RESEARCH ARTICLE

Enset (*Enset ventricosum*) plant disease and pests identification using image processing and deep convolutional neural network

Tsegaye Yibgeta, Million Meshesha, Muktar Bedaso*

Yibgeta T, Meshesha M, Bedaso M. *Enset* (*Enset ventricosum*) plant disease and pests identification using image processing and deep convolutional neural network. AGBIR.2024;40(3):1170-1181.

Enset is a monocarpic perennial crop that belongs to the Schistaminae order and the Musaceae family. *Enset* is a significant food security crop in Southern Ethiopia, where almost 20 million people depend on it for survival. Plant leaf diseases and damaging pests are foremost challenges in *Enset* production. This study looks into the use of deep learning to detect bacterial wilt disease and *Enset* mealy bug pest, where data is obtained in small amounts and collected under minimally controlled conditions. We employed data augmentation to get over the limits of the dataset size. The proposed approach is divided into four stages. The initial part entails gathering healthy and diseased *Enset* images with the support of agricultural experts, from various farms and research institutes. Then image processing tasks, resizing and segmentation are applied on the collected dataset in order to get an enhanced (simpler) image and to extract region of interest from the dataset images. Finally, using the collected dataset, the created model is trained and evaluated, and it is compared to the state of the art pre-trained convolutional neural network models: AlexNet, ResNet-50, Inception v3, DenseNet-201, VGG16 and EfficientNetB3. The proposed approach is implemented using Google collaboratory or "colab" for short. To detect and classify *Enset* diseases, the model has an accuracy of 99.68% for training and 98.87% for testing.
Keywords: *Enset* bacterial wilt; *Enset* mealybug; Convolutional neural network; Deep learning; Image processing

INTRODUCTION

Enset ventricosum (false banana) has high significance in the day to day life of more than 20 million Ethiopians as a food source, fiber, animal forage, construction materials, and medicines [1]. The production of Enset is hampered by biotic and abiotic agents such as insect pests, weeds, wild animals and soil nutrient depletion. Enset diseases are caused by several bacteria, fungi, viruses, and nematodes. The Enset bacterial wilt and mealy bug are the major constraints in the production of Enset [2].

The field of digital image processing refers to the use of a digital computer to process digital images. It's important to remember that a digital image is made up of a finite number of pieces, each of which has its own location and value. The ultimate objective of computer vision is to have computers mimic human vision, which includes learning and being able to draw inferences and conduct actions based on visual inputs. This field is a subset of Artificial Intelligence (AI), with the goal of simulating human intelligence. The application of computer vision to carry out duty for value examination, sorting, and automatic processes is growing in the agricultural business [3]. Image analysis (also known as image understanding) is a branch of computer vision that sits between image processing and computer vision [4]. CNN is an architecture that is being used for different computer vision tasks. Each layer of CNN learns to extract features from input images that will be used to classify the images in the end. The benefit of using a deep learning approach is that it reduces the number of image processing steps required when compared to typical machine learning methods [5]. The aim of this study is to detect and classify *Enset* diseases and pests using image processing techniques and a deep convolutional neural networks.

Statement of the problem

The bacterial wilt and mealy bug diseases of *Enset* are widespread in the country's major *Enset* growing regions causing losses of up to 100% destruction of farm fields in extreme cases. Currently, BWE has infected over 80% of *Enset* farms, and no *Enset* clone has been identified that is totally resistant to bacterial wilt. Mealy bug is also the major disease of *Enset*, about 37.6% of *Enset* plant is infected by these pests [6]. Because the majority of Ethiopian farmers are uneducated and do not receive accurate and thorough information regarding *Enset* crop diseases, they have the wrong concept about the symptoms and causes of the diseases, so they require expert advice. On the other hand, it is impossible for crop pathologists to visit every farm, and because even crop pathologists rely on manual eye inspection, the manual prediction approach is ineffective, time consuming, and labor intensive. The other constraint is mealybugs, which are white colored pests that attack the top root parts of *Enset*. Mealy bug infested *Enset* plants exhibit retarded growth, loss of vigor, dried lateral leaves but green central shoot and eventually plant death. By looking at the top root parts of the plant we can identify them by their whitish color.

Other minor but commonly reported diseases of *Enset* besides bacterial wilt and mealybug are black sigatoka and Corm rot [7]. To the best of our knowledge, only two studies on the detection of bacterial wilt of *Enset* have been undertaken, and no studies on the detection of mealybug have been conducted. Hence, it is crucial to design a model that can automatically identify and classify the two major diseases of *Enset*. Identifying plant diseases using computer vision has been done for more than a decade and it has produced promising results. But only few researches have been conducted in *Enset* disease detection and classification. It is therefore the aim of this study to apply image processing and deep learning. Finally, this study attempts to investigate and answer the following research questions.

- What are suitable methods and algorithms to apply to prepare a quality dataset for experimentation?
- Which deep learning algorithm is suitable for detecting and classifying *Enset* plant diseases and pests?
- To what extent the proposed model works in detecting *Enset* diseases?

The objective of the study: The main objective of this study is to design and develop an automatic *Enset* diseases and pests identification model by using image processing and deep CNN.

Specific objectives:

- To collect datasets of *Enset* according to the three classes we are going to classify.

Department of Information Science, Jimma Institute of Technology, Jimma, Ethiopia

Correspondence: *Muktar Bedaso, Department of Information Science, Jimma Institute of Technology, Jimma, Ethiopia; E-mail: isaakadiir.3@gmail.com*

Received: *27-Mar-2023, Manuscript No. AGBIR-23-93092;* ***Editor assigned:*** *29-Mar-2023, PreQC No. AGBIR-23-93092 (PQ);* ***Reviewed:*** *12-Apr-2023, QC No. AGBIR-23-93092;* ***Revised:*** *24-April-2024, Manuscript No. AGBIR-23-93092 (R);* ***Published:*** *01-May-2024, DOI: 10.37532/0970-1907.24.40.1170-1181*

- To select and construct algorithms that are best suited for colour image segmentation.
- To make comparison between different machine learning algorithms.
- To construct a CNN model for automating *Enset* disease detection.
- Test and evaluate the proposed model with an unseen dataset.

MATERIALS AND METHODS

Research design

In this study, the experimental research design is used for data preparation, model building, training the model, and testing the proposed model.

Data acquisition

Image data were collected from Wolkite university incubation center, Gurage zone 'Cheha', and 'Eza' district local *Enset* farms. Domain experts are involved during the collection, experimentation and evaluation of this proposed model. The collected sample images of *Enset* for this study were taken with a Sonny VIXIA HF M50 HD camera, which features a 24 MP camera with a default resolution of 4248 × 5664 pixels, giving a total of (24,060,672 pixels or 24 MP). Three hundred twenty eight (328) for each healthy and mealybug classes, and 331 images are collected for the bacterial wilt class, nine hundred eighty seven (987) images are collected in total. It is divided using 80/20 data split for training and testing respectively. Allocating 80% of the dataset for training is close to optimal for reasonable sized datasets (greater than 100 images) [8]. Since the amount of data collected is limited, data augmentation is applied during training to artificially increase the amount of data (Figure 1).

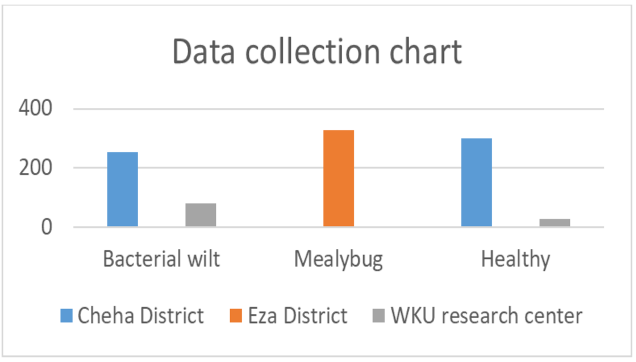


Figure 1: Data collection chart.

Data augmentation

The augmentation techniques used in this thesis are geometric transformations and color space transformations (Table 1).

TABLE 1
Augmentation techniques used

Parameters	Values
Rotation range	15
Rescale	1/255
Shear range	0.1
Zoom range	0.2
Horizontal flip	True
Width shift range	0.1
Height shift range	0.1

Image processing

For this study, we applied image resizing and image segmentation only. Image resizing is done because the CNN algorithm accepts a defined image size format so all of our images should be of the same size, and also it is easier for our hardware resources to work with low image sizes. Image segmentation is done to extract the region of interest from the images.

Implementation tools

Experiments are based on a Google colab and anaconda prototype built with Keras (Tensor flow as a backend). Google Colab offers a free Intel(R) Xeon(R) CPU 2.30 GHz with 2 CPUs, 8 GB of RAM, 78 GB of storage, and 15 GB of GPU, so why not take advantage of it. With a batch size of 50, the model is trained for 50 epochs.

System performance and user acceptance testing

The model is evaluated by running validation dataset on the developed model. The model was then tested by new test dataset and evaluated the prediction accuracy using confusion matrix. In addition, we compared our results to state of the art models like AlexNet, ResNet-50, VGG16, Inception v3, DenseNet-201 and EfficientNetB3, which have been widely employed in earlier research. Models that have won ILSVRC (Image net Large scale visual recognition challenges) in previous years are consideredstate of the art. User acceptance is performed by Gurage zone agricultural institute domain experts (Figure 2).

Proposed model architecture

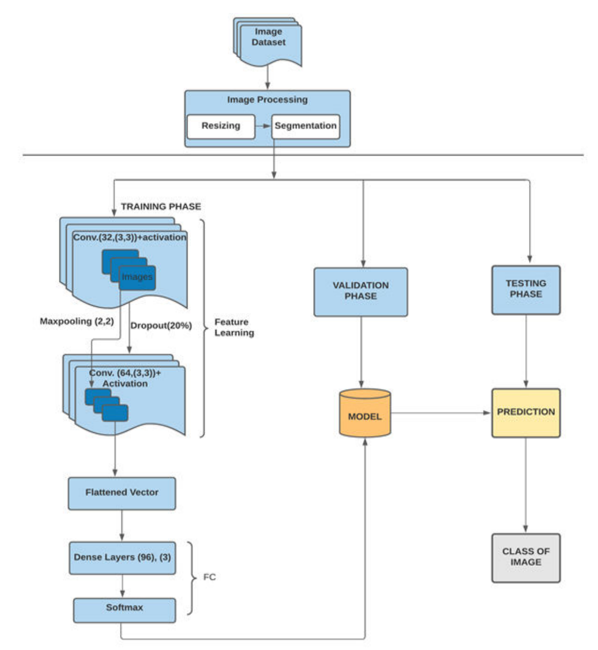


Figure 2: Proposed model architecture.

Image data preprocessing

The dataset collected has a dimension of 5664 × 4248 pixels, so it takes too much time and resource to process. As a result, for efficient processing, we scale the photos to a uniform 256*256 dimension. OpenCV is used for image resizing and conversion into Numpy array.

Segmentation

Our dataset contains shadows and noises so it is impossible segment the ROI part of the region with extreme color detection. In order to segment the leaves from the images, otsu thresholding, background marker and color segmentation were used together (Figure 3).

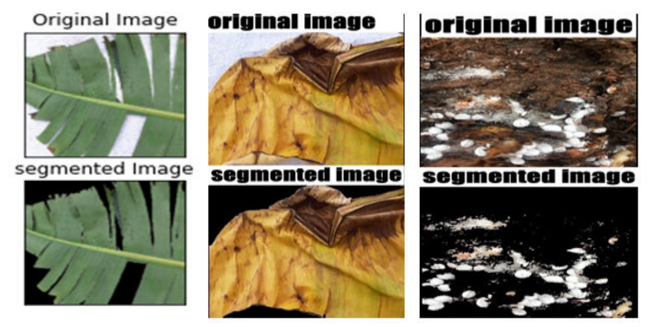
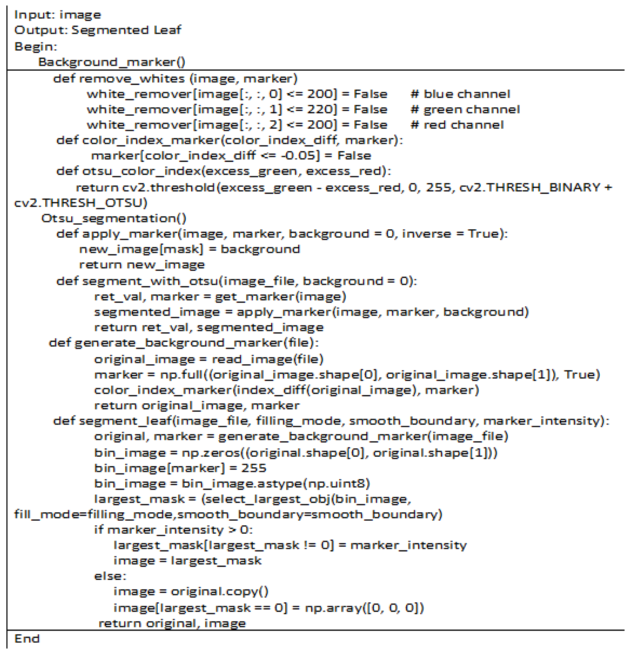


Figure 3: Segmentation of healthy, bacterial wilt and mealy bug plant.

The non-ROI part of the image is given a value of zero to accomplish the segmentation. Because the pixel values outside the region of interest are similar, they are (set to zero), which allows the CNN algorithm to focus on learning the distinguishing traits (Figure 4).



```
Input: image
Output: Segmented Leaf
Begin:
    Background_marker()
      def remove_whites (image, marker)
           white_remover[image[:, :, 0] <= 200] = False      # blue channel
           white_remover[image[:, :, 1] <= 220] = False      # green channel
           white_remover[image[:, :, 2] <= 200] = False      # red channel
       def color_index_marker(color_index_diff, marker):
            marker[color_index_diff <= -0.05] = False
       def otsu_color_index(excess_green, excess_red):
          return cv2.threshold(excess_green - excess_red, 0, 255, cv2.THRESH_BINARY +
cv2.THRESH_OTSU)
     Otsu_segmentation()
       def apply_marker(image, marker, background = 0, inverse = True):
          new_image[mask] = background
          return new_image
       def segment_with_otsu(image_file, background = 0):
            ret_val, marker = get_marker(image)
            segmented_image = apply_marker(image, marker, background)
            return ret_val, segmented_image
      def generate_background_marker(file):
            original_image = read_image(file)
            marker = np.full((original_image.shape[0], original_image.shape[1]), True)
            color_index_marker(index_diff(original_image), marker)
            return original_image, marker
      def segment_leaf(image_file, filling_mode, smooth_boundary, marker_intensity):
            original, marker = generate_background_marker(image_file)
            bin_image = np.zeros((original.shape[0], original.shape[1]))
            bin_image[marker] = 255
            bin_image = bin_image.astype(np.uint8)
            largest_mask = (select_largest_obj(bin_image,
fill_mode=filling_mode,smooth_boundary=smooth_boundary)
            if marker_intensity > 0:
               largest_mask[largest_mask != 0] = marker_intensity
               image = largest_mask
            else:
               image = original.copy()
               image[largest_mask == 0] = np.array([0, 0, 0])
             return original, image
End
```

Figure 4: Segmentation algorithm for segmenting ROI from image.

Feature extraction

In this study, CNN is used for feature extraction. During the training phase, CNN includes feature extraction and weight computation. A convolution operator, which is beneficial for solving complex processes, is used to give these networks their names. The given input data is first sent to a feature extraction network, and the retrieved features are then sent to a classifier network.

Classification

Based on the features gathered, classifiers are used to identify and categorize the diseases that affect plant leaves. K-Nearest Neighbors (K-NN), Support Vector Machines (SVM), Convolutional Neural Networks (CNN), and Artificial Neural Networks (ANN) are some of the classifiers that can be used to detect diseases in plants. Since CNNs have achieved significant achievements in the field of computer vision, deep Convolutional Neural Network (CNN) models are used in this study to identify diseases in *Enset* plant.

Training phase

Feature learning for training and constructing a model

Convolution layers: The input layer, hidden layers, and output layer make up a convolutional neural network. The hidden layers of a convolutional neural network include layers that perform convolution. This usually comprises a layer that does a dot product of the convolution kernel with the input matrix of the layer (Table 2).

TABLE 2

Summary of our model

Layer (type) param#	Output shape
conv2d_6 (Conv2D) 896	256, 256, 32

max_pooling2d_4 (MaxPooling2)	128, 128, 32
dropout_4 (Dropout)	128, 128, 32
conv2d_7 (Conv2D) 18496	128, 128, 64
max_pooling2d_5 (MaxPooling2)	64, 64, 64
dropout_5 (Dropout)	64, 64, 64
flatten_1 (Flatten)	262144
dense_2 (Dense) 25165920	96
dense_3 (Dense) 291	3
Total params: 25,185,603	
Trainable params: 25,185,603	
Non-trainable params: 0	

Pooling layer

The authors used two pooling layers in our model and for both of the pooling layers, we used a 2 × 2 pool size with the stride of 2 × 2. For this study, the authors have used max pooling because we are interested in the bright pixels of the images. For example, the following Figure 5 shows max pooling being applied on part of an image (Figure 5).

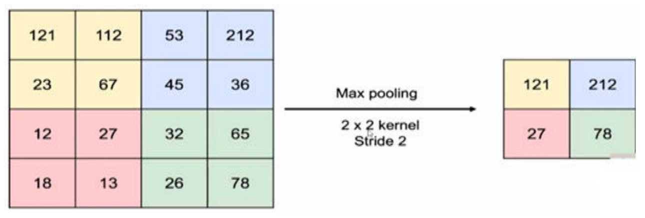


Figure 5: Maxpooling.

Activation function

The linearity activation function Rectified Linear Unit (ReLU) was employed.

Dropout

After each Maxpooling layer and also after the flatten layer, we utilized a dropout layer with a dropping probability of 0.25 and 0.4.

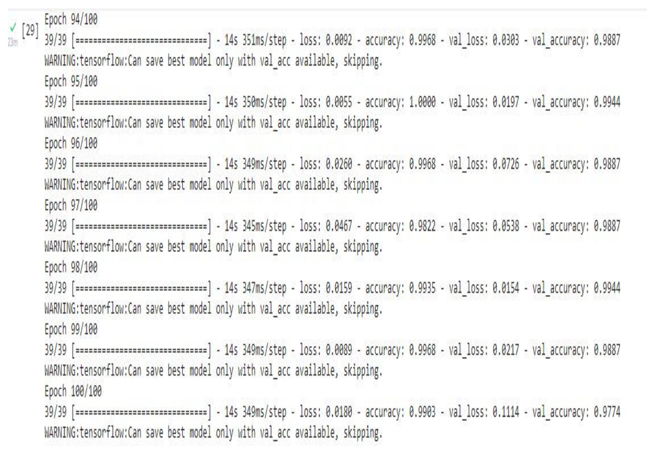
Classification using the model

Fully Connected layer (FC): To compute the final output probabilities for each class, the authors used only one fully connected layer. As there are three classes, the first layer has 96 neurons and the final (output) layer, which is the model's output layer, has three neurons. The output of the flattening layer, which is a vector value, is accepted by the first FC layer.

RESULTS

Experimenting CNN algorithm by applying image segmentation

Experimental result of CNN after image segmentation registers a training accuracy of 99.68% and a validation accuracy of 98.87%. It has a training loss of 0.92% and a validation loss of 3.03%. This classification accuracy is obtained when the model is trained by applying segmentation, data augmentation, maxpooling, stride, dropouts, and Adam optimizer (Figure 6).



```
[29]
23m
Epoch 94/100
39/39 [==============================] - 14s 351ms/step - loss: 0.0092 - accuracy: 0.9968 - val_loss: 0.0303 - val_accuracy: 0.9887
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 95/100
39/39 [==============================] - 14s 350ms/step - loss: 0.0055 - accuracy: 1.0000 - val_loss: 0.0197 - val_accuracy: 0.9944
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 96/100
39/39 [==============================] - 14s 349ms/step - loss: 0.0260 - accuracy: 0.9968 - val_loss: 0.0726 - val_accuracy: 0.9887
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 97/100
39/39 [==============================] - 14s 345ms/step - loss: 0.0467 - accuracy: 0.9822 - val_loss: 0.0538 - val_accuracy: 0.9887
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 98/100
39/39 [==============================] - 14s 347ms/step - loss: 0.0159 - accuracy: 0.9935 - val_loss: 0.0154 - val_accuracy: 0.9944
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 99/100
39/39 [==============================] - 14s 349ms/step - loss: 0.0089 - accuracy: 0.9968 - val_loss: 0.0217 - val_accuracy: 0.9887
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
Epoch 100/100
39/39 [==============================] - 14s 349ms/step - loss: 0.0180 - accuracy: 0.9903 - val_loss: 0.1114 - val_accuracy: 0.9774
WARNING:tensorflow:Can save best model only with val_acc available, skipping.
```

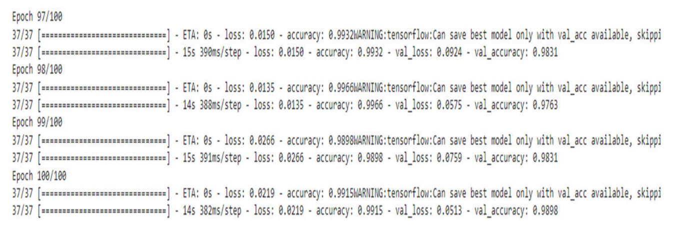
Figure 6: Classification accuracy and loss of our model.

The authors tested our model's accuracy in predicting this dataset. On all of our test datasets, our model predicted properly with an accuracy of 99.97%and a loss of only 0.01%. There are 34 mealybug class images, 34 bacterial wilt class images, and 32 healthy class images in our test dataset.

The model successfully predicted all of the mealybug images, all of the bacterial wilt images from 34 test images, and all of the healthy images from 32 images.

Changing training and testing dataset ratio

The outcome of the proposed model with a split ratio of 70/30% for the training and testing dataset is examined. The *Enset* disease detection model registered a training accuracy of 99.15% and a validation accuracy of 98.98%. It has a training loss of 2.19% and a validation loss of 5.13%. This classification accuracy is obtained when the model is trained by applying data augmentation, maxpooling, stride, and dropouts (Figure 7).



```
Epoch 97/100
37/37 [==============================] - ETA: 0s - loss: 0.0150 - accuracy: 0.9932WARNING:tensorflow:Can save best model only with val_acc available, skippi
37/37 [==============================] - 15s 390ms/step - loss: 0.0150 - accuracy: 0.9932 - val_loss: 0.0924 - val_accuracy: 0.9831
Epoch 98/100
37/37 [==============================] - ETA: 0s - loss: 0.0135 - accuracy: 0.9966WARNING:tensorflow:Can save best model only with val_acc available, skippi
37/37 [==============================] - 14s 388ms/step - loss: 0.0135 - accuracy: 0.9966 - val_loss: 0.0575 - val_accuracy: 0.9763
Epoch 99/100
37/37 [==============================] - ETA: 0s - loss: 0.0266 - accuracy: 0.9898WARNING:tensorflow:Can save best model only with val_acc available, skippi
37/37 [==============================] - 15s 391ms/step - loss: 0.0266 - accuracy: 0.9898 - val_loss: 0.0759 - val_accuracy: 0.9831
Epoch 100/100
37/37 [==============================] - ETA: 0s - loss: 0.0219 - accuracy: 0.9915WARNING:tensorflow:Can save best model only with val_acc available, skippi
37/37 [==============================] - 14s 382ms/step - loss: 0.0219 - accuracy: 0.9915 - val_loss: 0.0513 - val_accuracy: 0.9898
```

Figure 7: Training and validation accuracy and loss of 70/30 split.

Training and validation accuracy rises while training and validation loss reduces, as illustrated in the training loss and accuracy curve in Figures 8 and 9.

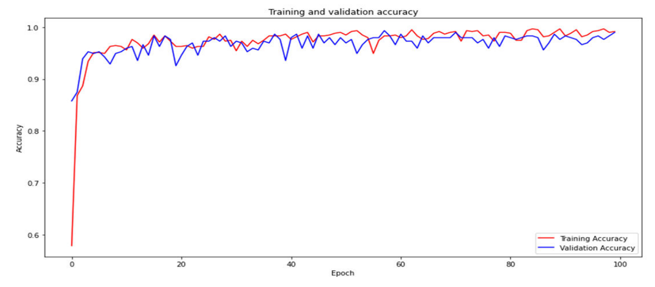


Figure 8: Training and validation accuracy of 70/30 split.

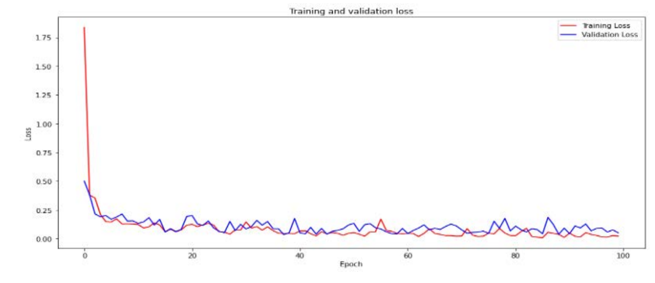


Figure 9: Training and validation loss of 70/30 split.

As shown in the confusion matrix below, the model successfully predicted correctly on all of our test datasets with an accuracy of 99.95% and a loss of only 0.7% (Figure 10).

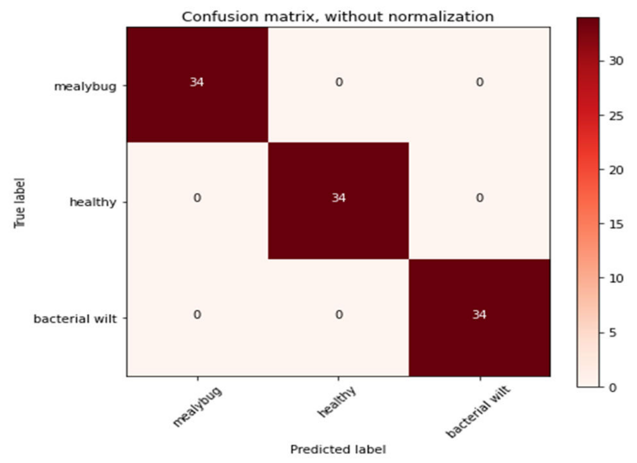
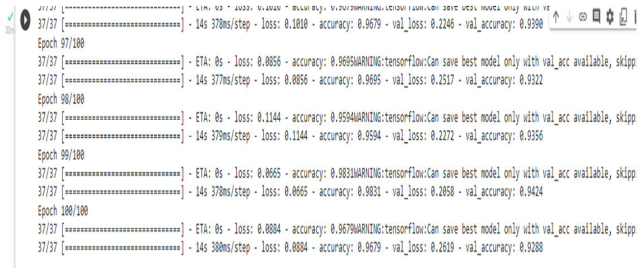


Figure 10: Confusion matrix of 70/30 split.

Changing optimizers

Experimental results of our model by applying AdaGrad optimizer: In this experiment, we tested our model by changing the optimizer from Adam to AdaGrad. The model registered a training accuracy of 96.79% and a validation accuracy of 92.88%. It has a training loss of 8.84% and a validation loss of 26.19% (Figure 11).



```
37/37 [==============================] - ETA: 0s - loss: 0.1010 - accuracy: 0.9679WARNING:tensorflow:Can save best model only with va
37/37 [==============================] - 14s 378ms/step - loss: 0.1010 - accuracy: 0.9679 - val_loss: 0.2246 - val_accuracy: 0.9390
Epoch 97/100
37/37 [==============================] - ETA: 0s - loss: 0.0856 - accuracy: 0.9695WARNING:tensorflow:Can save best model only with val_acc available, skipp
37/37 [==============================] - 14s 377ms/step - loss: 0.0856 - accuracy: 0.9695 - val_loss: 0.2517 - val_accuracy: 0.9322
Epoch 98/100
37/37 [==============================] - ETA: 0s - loss: 0.1144 - accuracy: 0.9594WARNING:tensorflow:Can save best model only with val_acc available, skipp
37/37 [==============================] - 14s 379ms/step - loss: 0.1144 - accuracy: 0.9594 - val_loss: 0.2272 - val_accuracy: 0.9356
Epoch 99/100
37/37 [==============================] - ETA: 0s - loss: 0.0665 - accuracy: 0.9831WARNING:tensorflow:Can save best model only with val_acc available, skipp
37/37 [==============================] - 14s 378ms/step - loss: 0.0665 - accuracy: 0.9831 - val_loss: 0.2058 - val_accuracy: 0.9424
Epoch 100/100
37/37 [==============================] - ETA: 0s - loss: 0.0884 - accuracy: 0.9679WARNING:tensorflow:Can save best model only with val_acc available, skipp
37/37 [==============================] - 14s 380ms/step - loss: 0.0884 - accuracy: 0.9679 - val_loss: 0.2619 - val_accuracy: 0.9288
```

Figure 11: Training and validation accuracy and loss by applying AdaGrad optimizer.

Below Figure 12 indicate the training and validation learning curves. The learning curves indicate that our model performs well by using AdaGrad optimizer too (Figures 12-14).

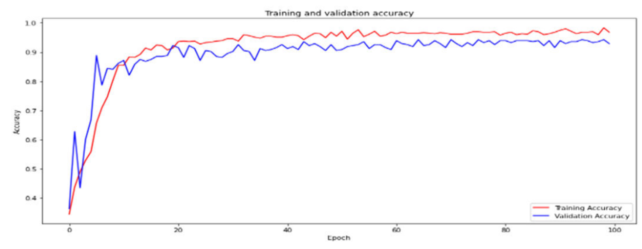


Figure 12: Training and validation accuracy graph of our model by applying AdaGrad optimizer.

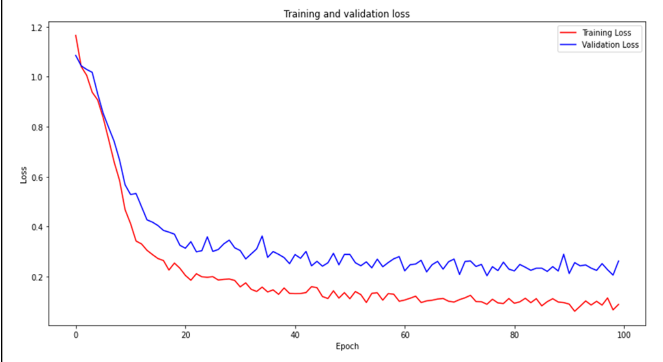


Figure 13: Training and validation loss graph of our model by applying AdaGrad optimizer.

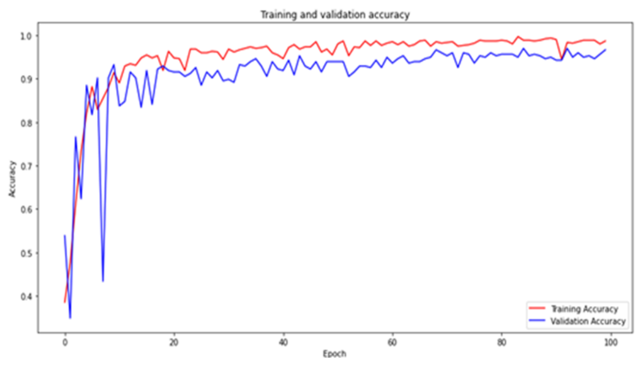


Figure 16: Training and validation accuracy graph with SGD.

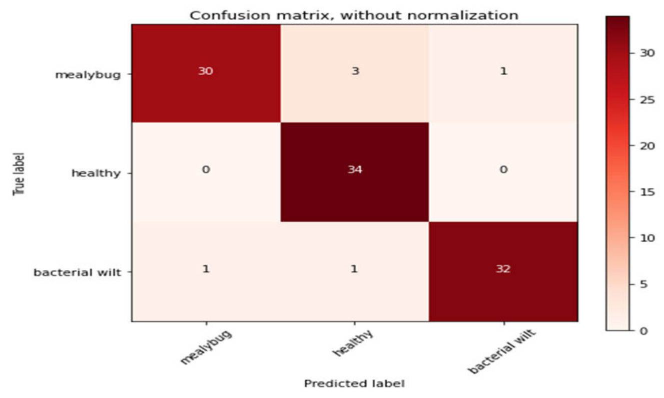


Figure 14: Confusion matrix of our model after applying AdaGrad optimizer.

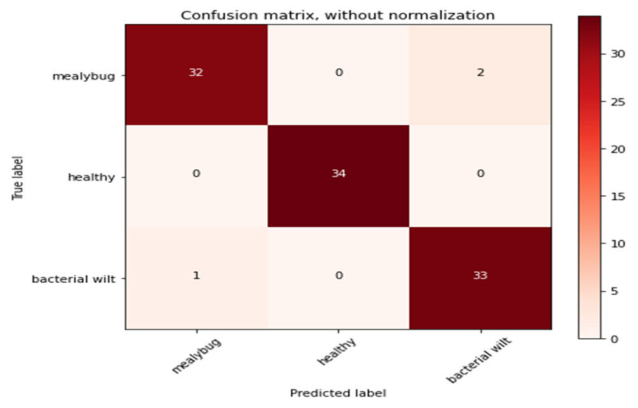
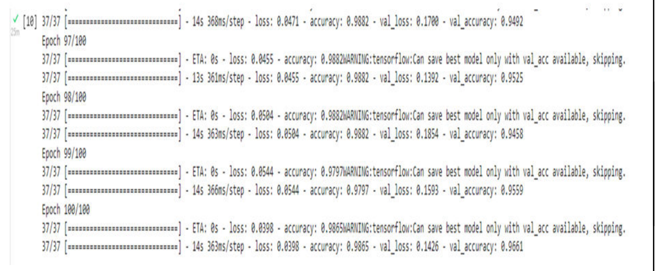


Figure 17: Confusion matrix after applying SGD.

Experimental result of our model by applying Stochastic Gradient Descent (SGD)

In this experiment, we tested our model by changing the optimizer from SGD. The model registered a training accuracy of 98.65% and a validation accuracy of 96.61%. It has a training loss of 3.98% and a validation loss of 14.26%. The model took 25 minutes to train, with each epoch requiring an average of 15 seconds (Figure 15).



```
✓ [10] 37/37 [==============================] - 14s 368ms/step - loss: 0.0471 - accuracy: 0.9882 - val_loss: 0.1700 - val_accuracy: 0.9492
25m
       Epoch 97/100
       37/37 [==============================] - ETA: 0s - loss: 0.0455 - accuracy: 0.9882WARNING:tensorflow:Can save best model only with val_acc available, skipping.
       37/37 [==============================] - 13s 361ms/step - loss: 0.0455 - accuracy: 0.9882 - val_loss: 0.1392 - val_accuracy: 0.9525
       Epoch 98/100
       37/37 [==============================] - ETA: 0s - loss: 0.0504 - accuracy: 0.9882WARNING:tensorflow:Can save best model only with val_acc available, skipping.
       37/37 [==============================] - 14s 363ms/step - loss: 0.0504 - accuracy: 0.9882 - val_loss: 0.1854 - val_accuracy: 0.9458
       Epoch 99/100
       37/37 [==============================] - ETA: 0s - loss: 0.0544 - accuracy: 0.9797WARNING:tensorflow:Can save best model only with val_acc available, skipping.
       37/37 [==============================] - 14s 366ms/step - loss: 0.0544 - accuracy: 0.9797 - val_loss: 0.1593 - val_accuracy: 0.9559
       Epoch 100/100
       37/37 [==============================] - ETA: 0s - loss: 0.0398 - accuracy: 0.9865WARNING:tensorflow:Can save best model only with val_acc available, skipping.
       37/37 [==============================] - 14s 363ms/step - loss: 0.0398 - accuracy: 0.9865 - val_loss: 0.1426 - val_accuracy: 0.9661
```

Figure 15: Training and validation accuracy and loss with SGD.

The graphs below indicate the training and validation learning curves. The learning curves indicate that our model performs well by using SGD optimizer too. The graphs indicate no sign of over fitting and under fitting (Figures 16 and 17).

Comparing the proposed model with the state of the art models

Comparison with AlexNet model: A detailed description of AlexNet model is explained in chapter two. The performance (accuracy and loss value) of AlexNet-50 model is shown in Figure 18. AlexNet-50 achieves 97.34%training and 94.30% validation accuracy. It has a training loss of 8.54% and a validation loss of 21.17%. It takes 24 minutes to train the model, taking 14.4 seconds per epoch on average. The size of the model is also very huge (321 MB). Training and validation accuracy rises while training and validation loss reduces, as illustrated in the training loss and accuracy curve in Figures 18 and 19.

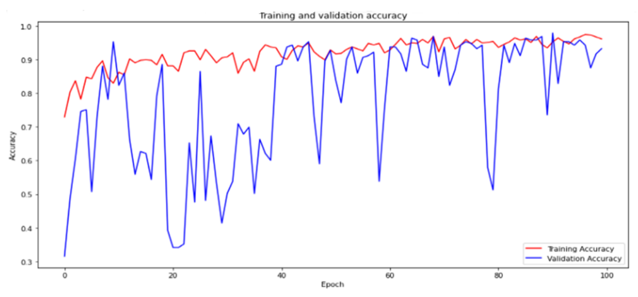


Figure 18: AlexNet training and validation accuracy graph.

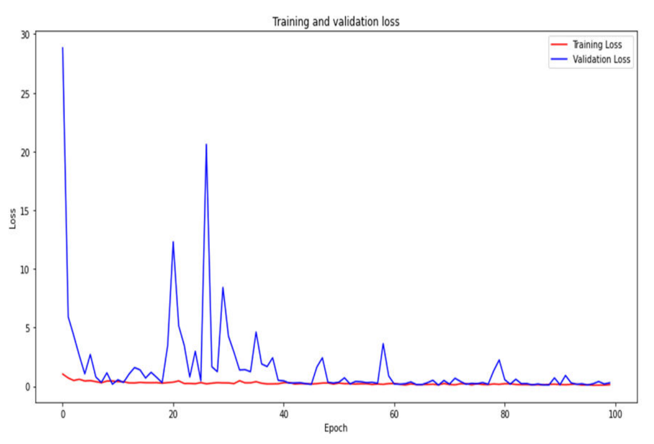
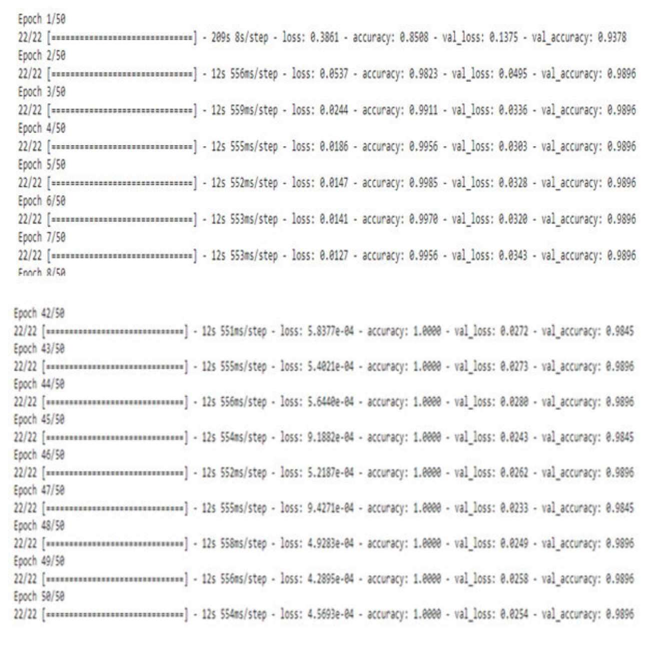


Figure 19: AlexNet training and validation loss graph.

Comparison with VGG16

The performance (accuracy and loss value) of the VGG16 model is shown in Figures 20-22. VGG16 achieves 99.89% training and 98.96 validation accuracy on our data, as shown in Figure 20.



```
Epoch 1/50
22/22 [==============================] - 209s 8s/step - loss: 0.3861 - accuracy: 0.8508 - val_loss: 0.1375 - val_accuracy: 0.9378
Epoch 2/50
22/22 [==============================] - 12s 556ms/step - loss: 0.0537 - accuracy: 0.9823 - val_loss: 0.0495 - val_accuracy: 0.9896
Epoch 3/50
22/22 [==============================] - 12s 559ms/step - loss: 0.0244 - accuracy: 0.9911 - val_loss: 0.0336 - val_accuracy: 0.9896
Epoch 4/50
22/22 [==============================] - 12s 555ms/step - loss: 0.0186 - accuracy: 0.9956 - val_loss: 0.0303 - val_accuracy: 0.9896
Epoch 5/50
22/22 [==============================] - 12s 552ms/step - loss: 0.0147 - accuracy: 0.9985 - val_loss: 0.0328 - val_accuracy: 0.9896
Epoch 6/50
22/22 [==============================] - 12s 553ms/step - loss: 0.0141 - accuracy: 0.9970 - val_loss: 0.0320 - val_accuracy: 0.9896
Epoch 7/50
22/22 [==============================] - 12s 553ms/step - loss: 0.0127 - accuracy: 0.9956 - val_loss: 0.0343 - val_accuracy: 0.9896
Epoch 8/50

Epoch 42/50
22/22 [==============================] - 12s 551ms/step - loss: 5.8377e-04 - accuracy: 1.0000 - val_loss: 0.0272 - val_accuracy: 0.9845
Epoch 43/50
22/22 [==============================] - 12s 555ms/step - loss: 5.4021e-04 - accuracy: 1.0000 - val_loss: 0.0273 - val_accuracy: 0.9896
Epoch 44/50
22/22 [==============================] - 12s 556ms/step - loss: 5.6440e-04 - accuracy: 1.0000 - val_loss: 0.0280 - val_accuracy: 0.9896
Epoch 45/50
22/22 [==============================] - 12s 554ms/step - loss: 9.1882e-04 - accuracy: 1.0000 - val_loss: 0.0243 - val_accuracy: 0.9845
Epoch 46/50
22/22 [==============================] - 12s 552ms/step - loss: 5.2187e-04 - accuracy: 1.0000 - val_loss: 0.0262 - val_accuracy: 0.9896
Epoch 47/50
22/22 [==============================] - 12s 555ms/step - loss: 9.4271e-04 - accuracy: 1.0000 - val_loss: 0.0233 - val_accuracy: 0.9845
Epoch 48/50
22/22 [==============================] - 12s 558ms/step - loss: 4.9283e-04 - accuracy: 1.0000 - val_loss: 0.0249 - val_accuracy: 0.9896
Epoch 49/50
22/22 [==============================] - 12s 556ms/step - loss: 4.2895e-04 - accuracy: 1.0000 - val_loss: 0.0258 - val_accuracy: 0.9896
Epoch 50/50
22/22 [==============================] - 12s 554ms/step - loss: 4.5693e-04 - accuracy: 1.0000 - val_loss: 0.0254 - val_accuracy: 0.9896
```

Figure 20: VGG16 training and validation accuracy and loss.

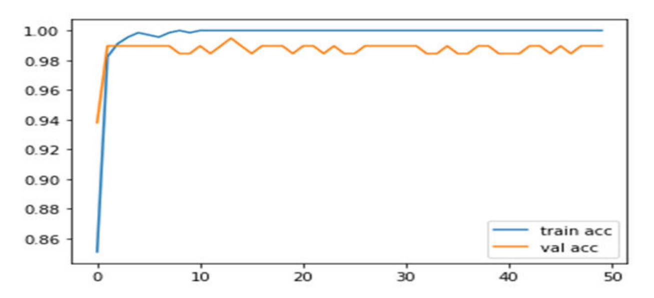


Figure 21: VGG16 training and validation accuracy graph.

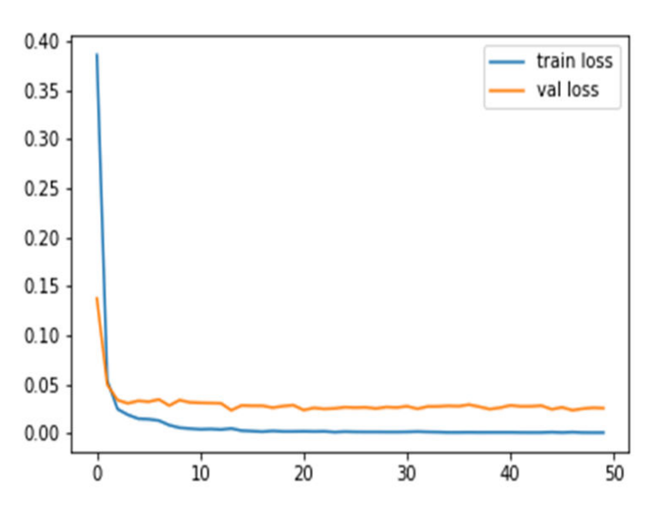
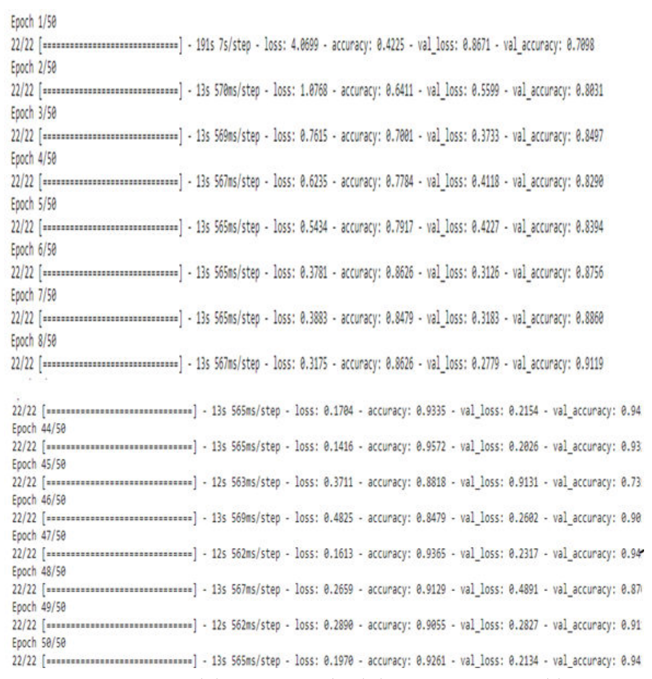


Figure 22: VGG16 training and validation loss graph.

As clearly depicted in Figure 22 the confusion matrix shows the following results. There are 34 mealybug class images, 34 bacterial wilt class images, and 32 healthy class images in our test dataset. VGG16 predicted all of the test dataset images correctly with an accuracy of 99.98% and a loss of only 0.36%.

Comparison with ResNet-50 model

The performance (accuracy and loss value) of ResNet model is shown in Figures 23-25.



```
Epoch 1/50
22/22 [==============================] - 191s 7s/step - loss: 4.0699 - accuracy: 0.4225 - val_loss: 0.8671 - val_accuracy: 0.7098
Epoch 2/50
22/22 [==============================] - 13s 570ms/step - loss: 1.0768 - accuracy: 0.6411 - val_loss: 0.5599 - val_accuracy: 0.8031
Epoch 3/50
22/22 [==============================] - 13s 569ms/step - loss: 0.7615 - accuracy: 0.7001 - val_loss: 0.3733 - val_accuracy: 0.8497
Epoch 4/50
22/22 [==============================] - 13s 567ms/step - loss: 0.6235 - accuracy: 0.7784 - val_loss: 0.4118 - val_accuracy: 0.8290
Epoch 5/50
22/22 [==============================] - 13s 565ms/step - loss: 0.5434 - accuracy: 0.7917 - val_loss: 0.4227 - val_accuracy: 0.8394
Epoch 6/50
22/22 [==============================] - 13s 565ms/step - loss: 0.3781 - accuracy: 0.8626 - val_loss: 0.3126 - val_accuracy: 0.8756
Epoch 7/50
22/22 [==============================] - 13s 565ms/step - loss: 0.3883 - accuracy: 0.8479 - val_loss: 0.3183 - val_accuracy: 0.8860
Epoch 8/50
22/22 [==============================] - 13s 567ms/step - loss: 0.3175 - accuracy: 0.8626 - val_loss: 0.2779 - val_accuracy: 0.9119

22/22 [==============================] - 13s 565ms/step - loss: 0.1704 - accuracy: 0.9335 - val_loss: 0.2154 - val_accuracy: 0.94
Epoch 44/50
22/22 [==============================] - 13s 565ms/step - loss: 0.1416 - accuracy: 0.9572 - val_loss: 0.2026 - val_accuracy: 0.93
Epoch 45/50
22/22 [==============================] - 12s 563ms/step - loss: 0.3711 - accuracy: 0.8818 - val_loss: 0.9131 - val_accuracy: 0.73
Epoch 46/50
22/22 [==============================] - 13s 569ms/step - loss: 0.4825 - accuracy: 0.8479 - val_loss: 0.2602 - val_accuracy: 0.90
Epoch 47/50
22/22 [==============================] - 12s 562ms/step - loss: 0.1613 - accuracy: 0.9365 - val_loss: 0.2317 - val_accuracy: 0.94
Epoch 48/50
22/22 [==============================] - 13s 567ms/step - loss: 0.2659 - accuracy: 0.9129 - val_loss: 0.4891 - val_accuracy: 0.87
Epoch 49/50
22/22 [==============================] - 12s 562ms/step - loss: 0.2890 - accuracy: 0.9055 - val_loss: 0.2827 - val_accuracy: 0.91
Epoch 50/50
22/22 [==============================] - 13s 565ms/step - loss: 0.1970 - accuracy: 0.9261 - val_loss: 0.2134 - val_accuracy: 0.94
```

Figure 23: ResNet model training and validation accuracy and loss.

The graphs below indicate the training and validation loss and accuracy curves of ResNet. Training and validation accuracy rises while training and validation loss reduces, as illustrated in the training loss and accuracy curve in Figures 24 and 25. The graph reveals some evidence of overfitting throughout some epochs where the validation accuracy is better than training accuracy.

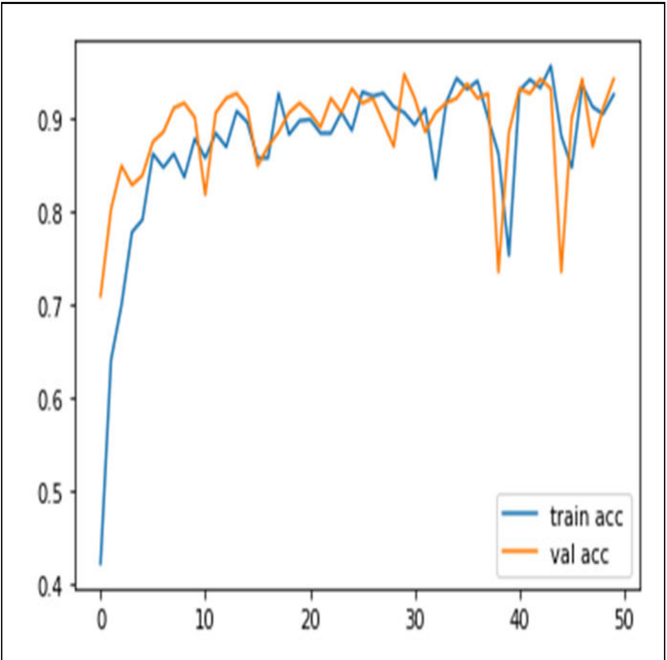


Figure 24: ResNet model training and validation accuracy graph.

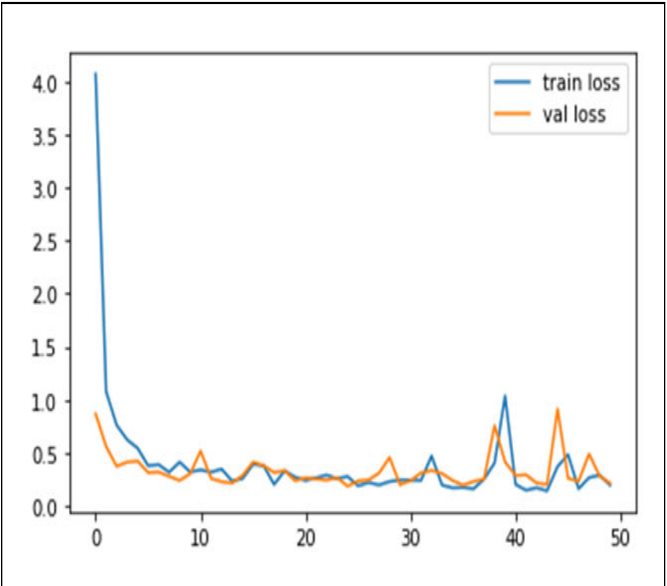
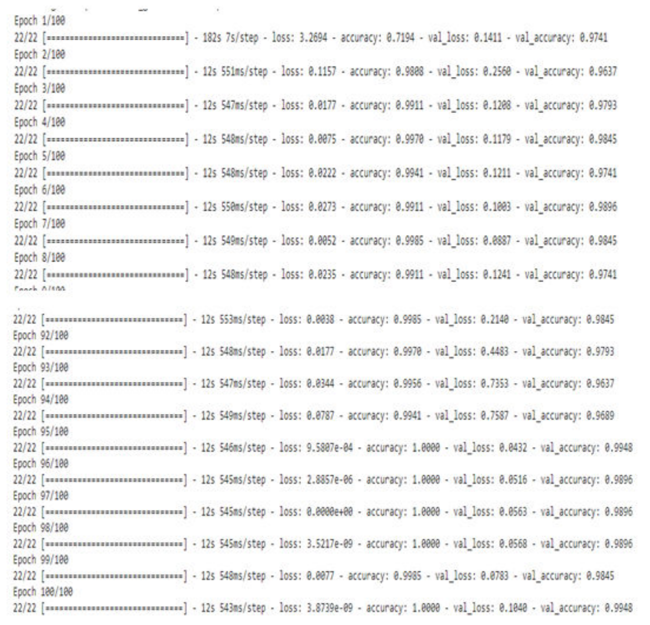


Figure 25: ResNet model training and validation loss graph.

Comparison with inception model

The performance (accuracy and loss value) of Inception v3 model is shown in Figures 26-28. Inception v3 achieves 99.78% training and 99.48% validation accuracy on our data. It has a training loss of 0.7% and a validation loss of 10.30%. It takes 23 minutes to train the model, taking 13.8 seconds per epoch on average. The size of the model is also huge (85.8 MB).



```
Epoch 1/100
22/22 [==============================] - 182s 7s/step - loss: 3.2694 - accuracy: 0.7194 - val_loss: 0.1411 - val_accuracy: 0.9741
Epoch 2/100
22/22 [==============================] - 12s 551ms/step - loss: 0.1157 - accuracy: 0.9808 - val_loss: 0.2560 - val_accuracy: 0.9637
Epoch 3/100
22/22 [==============================] - 12s 547ms/step - loss: 0.0177 - accuracy: 0.9911 - val_loss: 0.1208 - val_accuracy: 0.9793
Epoch 4/100
22/22 [==============================] - 12s 548ms/step - loss: 0.0075 - accuracy: 0.9970 - val_loss: 0.1179 - val_accuracy: 0.9845
Epoch 5/100
22/22 [==============================] - 12s 548ms/step - loss: 0.0222 - accuracy: 0.9941 - val_loss: 0.1211 - val_accuracy: 0.9741
Epoch 6/100
22/22 [==============================] - 12s 550ms/step - loss: 0.0273 - accuracy: 0.9911 - val_loss: 0.1003 - val_accuracy: 0.9896
Epoch 7/100
22/22 [==============================] - 12s 549ms/step - loss: 0.0052 - accuracy: 0.9985 - val_loss: 0.0887 - val_accuracy: 0.9845
Epoch 8/100
22/22 [==============================] - 12s 548ms/step - loss: 0.0235 - accuracy: 0.9911 - val_loss: 0.1241 - val_accuracy: 0.9741

22/22 [==============================] - 12s 553ms/step - loss: 0.0038 - accuracy: 0.9985 - val_loss: 0.2140 - val_accuracy: 0.9845
Epoch 92/100
22/22 [==============================] - 12s 548ms/step - loss: 0.0177 - accuracy: 0.9970 - val_loss: 0.4483 - val_accuracy: 0.9793
Epoch 93/100
22/22 [==============================] - 12s 547ms/step - loss: 0.0344 - accuracy: 0.9956 - val_loss: 0.7353 - val_accuracy: 0.9637
Epoch 94/100
22/22 [==============================] - 12s 549ms/step - loss: 0.0787 - accuracy: 0.9941 - val_loss: 0.7587 - val_accuracy: 0.9689
Epoch 95/100
22/22 [==============================] - 12s 546ms/step - loss: 9.5807e-04 - accuracy: 1.0000 - val_loss: 0.0432 - val_accuracy: 0.9948
Epoch 96/100
22/22 [==============================] - 12s 545ms/step - loss: 2.8857e-06 - accuracy: 1.0000 - val_loss: 0.0516 - val_accuracy: 0.9896
Epoch 97/100
22/22 [==============================] - 12s 545ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.0563 - val_accuracy: 0.9896
Epoch 98/100
22/22 [==============================] - 12s 545ms/step - loss: 3.5217e-09 - accuracy: 1.0000 - val_loss: 0.0568 - val_accuracy: 0.9896
Epoch 99/100
22/22 [==============================] - 12s 548ms/step - loss: 0.0077 - accuracy: 0.9985 - val_loss: 0.0783 - val_accuracy: 0.9845
Epoch 100/100
22/22 [==============================] - 12s 543ms/step - loss: 3.8739e-09 - accuracy: 1.0000 - val_loss: 0.1040 - val_accuracy: 0.9948
```

Figure 26: Inception v3 training and validation accuracy and loss.

The graphs below indicate the training and validation loss and accuracy curves of Inception v3. Training and validation accuracy rises while training and validation loss reduces, as illustrated in the training loss and accuracy curve in Figures 27 and 28. The graph reveals that inception performed very well on our dataset without overfitting and underfitting problems.

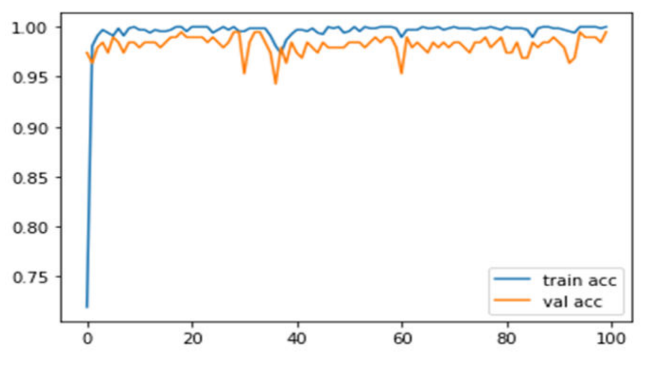


Figure 27: Inception v3 training and validation accuracy graph.

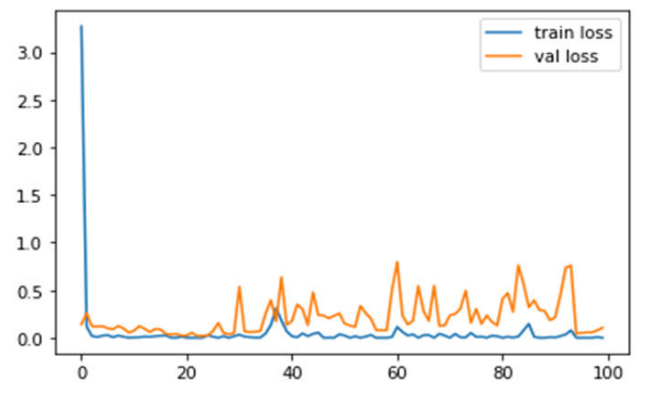
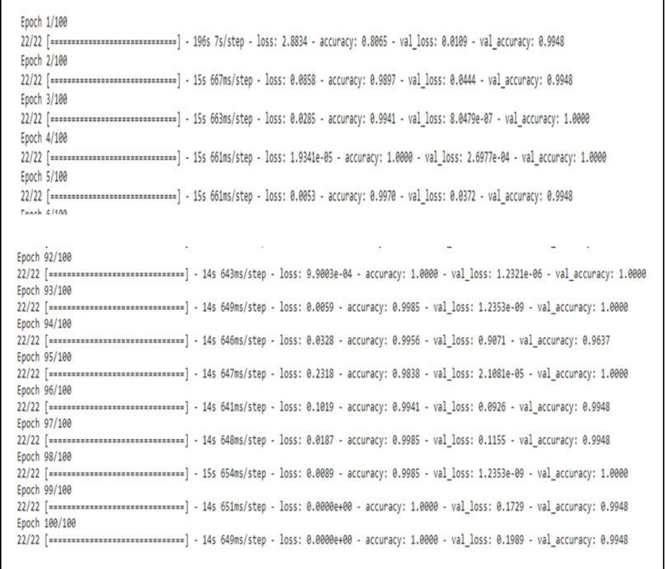


Figure 28: Inception v3 training and validation loss graph.

There are 34 mealybug class images, 34 bacterial wilt class images, and 32 healthy class images in our test dataset. From 34 images found in healthy class it correctly predicted 31 of them incorrectly predicted 2 images as mealybug and 1 image as bacterial wilt. From 34 images found in mealybug class it correctly predicted all of the images. It also correctly predicted all of the images found in bacterial wilt class. Totally Inception v3 predicted well on our test dataset with an accuracy of 97% and a loss of only 9.2%. As we can understand from the experimental results Inceptionv3 and ResNet-50 almost have the same performance towards our test dataset.

Comparison with DenseNet-201

The performance (accuracy and loss value) of DenseNet-201 model is shown in Figures 29-31. DenseNet-201 achieves 99.88% training and 99.48% validation accuracy on our data, as shown in figure 4. It has a training loss of 0.8% and a validation loss of 11.55%. It takes 27 minutes to train the model, taking 16.2 seconds per epoch on average. The size of the model is also huge (75 MB).



```
Epoch 1/100
22/22 [==============================] - 196s 7s/step - loss: 2.8834 - accuracy: 0.8065 - val_loss: 0.0109 - val_accuracy: 0.9948
Epoch 2/100
22/22 [==============================] - 15s 667ms/step - loss: 0.0858 - accuracy: 0.9897 - val_loss: 0.0444 - val_accuracy: 0.9948
Epoch 3/100
22/22 [==============================] - 15s 663ms/step - loss: 0.0285 - accuracy: 0.9941 - val_loss: 8.0479e-07 - val_accuracy: 1.0000
Epoch 4/100
22/22 [==============================] - 15s 661ms/step - loss: 1.9341e-05 - accuracy: 1.0000 - val_loss: 2.6977e-04 - val_accuracy: 1.0000
Epoch 5/100
22/22 [==============================] - 15s 661ms/step - loss: 0.0053 - accuracy: 0.9970 - val_loss: 0.0372 - val_accuracy: 0.9948

Epoch 92/100
22/22 [==============================] - 14s 643ms/step - loss: 9.9003e-04 - accuracy: 1.0000 - val_loss: 1.2321e-06 - val_accuracy: 1.0000
Epoch 93/100
22/22 [==============================] - 14s 649ms/step - loss: 0.0059 - accuracy: 0.9985 - val_loss: 1.2353e-09 - val_accuracy: 1.0000
Epoch 94/100
22/22 [==============================] - 14s 646ms/step - loss: 0.0328 - accuracy: 0.9956 - val_loss: 0.9071 - val_accuracy: 0.9637
Epoch 95/100
22/22 [==============================] - 14s 647ms/step - loss: 0.2318 - accuracy: 0.9838 - val_loss: 2.1081e-05 - val_accuracy: 1.0000
Epoch 96/100
22/22 [==============================] - 14s 641ms/step - loss: 0.1019 - accuracy: 0.9941 - val_loss: 0.0926 - val_accuracy: 0.9948
Epoch 97/100
22/22 [==============================] - 14s 648ms/step - loss: 0.0187 - accuracy: 0.9985 - val_loss: 0.1155 - val_accuracy: 0.9948
Epoch 98/100
22/22 [==============================] - 15s 654ms/step - loss: 0.0089 - accuracy: 0.9985 - val_loss: 1.2353e-09 - val_accuracy: 1.0000
Epoch 99/100
22/22 [==============================] - 14s 651ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.1729 - val_accuracy: 0.9948
Epoch 100/100
22/22 [==============================] - 14s 649ms/step - loss: 0.0000e+00 - accuracy: 1.0000 - val_loss: 0.1989 - val_accuracy: 0.9948
```

Figure 29: DenseNet-201 training and validation accuracy and loss.

The graphs below indicate the training and validation loss and accuracy curves of DenseNet-201.

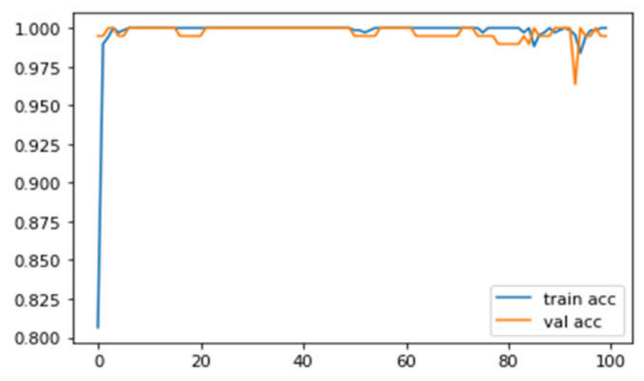


Figure 30: DenseNet-201 training and validation accuracy graph.

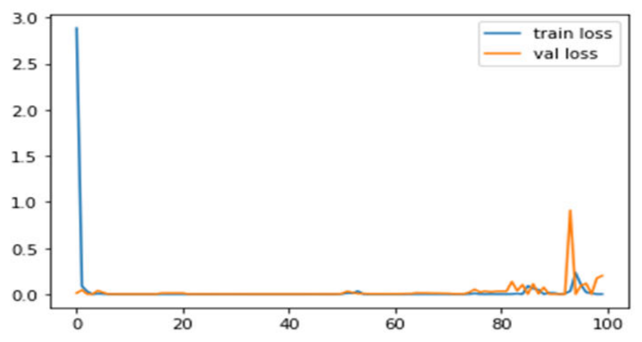
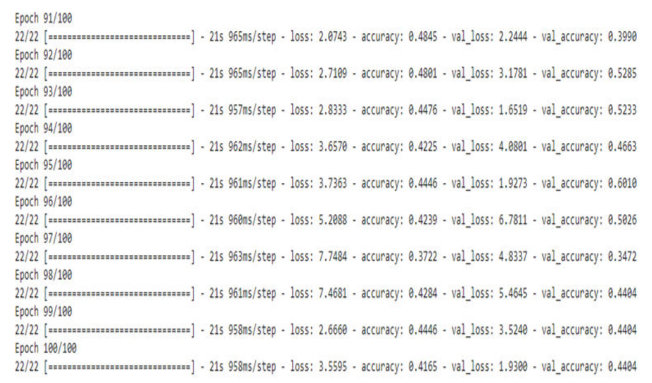


Figure 31: DenseNet training and validation loss graph.

There are 34 mealybug class images, 34 bacterial wilt class images, and 32 healthy class images in our test dataset. DenseNet-21 predicted all of our images to correct classes. Totally DenseNet-201 predicted perfectly on our test dataset with an accuracy of 100% and no loss at all, 0.0%. As we can understand from the experimental results DenseNet-201 performed best in test dataset.

Comparison with EfficientNet

The performance (accuracy and loss value) of EfficientNetB3 model is shown in Figures 32-34. DenseNet-201 achieves 48.01% training and 52.01% validation accuracy on our data. It has a training loss of 27.10% and a validation loss of 31.78%. It takes 41 minutes to train the model, taking 24.6 seconds per epoch on average. The size of the model is also huge (250 MB).



```
Epoch 91/100
22/22 [==============================] - 21s 965ms/step - loss: 2.0743 - accuracy: 0.4845 - val_loss: 2.2444 - val_accuracy: 0.3990
Epoch 92/100
22/22 [==============================] - 21s 965ms/step - loss: 2.7109 - accuracy: 0.4801 - val_loss: 3.1781 - val_accuracy: 0.5285
Epoch 93/100
22/22 [==============================] - 21s 957ms/step - loss: 2.8333 - accuracy: 0.4476 - val_loss: 1.6519 - val_accuracy: 0.5233
Epoch 94/100
22/22 [==============================] - 21s 962ms/step - loss: 3.6570 - accuracy: 0.4225 - val_loss: 4.0801 - val_accuracy: 0.4663
Epoch 95/100
22/22 [==============================] - 21s 961ms/step - loss: 3.7363 - accuracy: 0.4446 - val_loss: 1.9273 - val_accuracy: 0.6010
Epoch 96/100
22/22 [==============================] - 21s 960ms/step - loss: 5.2088 - accuracy: 0.4239 - val_loss: 6.7811 - val_accuracy: 0.5026
Epoch 97/100
22/22 [==============================] - 21s 963ms/step - loss: 7.7484 - accuracy: 0.3722 - val_loss: 4.8337 - val_accuracy: 0.3472
Epoch 98/100
22/22 [==============================] - 21s 961ms/step - loss: 7.4681 - accuracy: 0.4284 - val_loss: 5.4645 - val_accuracy: 0.4404
Epoch 99/100
22/22 [==============================] - 21s 958ms/step - loss: 2.6660 - accuracy: 0.4446 - val_loss: 3.5240 - val_accuracy: 0.4404
Epoch 100/100
22/22 [==============================] - 21s 958ms/step - loss: 3.5595 - accuracy: 0.4165 - val_loss: 1.9300 - val_accuracy: 0.4404
```

Figure 32: EfficientB3 training and validation accuracy and loss.

The graphs below indicate the training and validation loss and accuracy curves of EfficientNetB3.

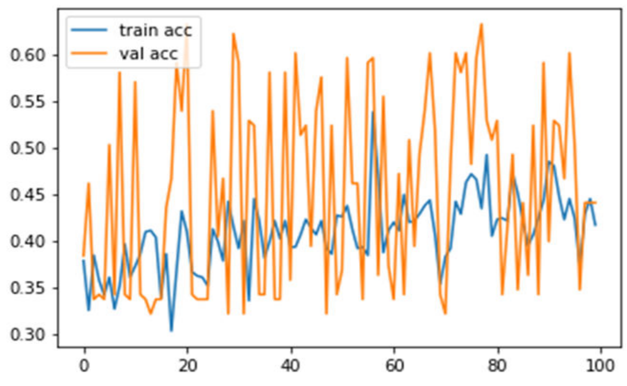


Figure 33: EfficientNetB3 training and validation accuracy graph.

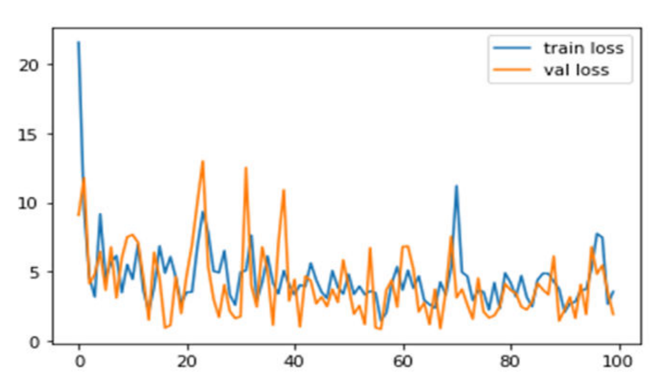


Figure 34: EfficientNetB3 training and accuracy loss graph.

As clearly depicted in Figure 35 the confusion matrix shows the following results. On our test dataset with an accuracy of 34% and a loss of 64%.

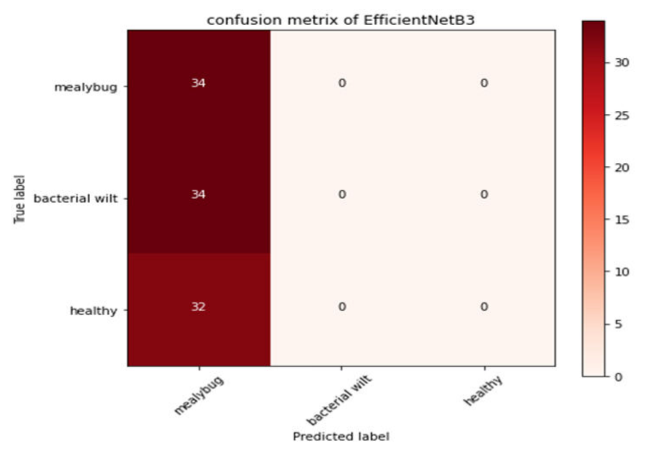
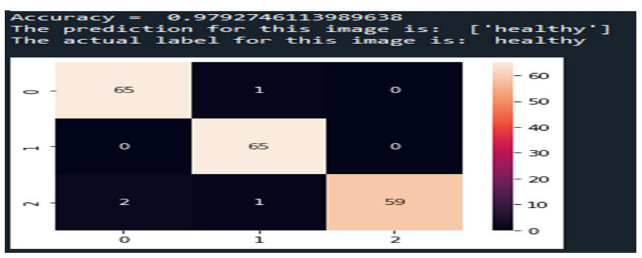


Figure 35: EfficientNetB3 confusion matrix.

Comparing our model with machine learning classifiers

Comparison with random forest classifier: Random forest classifier registered classification accuracy of 97.92%. As depicted in Figure 36, random forest missed 4 images from a total of 193 images.



```
Accuracy =  0.9792746113989638
The prediction for this image is:  ['healthy']
The actual label for this image is:  healthy
```

Figure 36: Random forest classifier.

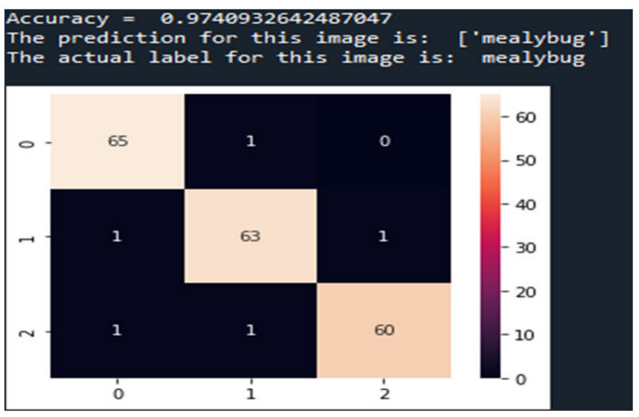
TABLE 3

Summary of comparisons

Models	Training accuracy, loss (in %)	Validation accuracy, loss	Number of misses during prediction	Model size (in Mb)	Training time (in minutes)
Our model	99.68, 0.0092	98.87, 0.0303	0	21	8 minutes
Our model (without segmentation)	99.26, 0.0238	97.93,0.0738	0	21	10.8
AlexNet-50	97.34, 0.0854	94.30, 0.2117	8	321	24
VGG16	99.89, 0.001	98.96, 0.0254	0	57.1	20

Comparison with XBoost classifier

Random forest classifier registered classification accuracy of 97.40%. As depicted in Figure 37, XBoost miss classified 5 images from a total of 193 images.

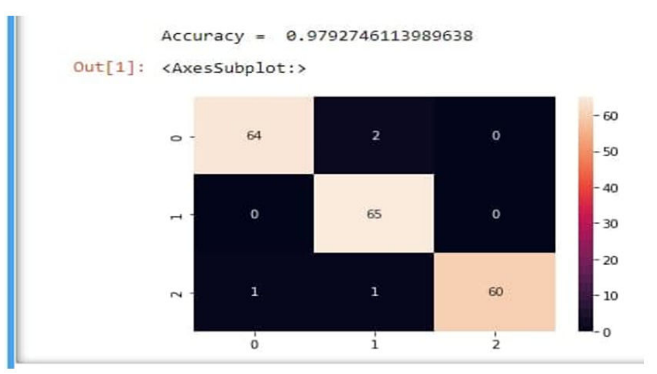


```
Accuracy =  0.9740932642487047
The prediction for this image is:  ['mealybug']
The actual label for this image is:  mealybug
```

Figure 37: XBoost classifier.

Comparison with support vector machine

Support vector machine registered classification accuracy of 97.92%. As depicted in Figure 38, support vector machine miss classified 4 images from a total of 193 images (Tables 3 and 4).



```
Accuracy =  0.9792746113989638
Out[1]: <AxesSubplot:>
```

Figure 38: SVM classification.

Summary of comparisons between proposed model and state of the art models (Tables 3 and 4).

ResNet-50	95.72, 0.1416	93.26, 0.2026	3	93.9	13
Inception v3	99.78, 0.007	99.48, 0.1030	3	85.8	23
DenseNet-201	99.88, 0.008	99.48, 0.1155	0	75	27
EfficientNetB3	48.01, 2.710	52.01, 3.178	66	250	41

TABLE 4

Summary of the comparison between our model and other models from related works

Researcher name	Test accuracy
Y. kibru	97.86%
Kibru A. and Getahun T.	94.04%
G. Selvaraj	97.36%
Our model	99.97%

DISCUSSION

With the exception of efficientNet, all of the models were evaluated on a second dataset that was not viewed during the model's training and yielded positive results. Model performance while training is measured using classification accuracy metrics, while model performance during testing is measured using the confusion matrix. Overall, our model is as accurate as the finest pre-trained models, but it is considerably smaller and requires less training time. The dataset that we utilized to train the model, *i.e.,* the images in the dataset are easily classified with human eyes, is the fundamental reason that the proposed model and most state of the art models offer good results. Another reason our proposed model performs well is that we used smaller filters in the network's convolution layer. The use of lower convolution sizes aids in the identification of extremely small features that are utilized to distinguish between the input image and the output image, and the risk of losing a crucial feature is greatly reduced.

This chapter describes in detail the experimental evaluation of the suggested methodology for the automatic detection of *Enset* disease and pests. The dataset used and the proposed model's implementation are detailed in depth. In addition, the outcomes of the tests are reported and compared to current models. Our model had a high level of classification accuracy and properly predicted all of the test image datasets. When compared with previous research on *Enset* disease identification, our model registered better accuracy and less training time. In comparison to state of the art models, it was also trained faster and weighs less.

CONCLUSION

Bacterial wilt and root mealy bugs are wreaking havoc on *Enset* production, reducing the quantity and quality of the crop. Robots, temperature and moisture sensors, aerial photos, GPS technology, and computer vision will all is used in future agriculture. To begin the future farming approach of *Enset*, we built this *Enset* disease detection using computer vision utilizing CNN. CNN was highlighted as a significant contributor and a recent research topic for the automatic detection of *Enset* diseases and pests in this study. Following a thorough study of the literature and relevant research, we discovered that the detection of *Enset* diseases and pests receives little attention and remains an unsolved problem. The aforementioned issues were addressed in this study by incorporating extra components like strides and a pooling module into the basic or default CNN design. Using images of normal and diseased *Enset*, a deep CNN system for detecting *Enset* diseases and pests were constructed. As a result, reliance on human specialists' talent and experience is lessened.

The importance of the proposed system will result in a shift in detection progress in terms of accuracy and efficiency. In the detection of *Enset* diseases and pests, the system has learned a 99.68% training accuracy and 98.87% validation accuracy. As a result, the established model can assist specialists, investors, and farmers in accurately detecting diseases and pests before they spread. The results of our experiment show that the suggested CNN model may considerably aid in the accurate detection of Bacterial Wilt, root mealy bug, and healthy *Enset* images with minimal computing effort and a limited number of images. The suggested model performs better in terms of accuracy, loss, training time, and model size. In terms of precision, we have a training accuracy of 99.68% and a testing accuracy of 98.87%. In comparison to earlier versions, the model is much lighter and trains much faster.

RECOMMENDATIONS

The deep CNN model suggested in this study can be utilized to detect problems such as bacterial wilt and mealybugs in bananas. Future agriculture will use sophisticated technologies such as autonomous drones and mobile applications. We recommend developing mobile application that automatically detects diseases in real time. We also recommend further study on developing autonomous drones that can detect diseases and take accordingly measures.

ACKNOWLEDGEMENT

The authors would like to thank Jimma university and Wolkite university for their financial support.

REFERENCES

1. Befekadu H, Fininsa C, Terefe H, et al. Distribution and management of bacterial wilt (*Xanthomonas campestris* pv. Musacearum) of Enset in Ethiopia. Ethiop J Agric Sci. 2020;30(3):40-53.
2. McKnight. Integrated management of bacterial wilt of *Enset* (*Ensete ventricosum* (Welw.) Cheesman) caused by *Xanthomonas campestris* pv. musacearum in Ethiopia. McKnight foundation, Hawassa. 2013.
3. Bedaso M, Meshesha M, Diriba C, et al. Comparing performance of classification algorithms to use for grading coffee's raw quality by using image processing techniques. AGBIR. 2023;39(2):491-495.
4. Gonzalez R, Woods R. Digital Image Processing. 4th edition. Pearson. Hudson Street, New York. 2018.
5. Yamashita R, Nishio M, Togashi K. Convolutional neural networks: An overview and application in radiology. Insights Imaging. 2018;9:611–629.
6. Addis T, Azerefegne F, Blomme. Density and distribution of *Enset* root mealybugs on *Enset*. Afr Crop Sci J. 2014;16(1):67-74.
7. Blomme G, Dita M, Molina A, et al. Bacterial diseases of bananas and *Enset*: Current state of knowledge and integrated approaches toward sustainable management. Front Plant Sci. 2017;8:25.

8. Dobbin K, Simon R. Optimally splitting cases for training and testing high dimensional. BMC Med Genomics. 2017;4(31):1-8.