else: os.mkdir("F:/eNsetDataSet/K_fold/validation/BacterialWilt")

    if os.path.exists("F:/eNsetDataSet/K_fold/validation/LeafSpot"):
                shutil.rmtree("F:/eNsetDataSet/K_fold/validation/LeafSpot")
                os.mkdir("F:/eNsetDataSet/K_fold/validation/LeafSpot")
    else: os.mkdir("F:/eNsetDataSet/K_fold/validation/LeafSpot")

    if os.path.exists("F:/eNsetDataSet/K_fold/validation/RootMealybug"):
                shutil.rmtree("F:/eNsetDataSet/K_fold/validation/RootMealybug")
                os.mkdir("F:/eNsetDataSet/K_fold/validation/RootMealybug")
    else: os.mkdir("F:/eNsetDataSet/K_fold/validation/RootMealybug")

#To copy the training images to training sub/class directories
    for j in range(1,11):
        if j!=i:
            for file in os.listdir('F:/eNsetDataSet/K_fold/Healthy Enset Tree/fold_'
+ str(j)):
                shutil.copy(os.path.join('F:/eNsetDataSet/K_fold/Healthy Enset
Tree/fold_' + str(j), file),
os.path.join("F:/eNsetDataSet/K_fold/train/HealthyEnsetTree", file)) # performs
copy&overwrite


            for file in os.listdir('F:/eNsetDataSet/K_fold/Bacterial Wilt/fold_' +
str(j)):
                shutil.copy(os.path.join('F:/eNsetDataSet/K_fold/Bacterial
Wilt/fold_' + str(j), file),
os.path.join("F:/eNsetDataSet/K_fold/train/BacterialWilt", file))

            for file in os.listdir('F:/eNsetDataSet/K_fold/Leaf Spot/fold_' +
str(j)):
                shutil.copy(os.path.join('F:/eNsetDataSet/K_fold/Leaf Spot/fold_' +
str(j), file), os.path.join("F:/eNsetDataSet/K_fold/train/LeafSpot", file))
```

```
            for file in os.listdir('F:/eNsetDataSet/K_fold/RootMealybug/fold_' +
str(j)):
                shutil.copy(os.path.join('F:/eNsetDataSet/K_fold/RootMealybug/fold_'
+ str(j), file), os.path.join("F:/eNsetDataSet/K_fold/train/RootMealybug", file))
#Assign Path of each class/sub training directories

    train_HealthyEnsetTree_dir = 'F:/eNsetDataSet/K_fold/train/HealthyEnsetTree' #
directory with our training Healthy EnsetTree pictures
    train_BacterialWilt_dir = 'F:/eNsetDataSet/K_fold/train/BacterialWilt' #
directory with our training BacterialWilt pictures
    train_LeafSpot_dir = 'F:/eNsetDataSet/K_fold/train/LeafSpot' # directory with
our training LeafSpot pictures
    train_RootMealybug_dir = 'F:/eNsetDataSet/K_fold/train/RootMealybug' # directory
with our training RootMealybug pictures

#To copy the validation images to validation sub/class directories
    for file in os.listdir('F:/eNsetDataSet/K_fold/Healthy Enset Tree/fold_' +
str(i)):
                shutil.copy(os.path.join('F:/eNsetDataSet/K_fold/Healthy Enset
Tree/fold_' + str(i), file),
os.path.join("F:/eNsetDataSet/K_fold/validation/HealthyEnsetTree", file)) # performs
copy&overwrite

    for file in os.listdir('F:/eNsetDataSet/K_fold/Bacterial Wilt/fold_' + str(i)):
                shutil.copy(os.path.join('F:/eNsetDataSet/K_fold/Bacterial
Wilt/fold_' + str(i), file),
os.path.join("F:/eNsetDataSet/K_fold/validation/BacterialWilt", file)) # performs
copy&overwrite

    for file in os.listdir('F:/eNsetDataSet/K_fold/Leaf Spot/fold_' + str(i)):
                shutil.copy(os.path.join('F:/eNsetDataSet/K_fold/Leaf Spot/fold_' +
str(i), file), os.path.join("F:/eNsetDataSet/K_fold/validation/LeafSpot", file)) #
performs copy&overwrite

    for file in os.listdir('F:/eNsetDataSet/K_fold/RootMealybug/fold_' + str(i)):
                shutil.copy(os.path.join('F:/eNsetDataSet/K_fold/RootMealybug/fold_'
+ str(i), file), os.path.join("F:/eNsetDataSet/K_fold/validation/RootMealybug",
file)) # performs copy&overwrite
#Assign Path of each class/sub validation directories
```

```
    validation_HealthyEnsetTree_dir =
"F:/eNsetDataSet/K_fold/validation/HealthyEnsetTree"
    validation_BacterialWilt_dir = "F:/eNsetDataSet/K_fold/validation/BacterialWilt"
    validation_LeafSpot_dir = "F:/eNsetDataSet/K_fold/validation/LeafSpot"
    validation_RootMealybug_dir = "F:/eNsetDataSet/K_fold/validation/RootMealybug"

#To print the number of images in each training class directories
    num_HealthyEnsetTree_tr = len(os.listdir(train_HealthyEnsetTree_dir))
    num_BacterialWilt_tr = len(os.listdir(train_BacterialWilt_dir))
    num_LeafSpot_tr = len(os.listdir(train_LeafSpot_dir))
    num_RootMealybug_tr = len(os.listdir(train_RootMealybug_dir))

#To print the number of images in each validation class directories

    num_HealthyEnsetTree_val = len(os.listdir(validation_HealthyEnsetTree_dir))
    num_BacterialWilt_val = len(os.listdir(validation_BacterialWilt_dir))
    num_LeafSpot_val = len(os.listdir(validation_LeafSpot_dir))
    num_RootMealybug_val = len(os.listdir(validation_RootMealybug_dir))

#To print total number of images in training directories

    total_train = num_HealthyEnsetTree_tr + num_BacterialWilt_tr + num_LeafSpot_tr +
num_RootMealybug_tr

#To print total number of images in validation directories

    total_val = num_HealthyEnsetTree_val + num_BacterialWilt_val + num_LeafSpot_val
+ num_RootMealybug_val

    print('Total training Healthy EnsetTree images:', num_HealthyEnsetTree_tr)
    print('Total training Enset BacterialWilt images:', num_BacterialWilt_tr)
    print('Total training Enset LeafSpot images:', num_LeafSpot_tr)
    print('Total training Enset RootMealybug images:', num_RootMealybug_tr)
    print("---------------------------------")
    print('Total validation Healthy EnsetTree  images:', num_HealthyEnsetTree_val)
    print('Total validation Enset BacterialWilt images:', num_BacterialWilt_val)
    print('Total validation Enset LeafSpot images:', num_LeafSpot_val)
    print('Total validation Enset RootMealybug images:', num_RootMealybug_val)

    print("---------------------------------")
    print("Total training images:", total_train)
```

```
    print("Total validation images:", total_val)

#To configure Batch_size, epochs, image size, learning rate
    batch_size = 90
    epochs =30
    IMG_HEIGHT = 150
    IMG_WIDTH = 150
    lr=0.001
#To augment the training data using following parameters

image_gen_train = ImageDataGenerator(
                    rescale=1./255,
                    rotation_range=45,
                    width_shift_range=.15,
                    height_shift_range=.15,
                    horizontal_flip=True,
                    zoom_range=0.5
                    )
#To augument the validation data using folowing parametrs
image_gen_val = ImageDataGenerator(
                    rescale=1./255,
                    rotation_range=45,
                    width_shift_range=.15,
                    height_shift_range=.15,
                    horizontal_flip=True,
                    zoom_range=0.5
                    )

# Generator for normalizing our training image data data
    train_image_generator = ImageDataGenerator(rescale=1./255)
# Generator for normalizingour validation image data
    validation_image_generator = ImageDataGenerator(rescale=1./255)
#To count a print the total number of training images with their number of class
class
    train_data_gen =
train_image_generator.flow_from_directory(batch_size=batch_size,
                                                           directory=train_dir,
                                                           shuffle=True,
                                                           target_size=(IMG_HEIGHT,
IMG_WIDTH),
                                                           class_mode='categorical')
```

```
#To count a print the total number of training images with their number of class
    val_data_gen =
validation_image_generator.flow_from_directory(batch_size=batch_size,

directory=validation_dir,

target_size=(IMG_HEIGHT, IMG_WIDTH),

class_mode='categorical')
#train the model 10(k) times
    model = Sequential([
    Conv2D(16, 3,
           padding='same',
           activation='relu',
           input_shape=(IMG_HEIGHT, IMG_WIDTH ,3)),
    MaxPooling2D(pool_size=(2, 2)),
    Conv2D(32, 3,
           padding='same',
           activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(64, 3,
           padding='same',
           activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(32, 3,
           padding='same',
           activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(512,
          activation='relu'),
    Dropout(0.2),
    Dense(100,
          activation='relu'),
    Dropout(0.2),
    Dense(4, activation='softmax')

    ])
```

```
    model.summary()
    model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
# To check and save best model among 30 epochs that has maximum validation value
    from tensorflow.keras.callbacks import ModelCheckpoint
    filepath="F:/eNsetDataSet/K_fold/ensetNetModel_fold_" + str(i) + ".hdf5"
    checkpoint = ModelCheckpoint(filepath, monitor='val_acc', verbose=1,
save_best_only=True, mode='max')
    callbacks_list = [checkpoint]

    history = model.fit_generator(
        train_data_gen,
        callbacks=callbacks_list, verbose=1,
        steps_per_epoch=total_train // batch_size,
        epochs=epochs,
        validation_data=val_data_gen,
        validation_steps=total_val // batch_size
    )

#To plote the training accuracy and validation accuracy with their loss
    acc = history.history['acc']
    val_acc = history.history['val_acc']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs_range = range(epochs)

    plt.figure(figsize=(15, 5))
    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, acc, label='Training Accuracy')
    plt.plot(epochs_range, val_acc, label='Validation Accuracy')
    plt.legend(loc='lower right')
    plt.title('Training and Validation Accuracy')

    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, loss, label='Training Loss')
    plt.plot(epochs_range, val_loss, label='Validation Loss')
    plt.legend(loc='upper right')
    plt.title('Training and Validation Loss')
    plt.show()
```

```
    plt.savefig("F:/eNsetDataSet/K_fold/plot figures from fold_" + str(i)+".png")

#To plot confusion matrix and print class labels with its accuracy and loss
    saved_model =
tf.keras.models.load_model("F:/eNsetDataSet/K_fold/ensetNetModel_fold_" + str(i) +
".hdf5")
    image_gen_val = ImageDataGenerator(rescale=1./255)
    val_data_gen = image_gen_val.flow_from_directory(batch_size=batch_size,
                                                     directory=validation_dir,
                                                     shuffle=False,
                                                     target_size=(IMG_HEIGHT,
IMG_WIDTH),
                                                     class_mode='categorical')

    pred = saved_model.predict(val_data_gen)
    y_pred = np.rint(pred)
    max_index = np.argmax(y_pred, axis=1)
    print("Predicted Classes ")
    print(max_index)
    print("\n")
    y_true = val_data_gen.classes
    print("True Classes ")
    print(y_true)
    print("\n")
    cnf_matrix = confusion_matrix(y_true, max_index)
    print("Confusion Matrix")
    print(cnf_matrix)
    print("\n")
    print("precision, recall, fscore and support for each class")
    print(precision_recall_fscore_support(y_true, max_index, average=None))
    print("\n")

    classes=['Bacterial Wilt', 'Healthy Enset', 'Leaf Spot', 'Root Mealy Bug']
    print("Plot Confusion Matrix")
    plt.figure()
    plot_confusion_matrix(cnf_matrix, target_names=classes,
                      title='Confusi on matrix')
    plt.savefig('Confussion Matrix of fold_'+ str(i) +'.png')
    print("\n")

    filePath = "F:/eNsetDataSet/K_fold/validation/BacterialWilt"
```

```
    if os.path.exists(filePath):
        os.remove(filePath)
    else:print("OK GO FOR->>>>>>NEXT FOLD")
```

```
#To test/predict class of single image
saved_model =
tf.keras.models.load_model("F:/eNsetDataSet/K_fold/ensetNetModel_fold_" + str(i) +
".hdf5")
import numpy as np
from keras.preprocessing import image
import tensorflow as tf
img_width, img_height = 150, 150
img = image.load_img('F:/eNsetDataSet/Validation/Bacterial Wilt/B_W (2831).png ',
target_size = (img_width, img_height))
img = image.img_to_array(img)
img = np.expand_dims(img, axis = 0)
pred = saved_model.predict(img)
print(pred)
max_index = np.where(pred[0] == np.amax(pred[0]))
if(max_index[0][0]==0):
    print("Bacterial Wilt")
elif(max_index[0][0]==1):
    print("Healthy Enset")
elif(max_index[0][0]==2):
    print("Leaf Spot")
elif(max_index[0][0]==3):
    print("Root Mealy Bug")
```