CORN LEAF DISEASE CLASSIFICATION USING TRANSFER LEARNING METHODS

Report submitted to the SASTRA Deemed to be University in partial fulfillment of the requirements for the award of degree of

Bachelor of Technology

Submitted by

Rakshit Acharya

(Reg. No.:123003203, CSE)

Palakurty S Venkata Sai Sathwik

(Reg. No.: 123003175 , CSE)

Alapati Yashwanth

(Reg. No.: 123003276, CSE)

MAY 2023

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

THANJAVUR | KUMBAKONAM | CHENNAI

SCHOOL OF COMPUTING
THANJAVUR – 613 401

<u>Bonafide Certificate</u>

This is to certify that the project report titled "**Corn Leaf Disease Classification Using Transfer Learning Methods**" submitted in partial fulfillment of the requirements for the award of the degree of Tech Computer Science and Engineering to the SASTRA Deemed to be University, is a bona-fide record of the work done by **Mr. Rakshit Acharya (Reg. No. 123003203), Mr. Palakurty S Venkata Sai Sathwik (Reg. No. 123003175) & Mr. Alapati Yashwanth (Reg. No.123003276)** during the final semester of the academic year 2022- 23, in the **School of Computing**, under my supervision. This report has not formed the basis for the award of any degree, diploma, associate ship, fellowship or other similar title to any candidate of any University.

Signature of Project Supervisor	: [illegible]
Name with Affiliation	: Ms. Renuga Devi T, Asst. Professor – II
Date	: 11-May-2023

Project *Viva voce* held on ____________________

Examiner 1 **Examiner 2**

SASTRA
ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION
DEEMED TO BE UNIVERSITY
(U/S 3 of the UGC Act, 1956)
THINK MERIT | THINK TRANSPARENCY | THINK SASTRA
THANJAVUR | KUMBAKONAM | CHENNAI

SCHOOL OF COMPUTING
THANJAVUR – 613 401

Declaration

We declare that the project report titled "**Corn Leaf Disease Classification Using Transfer Learning Methods**" submitted by us is an original work done by us under the guidance of **Ms. Renuga Devi T, Asst. Professor – II, School of Computing, SASTRA Deemed to be University** during the final semester of the academic year 2022-23, in the **School of Computing**. The work is original and wherever we have used materials from other sources, we have given due credit and cited them in the text of the report. This report has not formed the basis for the award of any degree, diploma, associate-ship, fellowship or other similar title to any candidate of any University.

Signature of the candidate(s) : Sathwik
Rakshit
[illegible]

Name of the candidate(s) : **1)** Palakurty S Venkata Sai Sathwik
2) Rakshit Acharya
3) Alapati Yashwanth

Date : 11-May-2023

Acknowledgements

We would like to thank our Honorable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honorable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan,** Dean**,** Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli,** Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. A. Umamakeswari**, Dean, School of Computing, **Dr. B.Santhi**, Associate Dean, Research, Dr. S.Gopalakrishnan, Department of Computer Application. **Dr. R. Muthaiah**, Associate Dean, Department of Information Technology and Information & Communication Technology and **Dr. V. S. Shankar Sriram**, Associate Dean, Department of Computer Science and Engineering for their motivation and support offered in materializing this project.

Our guide Ms. Renuga Devi T, Asst. Professor - II School of Computing was the driving force behind this whole idea from the start. Her deep insight in the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing me an opportunity to showcase my skills through project.

LIST OF FIGURES

Figure No.	Title	Page No.
1.1	Gray leaf spot	1
1.2	Common rust	1
1.3	Blight	1
1.4	Late blight	2
1.5	Early blight	2
3.1	Workflow Diagram	4
3.2	Dataset before and after RandomOverSampler for corn leaf	6
3.3	Dataset before and after RandomOverSampler for potato leaf	6
5.1.1	Accuracy and loss plots of RESNET152 for corn leaf	37
5.1.2	Classification Report of RESNET152 for corn leaf	37
5.2.1	Accuracy and loss plots of Inception V3 for corn leaf	38
5.2.2	Classification Report of Inception V3 for corn leaf	38
5.3.1	Accuracy and loss plots of DenseNet121 for corn leaf	39
5.3.2	Classification Report of DenseNet121 for corn leaf	39
5.4.1	Accuracy and loss plots of EfficientB0 for corn leaf	40
5.4.2	Classification Report of EfficientB0 for corn leaf	40
5.5.1	Accuracy and loss plots of proposed model for corn leaf	41
5.5.2	Classification Report of proposed model for corn leaf	41

5.6.1	Graphical User Interface for corn leaf	42
6.1.1	Accuracy and loss plots of RESNET152 for potato leaf	43
6.1.2	Classification Report of RESNET152 for potato leaf	43
6.2.1	Accuracy and loss plots of DENSENET121 for potato leaf	44
6.2.2	Classification Report of DENSENET121 for potato leaf	44
6.3.1	Accuracy and loss plots of EFFICIENTNET B0 for potato leaf	45
6.3.2	Classification Report of EFFICIENTNET B0 for potato leaf	45
6.4.1	Accuracy and loss plots of INCEPTION V3 for potato leaf	46
6.4.2	Classification Report of INCEPTION V3 for potato leaf	46
6.5.1	Accuracy and loss plots of proposed model for potato leaf	47
6.5.2	Classification Report of proposed model for potato leaf	47
6.6	GUI for potato leaf	48

LIST OF TABLES

Table No.	Table name	Page No.
7.1	Individual Contributions	49

TABLE OF CONTENTS

Title	Page No.
Bonafide Certificate	i
Declaration	ii
Acknowledgements	iii
List of Figures	v
List of Tables	vi
Synopsis	x
Abbreviation	xii
1. Introduction	1
1.1. Summary of the Base Paper	2
2. Literature Survey	3
3. Proposed Work	4
3.1 Methodology	4
3.1.1 Dataset Collection	5
3.1.2 Data Preprocessing	6
3.1.3 Model Implementation	7
3.1.4 Choosing the best performing model	7
3.1.5 Interactive WebApp for prediction	8
4. Source Code	9
5. Results and Snapshots for corn leaf	37
6. Results and Snapshots for potato leaf	43
7. Project Team and Individual Contributions	49
8. Conclusion and Future Plans	50
9. References	51

SYNOPSIS

Project Team Number: 23SOCU2084

Register No:	**Name:**
123003175	Palakurty S Venkata Sai Sathwik

Project Title: Corn leaf disease classification using transfer learning methods

Name of the guide: Renuga Devi T

Abstract

The problem of accurately identifying and classifying the plant diseases is of great importance in field of agriculture. In recent years, deep learning models have shown promising results in image classification tasks. However, the problem of large parameter size in the models still persists.

In our work, an end-to-end deep learning model is used for corn leaf disease classification into healthy and unhealthy categories while considering the models' number of parameters. We carry out the pre-processing steps wherein data augmentation techniques are used which increase the number and variety of images thus enabling the model to learn complex scenarios efficiently. The pre-processing step is followed by a convolutional neural network (CNN) architecture that is trained on a large dataset of labeled images of different corn leaf diseases. We employ two pre-trained CNN's: EfficientNetB0 and DenseNet121.The model is optimized using transfer learning techniques, where the pre-trained CNN models are fine-tuned on the corn leaf disease dataset.

Experimental results are compared with other pre-trained convolutional neural network models such as RenseNet152 and InceptionV3. The end-to-end deep learning CNN model for corn leaf disease classification has the potential to be applied in real-world scenarios to help farmers detect and diagnose plant diseases accurately and quickly, which could ultimately lead to increased crop yield and reduced economic losses.

Specific Contribution

- Data Pre-processing and Data collection.
- Worked on two models named DENSENET121 and EFFICIENTNET B0 for corn leaf model generation and other two models named RENSENET152 and INCEPTION V3 for potato leaf model creation.

Specific Learning

- Learned various Data Pre-processing concepts such as ImageDataGenerator, various CNN models with basics of deep-learning techniques and hands-on experience with the Streamlit package for GUI design.

Project Team Number: 23SOCU2084

Register No:	**Name:**
123003276	Alapati Yashwanth

Project Title: Corn leaf disease classification using transfer learning methods

Name of the guide: Renugadevi T

Abstract

The problem of accurately identifying and classifying the plant diseases is of great importance in field of agriculture. In recent years, deep learning models have shown promising results in image classification tasks. However, the problem of large parameter size in the models still persists.
In our work, an end-to-end deep learning model is used for corn leaf disease classification into healthy and unhealthy categories while considering the models' number of parameters. We carry out the pre-processing steps wherein data augmentation techniques are used which increase the number and variety of images thus enabling the model to learn complex scenarios efficiently. The pre-processing step is followed by a convolutional neural network (CNN) architecture that is trained on a large dataset of labeled images of different corn leaf diseases. We employ two pre-trained CNN's: EfficientNetB0 and DenseNet121.The model is optimized using transfer learning techniques, where the pre-trained CNN models are fine-tuned on the corn leaf disease dataset.
Experimental results are compared with other pre-trained convolutional neural network models such as RenseNet152 and InceptionV3. The end-to-end deep learning CNN model for corn leaf disease classification has the potential to be applied in real-world scenarios to help farmers detect and diagnose plant diseases accurately and quickly, which could ultimately lead to increased crop yield and reduced economic losses.

Specific Contribution

- Data Pre-processing and Data collection.
- Worked on two models named RENSENET152 and INCEPTION V3 for corn leaf model generation and other two models named DENSENET121 and EFFICIENTNET B0 for potato leaf model creation.

Specific Learning

- Learned various Data Pre-processing concepts such as ImageDataGenerator, various CNN models with basics of deep-learning techniques and hands-on experience with the Django framework for GUI design.

Project Team Number: 23SOCU2084

Register No:	**Name:**
123003203	Rakshit Acharya

Project Title: Corn leaf disease classification using transfer learning methods

Name of the guide: Renugadevi T

Abstract

The problem of accurately identifying and classifying the plant diseases is of great importance in field of agriculture. In recent years, deep learning models have shown promising results in image classification tasks. However, the problem of large parameter size in the models still persists.
In our work, an end-to-end deep learning model is used for corn leaf disease classification into healthy and unhealthy categories while considering the models' number of parameters. We carry out the pre-processing steps wherein data augmentation techniques are used which increase the number and variety of images thus enabling the model to learn complex scenarios efficiently. The pre-processing step is followed by a convolutional neural network (CNN) architecture that is trained on a large dataset of labeled images of different corn leaf diseases. We employ two pre-trained CNN's: EfficientNetB0 and DenseNet121.The model is optimized using transfer learning techniques, where the pre-trained CNN models are fine-tuned on the corn leaf disease dataset.
Experimental results are compared with other pre-trained convolutional neural network models such as RenseNet152 and InceptionV3. The end-to-end deep learning CNN model for corn leaf disease classification has the potential to be applied in real-world scenarios to help farmers detect and diagnose plant diseases accurately and quickly, which could ultimately lead to increased crop yield and reduced economic losses.

Specific Contribution

- Data Pre-processing and Data collection.
- Worked on proposed model which is the combination of DENSENET121 and EFFICIENTNET B0 for both corn leaf and potato leaf.
- Developed GUI for all the model execution of corn and potato leaf using Django framework and Streamlit package.

Specific Learning

- Learned various Data Pre-processing concepts such as ImageDataGenerator, basics of deep-learning techniques and hands-on experience with the Streamlit package.

ABBREVIATIONS

GUI	Graphical User Interface
CNN	Convolutional Neural Network
RESNET	Residual Network
WebApp	Web Application

CHAPTER 1

INTRODUCTION

In order to be used for various purposes, e.g. food, animal feed or biofuels, corn is a major crop around the world. However, corn is at risk of various diseases that can have significant effects on crop yields and quality which could result in economic losses for farmers.

Corn leaf blight, which occurs because of a fungus named Exserohilum Turcicum, is one common corn disease. The fungus may be able to infect corn leaves, resulting in lesions that are initially small, water soaked and brown, but gradually develop into grey with a tan colour. In addition, the disease may reduce leaf photosynthetic capacity and result in a reduction in yields and poor kernel quality.

Common rust caused by the fungus Puccinia sorgia is also one of the commonly occurring corn diseases. Rust infections can lead to yellowish to reddish-brown pustules on the leaves, reducing photosynthesis and causing defoliation, which can also lead to yield losses.

As farmers have to use more effective treatments such as fungicides, they find themselves in many difficulties with regards to dealing with corn disease, which has an impact on yields, quality and costs of production. These diseases can also have a detrimental effect on the quality of corn used for animal feed, which may lead to further economic losses.

Overall, corn disease can have a significant effect on yield, quality and economic well-being of farmers which make it essential for them to take precautionary measures in order to prevent illness from affecting their crops.

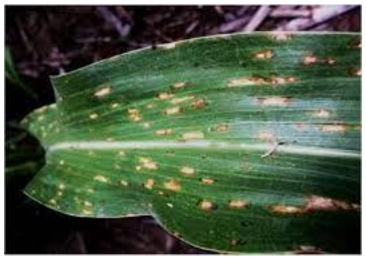


Fig 1.1 Gray leaf spot

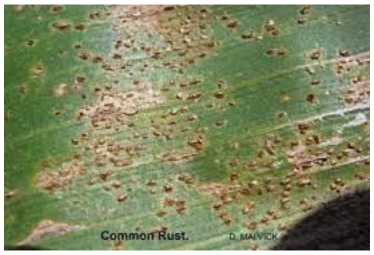


Fig 1.2 Common rust

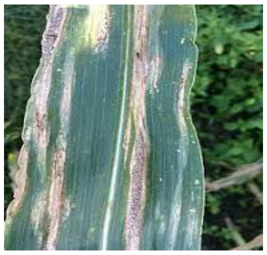


Fig 1.3 Blight

Additionally, both late blight and early blight are proven to affect the potato crop at any point of time in its growth and development. Visual inspection of the crop to find symptoms for these diseases is a very tedious and time-consuming job. Once the disease reaches a certain threshold, it becomes

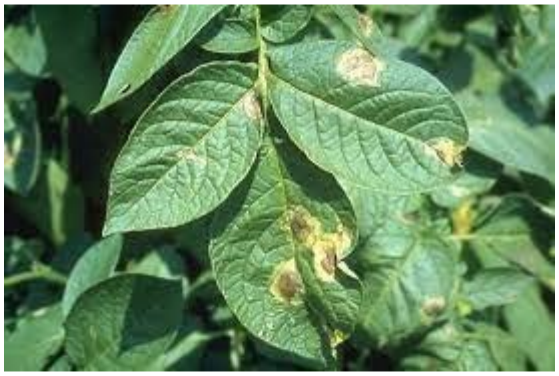


Fig.1.4 Late blight

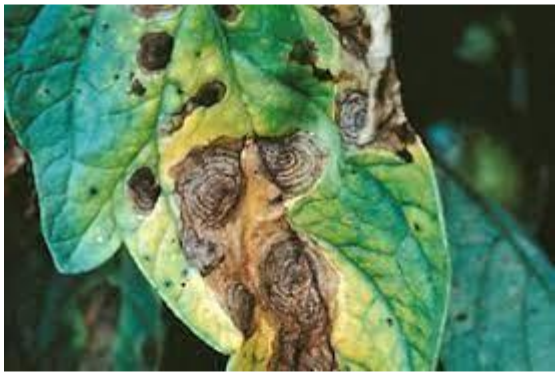


Fig.1.5 Early Blight

Current advancements in the field of imaging and sensing, artificial intelligence and machine learning enable us to do this task the “smart” way. Various machine learning algorithms and models have been developed for this sake, but they weren’t able to produce the desired result due to the vast amount of data, and huge amount of variation in the data. Hence we need to develop a model that thinks like us, the human brain. The model that closely resembles and imitates the human brain is the Neural Network, which is a part of deep learning.

So in the proposed work, we explore four deep learning models, namely – RESNET152, Inception V3, DneseNet121, EfficientB0 and the proposed model to classify the corn and potato leaf diseases. Then all the models are linked with the WebApp GUI for use.

1.1. Summary of Base Paper

Title: End-To-End Deep Learning Model for Corn Leaf Disease Classification

Author: Hassan amin, Ashraf darwish, Aboul ella hassanien and Mona soliman

Journal Name: IEEE access

Year: 2022

Indexing: SCOPUS INDEXED - SITE SCORE - 6.7, IMPACT FACTOR 3.476

Base paper URL: https://ieeexplore.ieee.org/document/973401

CHAPTER 2
LITERATURE SURVEY

Paper Title	Pub- lished year	Work	Method
The identification of corn leaf diseases based on transfer learning and data augmentation	2020	The algorithm augments data and uses transfer learning to build a convolutional neural network for improved model accuracy and generalization.	The optimization model was obtained by fine-tuning GoogLeNet and adjusting parameters such as optimizer and learning rate.
Maize leaf disease classification using convolutional neural networks and hyperparameter optimization.	2020	Maize is a vital food crop worldwide, but its production is adversely affected by various diseases.Disease identification is a time-consuming and subjective task.	Computer vision automates maize disease identification in agriculture. The study examines the correlation between maize leaf classes and data augmentation impact on pre-trained models. .
An optimized dense convolutional neural network model for disease recognition and classification in corn leaf.	2020	Optimized DenseNet model performs similarly to established CNN architectures with fewer parameters and computation time.	Uses fewer parameters compared to other CNNs like EfficientNet, VGG19Net, NASNet, and Xception Net.
An effective automatic system deployed in agricultural Internet of Things using Multi-Context Fusion Network towards crop disease recognition in the wild.	2020	A deep learning system called Multi-Context Fusion Network is developed for crop disease recognition in agricultural IoT.	MCFN with deep fusion model outperforms state-of-the-art methods in wild crop disease recognition using domain-specific dataset.

CHAPTER 3

PROPOSED WORK

The objective of this project is to present a system to classify the leaves into the four classes i.e. three types of diseases for corn crop and three classes i.e. two types of diseases and healthy categories for potato crop based on the leaf condition by utilizing deep learning techniques. Transfer learning methods are used for classification and the models are compared based on various evaluation metrics such as precision, recall and accuracy. The various models used for classification are RESNET152, Inception V3, EfficientNetB0, DenseNet121 and Multi-input model (proposed model). An interactive GUI was built for predicting if the newly uploaded image of a corn leaf is gray leaf spot, common rust, blight or healthy and for potato leaf it is early blight, late blight, or healthy.

3.1 Methodology:

The proposed methodology for classifying the corn leaf and potato leaf diseases is carried out in five phases. The work flow of the steps involved is shown in Figure 3.1.1.

- Dataset Collection
- Data Preprocessing
    - RandomOverSampler
    - ImageDataGenerator
- Implementing the models
- Choosing the best performing model
- Interactive web app for prediction.

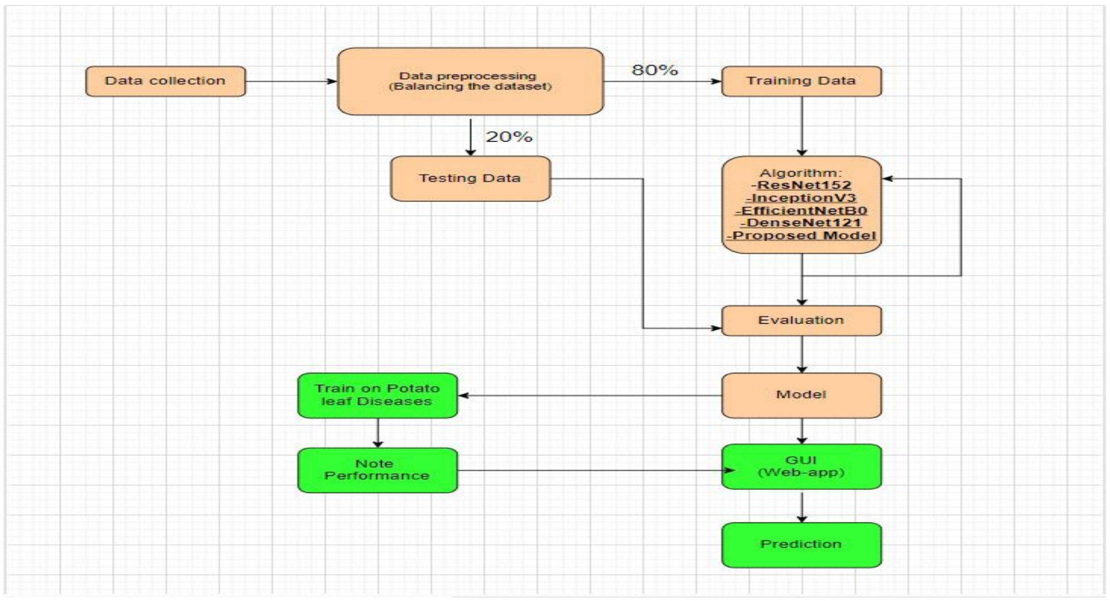

Data collection
Data preprocessing (Balancing the dataset)
80%
Training Data
20%
Testing Data
Algorithm: -ResNet152 -InceptionV3 -EfficientNetB0 -DenseNet121 -Proposed Model
Evaluation
Model
Train on Potato leaf Diseases
Note Performance
GUI (Web-app)
Prediction

Figure 3.1 Work flow diagram

3.1.1 Dataset Collection:

The dataset for the potato leaf diseases was obtained from the PlantVillage dataset, which was published by David. P. Hughes, Marcel Salathe. The datasets include over 50,000 images of 14 crops such as potatoes, grapes, tomato, apples etc.

For the corn leaves-

Total number of images = 4185

No. of images in Blight = 1146

No. of images in Common rust =1303

No. of images in Green leaf spot = 574

No. of images in Healthy =1162

For the potato leaves-

Total number of images = 2152

No. of images in Early Blight = 1000

No. of images in Late Blight =1000

No. of images in Healthy =152

3.1.2 Data Preprocessing:

For the corn crop the number of images in the green leaf spot category is 574 and the other categories have more than 1000 each. Imbalanced classifications pose a challenge for predictive modeling as most of the deep learning algorithms will have poor predictive accuracy as there isn’t enough data for the model to learn the features of the minority class thus the images from the minority class will be classified wrongly.

RandomOverSampler: (Novelty)

To solve the above problem, RandomOverSampler was used which randomly chooses samples from the minority class and duplicates them with replacement in the dataset. This automatically balances the minority classes with the majority class/classes. The major demerit of this method is that it may cause overfitting of data in few models as the same samples of data are being duplicated and used for training purposes.

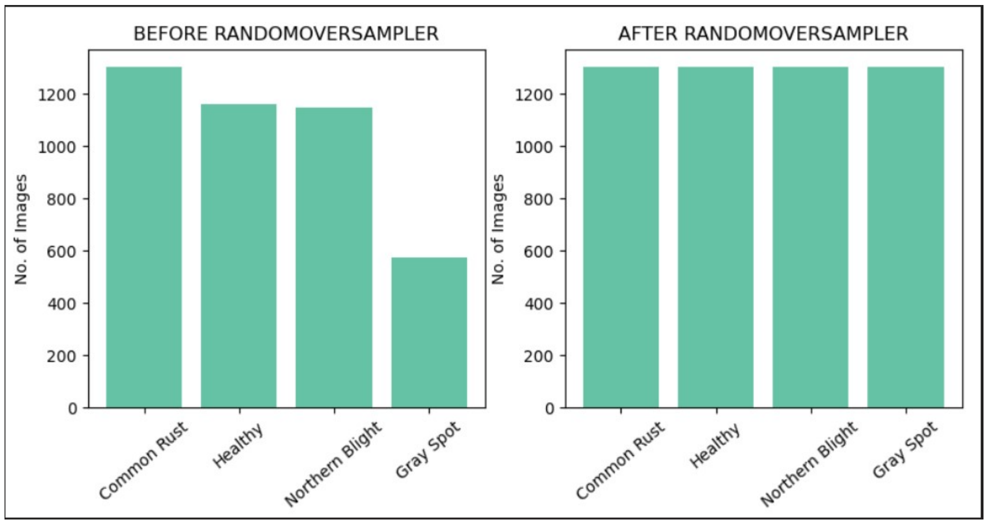


Figure 3.2 Dataset before and after RandomOverSampler for corn crop

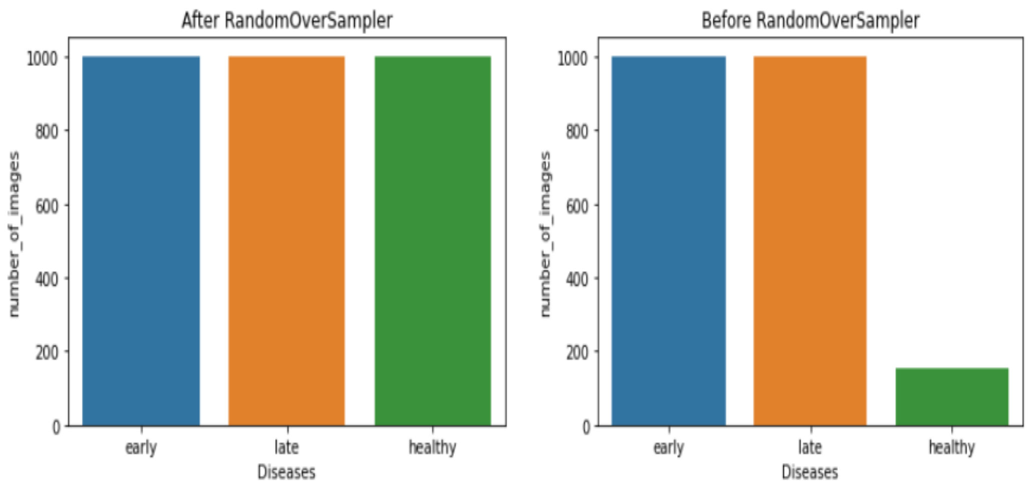


Figure 3.3 Dataset before and after RandomOverSampler for potato crop

ImageDataGenerator:

It is an image augmentation technique that is used when there is not enough data to train the model. This is a great way to expand the size of our dataset. The transformation techniques used were horizontal_flip, rotation_range, zoom_range, width_shift_range, height_shift_range. Using these techniques, new transformed images from the original dataset will be used to train the model. This also helps in the model learning various images in different angles and perspectives and thus reduces the chances of the model overfitting the data.

3.1.3 Model Implementation

A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data. In this study, several deep learning models, namely RESNET152, InceptionV3, DenseNet121, EfficientB0 and Multi-input model were trained to classify diseases in the potato and corn leaf. The results of these models were evaluated on the basis of a few metrics such as accuracy, precision and recall. Transfer learning from the imagenet dataset was used where the model is pre-trained and the weights are being used for classification of images on a different problem. All four models were trained on the training dataset and the proposed model which is the combination of EfficientNetB0 and DenseNet121; DensNet121 gave the best results with around **97.79%** validation accuracy.

The models were run for 20 epochs; batch size = 64,optimizer used - RMSprop, learning rate = 0.002 , test size = 20%, validation split = 20%.

3.1.4 Choosing the best performing model

Among all the four given models, DenseNet121 outperforms the other models. The model after training for 20 epochs gave ~97.79% validation accuracy and 92%, 100%, 96% and 100% precision on the 4 classes respectively. During the training of the model, first the raw conventional base of the model was imported with all the layers (convolutional and pooling layers), and the weights were initialized using ImageNet. On top of these layers a classifier was added. The output from the conventional base is the input for the last five classifier layers, that consists of a flatten layer and three dense layers and finally the output layer will be used to predict the image as any of thefour classes.

3.1.5 Interactive WebApp for prediction(GUI):

An interactive web application was developed for corn leaf crop disease detection which uses all the model trained on the given dataset for classifying newly uploaded images.

This web-app was developed using the django package in python. Django is a model-template-views-based web framework that is free and open-source that uses Python.

For the potato crop based disease detection a separate web app was developed which, in order to classify new uploaded images, uses all models that are trained on this dataset.

This web-app was developed using the streamlit package in python. Streamlit is an open source app framework which is used to build webapps for machine learning and data science related projects in a short period of time.

CHAPTER 4

SOURCE CODE

4.1: Corn Leaf Disease

```
import numpy as np import os
import pandas as pd
import matplotlib.pyplot as plt
from keras.applications.resnet import ResNet50,ResNet152
from keras.applications.inception_v3 import InceptionV3
from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Sequential, Model, load_model
from tensorflow.keras.layers import GlobalAveragePooling2D,AveragePooling2D,
Dense, Dropout, BatchNormalization, Input, Conv2D, MaxPooling2D, Dense,
GlobalAveragePooling2D, Activation, Dropout, Flatten, Dense, Input
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers



train_PATH = '/kaggle/input/aug-data/aug_data/Train' val_PATH =
"/kaggle/input/aug-data/aug_data/valid" SIZE = 224
BATCH_SIZE = 32
train_ds = tf.keras.preprocessing.image_dataset_from_directory( train_PATH,
seed=1337, image_size=(SIZE,SIZE), batch_size=BATCH_SIZE,
label_mode='categorical',
)
val_ds = tf.keras.preprocessing.image_dataset_from_directory( val_PATH,
seed=1337, image_size=(SIZE,SIZE), batch_size=BATCH_SIZE,
label_mode='categorical',
)
CATEGORIES = ["Gray_leaf_spot","Common_Rust","Northern_Lead_Blight","healthy"]
Found 14632 files belonging to 4 classes.
Found 3658 files belonging to 4 classes.

import matplotlib.pyplot as plt
plt.figure(figsize=(15, 10))
```

```
for images, labels in train_ds.take(1): for i in range(10):
ax = plt.subplot(3, 5, i + 1) plt.imshow(images[i].numpy().astype("uint8"))
plt.title(CATEGORIES[np.argmax(labels[i])]) plt.axis("off")

base_model = tf.keras.applications.InceptionV3(input_shape=(224,224,3),
include_top=False, weights='imagenet')
for layer in base_model.layers: layer.trainable=False
x = tf.keras.layers.Flatten()(base_model.output)
x = tf.keras.layers.Dense(512,activation="relu")(x) x =
tf.keras.layers.Dense(256,activation="relu")(x) x =
tf.keras.layers.Dense(128,activation="relu")(x) x =
tf.keras.layers.Dropout(0.2)(x)
prediction = tf.keras.layers.Dense(4, activation='softmax')(x) model =
Model(inputs=base_model.input, outputs=prediction)
from tensorflow.keras.optimizers import RMSprop
model.compile(optimizer=RMSprop(learning_rate=0.001),
loss="categorical_crossentropy", metrics=["accuracy"])
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy'
, patience = 2
, verbose=1
,factor=0.75
, min_lr=0.00001)
history2 = model.fit(train_dataset,
epochs=20 , batch_size=64,
validation_data = val_dataset, callbacks = [reduce_lr])
In [22]:
base_model = tf.keras.applications.DenseNet121(input_shape=(224,224,3),
include_top=False, weights='imagenet')

for layer in base_model.layers: layer.trainable=False

x = tf.keras.layers.Flatten()(base_model.output)
x = tf.keras.layers.Dense(512,activation="relu")(x) x =
tf.keras.layers.Dense(128,activation="relu")(x) x =
tf.keras.layers.Dropout(0.2)(x)
prediction = tf.keras.layers.Dense(4, activation='softmax')(x)
model = Model(inputs=base_model.input, outputs=prediction)
from tensorflow.keras.optimizers import RMSprop
model.compile(optimizer=RMSprop(learning_rate=0.001),
loss="categorical_crossentropy", metrics=["accuracy"])
```

```
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy'
, patience = 2
, verbose=1
,factor=0.75
, min_lr=0.00001)

history2 = model.fit(train_dataset,
epochs=20 , batch_size=64,
validation_data = val_dataset, callbacks = [reduce_lr])

base_model = tf.keras.applications.EfficientNetB0(input_shape=(224,224,3),
include_top=False, weights='imagenet')

for layer in base_model.layers: layer.trainable=False

x = tf.keras.layers.Flatten()(base_model.output)
x = tf.keras.layers.Dense(512,activation="relu")(x)
x = tf.keras.layers.Dense(256, activation = "relu")(x) x =
tf.keras.layers.Dense(128,activation="relu")(x)
x = tf.keras.layers.Dropout(0.2)(x)
prediction = tf.keras.layers.Dense(4, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=prediction)

from tensorflow.keras.optimizers import RMSprop
model.compile(optimizer=RMSprop(learning_rate=0.002),
loss="categorical_crossentropy", metrics=["accuracy"])

history2 = model.fit(train_dataset,
epochs=20 , batch_size=64,
validation_data = val_dataset)

import os os.chdir(r'/kaggle/working')
from IPython.display import FileLink FileLink(r'densenet.h5')
FileLink(r'inceptionv3.h5') FileLink(r'resnet152.h5')
FileLink(r'EfficientNetB0.h5')

import numpy as np
import os
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from keras.applications.resnet import ResNet50,ResNet152
from keras.applications.inception_v3 import InceptionV3
from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Sequential, Model, load_model
from tensorflow.keras.layers import GlobalAveragePooling2D,AveragePooling2D,
Dense, Dropout, BatchNormalization, Input, Conv2D, MaxPooling2D, Dense,
GlobalAveragePooling2D, Activation, Dropout, Flatten, Dense, Input
PATH = '/kaggle/input/corn-leaf/data'
CATEGORIES = os.listdir(PATH)
NUM_CLASSES = len(CATEGORIES)
print(CATEGORIES, NUM_CLASSES)
SIZE = 224
BATCH_SIZE = 32
EPOCHS = 12
train_ds = tf.keras.preprocessing.image_dataset_from_directory(
PATH,
validation_split=0.2,
subset="training",
seed=1337,
image_size=(SIZE,SIZE),
batch_size=BATCH_SIZE,
class_names=CATEGORIES,
label_mode='categorical',
)
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
PATH,
validation_split=0.2,
subset="validation",
seed=1337,
image_size=(SIZE,SIZE),
batch_size=BATCH_SIZE,
class_names=CATEGORIES,
label_mode='categorical',
)
import matplotlib.pyplot as plt
['Common_Rust', 'Blight', 'Healthy', 'Gray_Leaf_Spot'] 4
Found 4188 files belonging to 4 classes.
Using 3351 files for training.
Found 4188 files belonging to 4 classes.
Using 837 files for validation.
plt.figure(figsize=(15, 10))
for images, labels in train_ds.take(1):
for i in range(10):
ax = plt.subplot(3, 5, i + 1)
plt.imshow(images[i].numpy().astype("uint8"))
plt.title(CATEGORIES[np.argmax(labels[i])])
plt.axis("off")
DATA PREPROCESSING
MAKE IMAGES INTO CSV
import os
from PIL import Image
test_dic = {"Blight":[],"Common_Rust":[],"Gray_Leaf_Spot":[],"Healthy":[]}
x = ["Blight","Common_Rust","Gray_Leaf_Spot","Healthy"]
for i in x:
count = 0
for dirname, _, filenames in os.walk('/kaggle/input/corn-leaf/data'):
if i in dirname:
for filename in filenames:
count += 1
x= os.path.join(dirname, filename)
im = Image.open(x)
im = im.resize((96,96))
pix_val = list(im.getdata())
pix_val_flat = [x for sets in pix_val for x in sets]
test_dic[i].append(pix_val_flat)
print(count)
print(len(test_dic["Healthy"]))
import pandas as pd
1146
1306
574
1162
1162
```

```
import pandas as pd
import csv
## BLIGHT
df = pd.DataFrame([test_dic['Blight'][0]])
df.to_csv("try0.csv" , index = False)
with open("./try0.csv", "a") as f:
writer = csv.writer(f)
for i in range(1, len(test_dic['Blight'])):
writer.writerow(test_dic['Blight'][i])
df = pd.read_csv("./try0.csv", on_bad_lines='skip')
df["class"] = 0
print(df.shape)
## COMMON_RUST
df1 = pd.DataFrame([test_dic['Common_Rust'][0]])
df1.to_csv("try1.csv" , index = False)
with open("./try1.csv", "a") as f:
writer = csv.writer(f)
for i in range(1, len(test_dic['Common_Rust'])):
writer.writerow(test_dic['Common_Rust'][i])
df1 = pd.read_csv("./try1.csv", on_bad_lines='skip')
df1["class"] = 1
print(df1.shape)
## GRAY LEAF SPOT
df2 = pd.DataFrame([test_dic['Gray_Leaf_Spot'][0]])
df2.to_csv("try2.csv" , index = False)
with open("./try2.csv", "a") as f:
writer = csv.writer(f)
for i in range(1, len(test_dic['Gray_Leaf_Spot'])):
writer.writerow(test_dic['Gray_Leaf_Spot'][i])
df2 = pd.read_csv("./try2.csv", on_bad_lines='skip')
df2["class"] = 2
print(df2.shape)
##HEALTHY
df3 = pd.DataFrame([test_dic['Healthy'][0]])
df3.to_csv("try3.csv" , index = False)
with open("./try3.csv", "a") as f:
writer = csv.writer(f)
for i in range(1, len(test_dic['Healthy'])):
writer.writerow(test_dic['Healthy'][i])
df3 = pd.read_csv("./try3.csv", on_bad_lines='skip')
```

```
df3["class"] = 3
print(df3.shape)
In [14]:
test_df = pd.concat([df, df1, df2, df3], ignore_index=True)
test_df.shape
In [ ]:
test_df.to_csv("datacsv.csv" , index = False)
In [ ]:
(1146, 27649)
(1303, 27649)
(574, 27649)
(1162, 27649)
Out[14]:
(4185, 27649)
In [ ]:
test_df = pd.read_csv("/kaggle/input/datacsv/data.csv")
In [15]:
test_df["class"].value_counts()
In [16]:
Label = test_df["class"]
Data = test_df.drop(columns=["class"])
from imblearn.over_sampling import RandomOverSampler
oversample = RandomOverSampler()
Data,Label = oversample.fit_resample(Data,Label)
Data = np.array(Data).reshape(-1,96,96,3)
print('Shape of Data :',Data.shape)
In [18]:
Label.value_counts()
In [19]:
classes = {0 : "Blight",
1: "Common_Rust",
2 : "Gray_Leaf_Spot",
3 : "Healthy"}
Splitting the data into training set and testing set
In [20]:
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test = train_test_split(Data , Label , test_size =
0.20 ,
random_state = 49)
In [21]:
```

```
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
Out[15]:
1 1303
3 1162
0 1146
2 574
Name: class, dtype: int64
Shape of Data : (5212, 96, 96, 3)
Out[18]:
0 1303
1 1303
2 1303
3 1303
Name: class, dtype: int6
In [22]:
from keras.callbacks import ReduceLROnPlateau
learning_rate_reduction = ReduceLROnPlateau(monitor='val_accuracy'
, patience = 2
, verbose=1
,factor=0.75
, min_lr=0.00001)
In [23]:
import matplotlib.pyplot as plt
def draw_Accuracy_plots(history1):
plt.figure(figsize=(14,5))
plt.subplot(1,2,2)
plt.plot(history1.history['accuracy'])
plt.plot(history1.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend(['train', 'test'], loc='upper left')
plt.subplot(1,2,1)
plt.plot(history1.history['loss'])
plt.plot(history1.history['val_loss'])
plt.title('model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
In [24]:
import tensorflow as tf
from tensorflow.keras.applications.efficientnet import preprocess_input as
efficientnet_p
reprocess_input
from tensorflow.keras.applications import DenseNet121
input_shape1 = (96, 96, 3)
input_shape2 = (96, 96, 3)
base_model1 = tf.keras.applications.EfficientNetB0(include_top=False,
weights='imagenet'
, input_shape=input_shape1)
for layer in base_model1.layers:
layer.trainable = False
x1 = base_model1.output
x1 = tf.keras.layers.GlobalAveragePooling2D()(x1)
x1 = Dense(1024, activation = "relu")(x1)
base_model2 = DenseNet121(include_top=False, weights='imagenet',
input_shape=input_shape
2)
for layer in base_model2.layers:
layer.trainable = False
x2 = base_model2.output
x2 = tf.keras.layers.GlobalAveragePooling2D()(x2)
x2 = Dense(1024, activation = "relu")(x2)
concat = tf.keras.layers.Concatenate()([x1, x2])
hidden1 = tf.keras.layers.Dense(1024, activation='relu')(concat)
dropout = tf.keras.layers.Dropout(0.2)
hidden2 = Dense(512, activation = "relu")(hidden1)
hidden2 = Dropout(0.2)(hidden2)
hidden3 = Dense(256, activation = "relu")(hidden2)
output = tf.keras.layers.Dense(4, activation='softmax')(hidden3)
model = tf.keras.Model(inputs=[base_model1.input, base_model2.input],
outputs=output)
model.compile(optimizer='Adam',
loss='categorical_crossentropy',
metrics=['accuracy'])
In [25]:
reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy'
, patience = 2
, verbose=1
,factor=0.75
, min_lr=0.00001)
tf.config.run_functions_eagerly(True)
history = model.fit(
[X_train, X_train],
y_train,
epochs = 20,
batch_size = 64,
validation_split = 0.2,
callbacks =[reduce_lr]
)

import tensorflow as tf
tf.__version__

from sklearn.metrics import classification_report
y_pred = model.predict([X_test,X_test]).round()
target_names = [f"{classes[i]}" for i in range(4)]
print(classification_report(y_test, y_pred , target_names =target_names ))

from tensorflow.keras.models import load_model
model0 = load_model("/kaggle/input/model-resnet/resnet152.h5")
model1 = load_model("/kaggle/input/models-corn/models/densenet.h5")
model2 = load_model("/kaggle/input/models-corn/models/inceptionv3.h5")
model3 = load_model("/kaggle/input/models-corn/models/proposed_model.h5")
model4 = load_model("/kaggle/input/models-corn/models/proposed_model_with_vgg19.
h5")
from keras.applications.resnet import ResNet50,ResNet152
from keras.applications.inception_v3 import InceptionV3
from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Sequential, Model, load_model
from tensorflow.keras.layers import GlobalAveragePooling2D,AveragePooling2D, Den
se, Dropout, BatchNormalization, Input, Conv2D, MaxPooling2D, Dense, GlobalAvera
gePooling2D, Activation, Dropout, Flatten, Dense, Input
train_data_generator = ImageDataGenerator(rotation_range=40,
 rescale=1./255,
 shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True)
valid_data_generator = ImageDataGenerator(rescale = 1./255)

train_dataset = train_data_generator.flow_from_directory(
 "/kaggle/input/aug-data/aug_data/Train",
target_size = (224, 224),
 class_mode = "categorical",
 batch_size = 64)
val_dataset = train_data_generator.flow_from_directory(
 "/kaggle/input/aug-data/aug_data/valid",
target_size = (224, 224),
 class_mode = "categorical",
 batch_size = 64)
 import tensorflow as tf
def predict(model, img):
 img_array = tf.keras.preprocessing.image.img_to_array(img)
 img_array = tf.expand_dims(img_array, 0)
 predictions = model.predict(img_array)
 predicted_class = class_names[np.argmax(predictions[0])]
 confidence = round(100 * (np.max(predictions[0])), 2)
 return predicted_class, confidence
import matplotlib.pyplot as plt
import numpy as np
class_names = ["Gray_spot","Common_Rust","Northern_leaf_blight" ,"Healthy"]
plt.figure(figsize=(15, 15))
for images, labels in val_dataset:
 for i in range(9):
 ax = plt.subplot(3, 3, i + 1)
 plt.imshow(images[i])
 predicted_class, confidence = predict(model0, images[i])
 index = labels[i].argmax(axis=0)
 actual_class = class_names[index]
 plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence:
{confidence}%")
 plt.axis("off")
 break
import matplotlib.pyplot as plt
import numpy as np
```

```
class_names = ["Gray_spot","Common_Rust","Northern_leaf_blight" ,"Healthy"]
plt.figure(figsize=(15, 15))
for images, labels in val_dataset:
 for i in range(9):
 ax = plt.subplot(3, 3, i + 1)
 plt.imshow(images[i])
 print(type(images[i]))

 predicted_class, confidence = predict(model1, images[i])
 index = labels[i].argmax(axis=0)
 actual_class = class_names[index]

 plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence:
{confidence}%")

 plt.axis("off")
 break
import matplotlib.pyplot as plt
import numpy as np
class_names = ["Gray_spot","Common_Rust","Northern_leaf_blight" ,"Healthy"]
plt.figure(figsize=(15, 15))
for images, labels in val_dataset:
 for i in range(9):
 ax = plt.subplot(3, 3, i + 1)
 plt.imshow(images[i])

 predicted_class, confidence = predict(model2, images[i])
 index = labels[i].argmax(axis=0)
 actual_class = class_names[index]

 plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence:
{confidence}%")

 plt.axis("off")
 break
from PIL import Image
class_names2 =["Blight", "Common_rust", "Gray_leaf_spot", "healthy"]
path = "/kaggle/input/corn-leaf/data/Common_Rust/Corn_Common_Rust (100).JPG"
im = Image.open(path)
im = im.resize((96,96))
```

```
pix_val = list(im.getdata())
pix_val_flat = [x for sets in pix_val for x in sets]
Data = np.array(pix_val_flat).reshape(-1,96,96,3)
ans = model3.predict([Data, Data])[0]
index = ans.argmax(axis = 0)
print(class_names2[index])
print("CONFIDENCE =",max(ans)*100, "%")
from PIL import Image
class_names2 =["Northern_Blight", "Common_rust", "Gray_leaf_spot", "healthy"]
path = "/kaggle/input/aug-
data/aug_data/valid/Corn_(maize)___Northern_Leaf_Blight/00a14441-7a62-4034-bc40-
b196aeab2785___RS_NLB 3932.JPG"
im = Image.open(path)
im = im.resize((96,96))
pix_val = list(im.getdata())
pix_val_flat = [x for sets in pix_val for x in sets]
Data = np.array(pix_val_flat).reshape(-1,96,96,3)
ans = model4.predict([Data, Data])[0]
index = ans.argmax(axis = 0)
print(class_names2[index])
print("CONFIDENCE =",max(ans)*100, "%")
import tensorflow as tf
def predict(model, img):
 img_array = tf.keras.preprocessing.image.img_to_array(img)
 print(len(img_array))
 img_array = tf.expand_dims(img_array, 0)
 predictions = model.predict(img_array)
 predicted_class = class_names[np.argmax(predictions[0])]
 confidence = round(100 * (np.max(predictions[0])), 2)
 return predicted_class, confidence
 import matplotlib.pyplot as plt
import numpy as np
class_names = ["Gray_spot","Common_Rust","Northern_leaf_blight" ,"Healthy"]
plt.figure(figsize=(15, 15))
for images, labels in val_dataset:
 for i in range(3):
 ax = plt.subplot(3, 3, i + 1)
 print(type(images[i]))
 plt.imshow(images[i])
 predicted_class, confidence = predict(model0, images[i])
 index = labels[i].argmax(axis=0)
 actual_class = class_names[index]
 plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n Confidence:
{confidence}%")
 plt.axis("off")
 break
from PIL import Image
class_names =["Gray_Spot", "Common_Rust", "Northern_Leaf_Blight", "Healthy"]
path = "/kaggle/input/corn-leaf-with-train-test/data/Train/Gray_Leaf_Spot/Corn_G
ray_Spot (154).JPG"
im = Image.open(path)
im = im.resize((224,224))
im = np.asarray(im)
im = im/255
ans,confidence = predict(model0,im)
print(ans)
print("CONFIDENCE =",confidence, "%")
```

4.2: Potato Leaf disease

```
from tensorflow.keras.layers import Dense,Activation,Flatten,Dropout, Dot,
RepeatVector, ConvLSTM1D, Reshape, MaxPooling3D

from tensorflow.keras.layers import Conv2D,MaxPooling2D ,Conv3D,
BatchNormalization,Permute, Concatenate import os
from PIL import Image import pandas as pd import csv
test_dic =
{"Potato    Late_blight":[],"Potato Early_blight":[],"Potato    healthy":[]}

x = ["Potato    Late_blight","Potato    Early_blight","Potato   healthy"]

for i in x: count = 0
for dirname, _, filenames in os.walk('/kaggle/input'): if "Test" in dirname:
if i in dirname:

for filename in filenames: count += 1
x= os.path.join(dirname, filename) im = Image.open(x)
im = im.resize((128,128)) pix_val = list(im.getdata())
pix_val_flat = [x for sets in pix_val for x in sets]
test_dic[i].append(pix_val_flat)
print(count)

df = pd.DataFrame([test_dic['Potato healthy'][0]])

df.to_csv("try0.csv" , index = False) with open("./try0.csv", "a") as f:
writer = csv.writer(f)

for i in range(1, len(test_dic['Potato  healthy'])):

#print(len(test_dic['Potato healthy'][i]))

writer.writerow(test_dic['Potato    healthy'][i])

df2 = pd.read_csv("./try0.csv") df2["class"] = 0 print(df2.shape)
df = pd.DataFrame([test_dic['Potato Late_blight'][0]])

df.to_csv("try1.csv" , index = False) with open("./try1.csv", "a") as f:
writer = csv.writer(f)
```

```
for i in range(1, len(test_dic['Potato  Late_blight'])):

writer.writerow(test_dic['Potato    Late_blight'][i])

df3 = pd.read_csv("./try1.csv") df3["class"] = 1 print(df3.shape)
df = pd.DataFrame([test_dic['Potato Early_blight'][0]])

df.to_csv("try2.csv" , index = False)

with open("./try2.csv", "a") as f: writer = csv.writer(f)
for i in range(1, len(test_dic['Potato  Early_blight'])):

writer.writerow(test_dic['Potato    Early_blight'][i])

df4 = pd.read_csv("./try2.csv") df4["class"] = 2 print(df4.shape)
test_df = pd.concat([df2, df3, df4], ignore_index=True) test_df.shape
train_dic =
{"Potato    Late_blight":[],"Potato Early_blight":[],"Potato    healthy":[]}

x = ["Potato    Late_blight","Potato    Early_blight","Potato   healthy"]

for i in x: count = 0
for dirname, _, filenames in os.walk('/kaggle/input'): if "Train" in dirname:
if i in dirname:

for filename in filenames: count += 1
x= os.path.join(dirname, filename) im = Image.open(x)
im = im.resize((128,128)) pix_val = list(im.getdata())
pix_val_flat = [x for sets in pix_val for x in sets]

train_dic[i].append(pix_val_flat) print(count)
df = pd.DataFrame([train_dic['Potato    healthy'][0]])

df.to_csv("tryt0.csv" , index = False) df.head()
with open("./tryt0.csv", "a") as f: writer = csv.writer(f)
for i in range(1, len(train_dic["Potato healthy"])):

#print(len(test_dic["Potato healthy"][i]))
```

```
writer.writerow(train_dic["Potato   healthy"][i])

healthy_train = pd.read_csv("./tryt0.csv") healthy_train["class"] = 0
healthy_train.shape
df = pd.DataFrame([train_dic['Potato    Late_blight'][0]])

df.to_csv("tryt1.csv" , index = False) with open("./tryt1.csv", "a") as f:
writer = csv.writer(f) for i in range(1, 300):
#print(len(test_dic["Potato healthy"][i]))

writer.writerow(train_dic["Potato   Late_blight"][i])

df6 = pd.read_csv("./tryt1.csv") df6["class"] = 1 df6.to_csv("late_blight0.csv",
index = False)

print(df6.shape)

df = pd.DataFrame([train_dic['Potato    Late_blight'][0]])

df.to_csv("tryt1.csv" , index = False) with open("./tryt1.csv", "a") as f:
writer = csv.writer(f) for i in range(300, 600):
#print(len(test_dic["Potato healthy"][i]))

writer.writerow(train_dic["Potato   Late_blight"][i])

df8 = pd.read_csv("./tryt1.csv") df8["class"] = 1 df8.to_csv("late_blight2.csv",
index = False) print(df8.shape)
df = pd.DataFrame([train_dic['Potato    Late_blight'][0]])

df.to_csv("tryt1.csv" , index = False) df.head()
with open("./tryt1.csv", "a") as f: writer = csv.writer(f)
for i in range(600, 800):

#print(len(test_dic["Potato healthy"][i]))

writer.writerow(train_dic["Potato   Late_blight"][i])

df19 = pd.read_csv("./tryt1.csv") df19["class"] = 1
df19.to_csv("late_blight4.csv", index = False)
```

```
print(df19.shape)

late_blight_df = pd.concat([df6, df8,df19], ignore_index=True)
late_blight_df.shape
df = pd.DataFrame([train_dic['Potato    Early_blight'][0]])

df.to_csv("tryt2.csv" , index = False) df.head()
with open("./tryt2.csv", "a") as f: writer = csv.writer(f)
for i in range(1, 400):

#print(len(test_dic["Potato healthy"][i]))

writer.writerow(train_dic["Potato   Early_blight"][i])

df10 = pd.read_csv("./tryt2.csv") df10["class"] = 2
df10.to_csv("early_blight0.csv", index = False) df10.shape
df = pd.DataFrame([train_dic['Potato    Early_blight'][0]])

df.to_csv("tryt2.csv" , index = False) df.head()
with open("./tryt2.csv", "a") as f: writer = csv.writer(f)
for i in range(400, 800):

#print(len(test_dic["Potato healthy"][i]))

writer.writerow(train_dic["Potato   Early_blight"][i])

df12 = pd.read_csv("./tryt2.csv") df12["class"] = 2
df12.to_csv("early_blight2.csv", index = False) df12.shape
early_blight_df = pd.concat([df10, df12], ignore_index=True) early_blight_df.shape
final_df = pd.concat([healthy_train, early_blight_df , test_df, late_blight_df],
ignore_index = True)
final_df.shape final_df["class"].value_counts() Label = final_df["class"]
Data = final_df.drop(columns=["class"]) oversample = RandomOverSampler()
Data,Label = oversample.fit_resample(Data,Label) Data = np.array(Data).reshape(-
1,128,128,3) print('Shape of Data :',Data.shape) Label.value_counts()
classes = {0 : "healthy", 1: "late_blight",
2 : "early_blight"}

X_train , X_test , y_train , y_test = train_test_split(Data , Label , test_size =
0.14 , random_state = 49)
```

```
from tensorflow.keras.utils import to_categorical y_train =
to_categorical(y_train)

y_test = to_categorical(y_test)

from keras.callbacks import ReduceLROnPlateau learning_rate_reduction =
ReduceLROnPlateau(monitor='val_accuracy'
, patience = 2

, verbose=1

,factor=0.75

, min_lr=0.00001) datagen = ImageDataGenerator(rescale=(1./255),
horizontal_flip = True

,rotation_range=10

,zoom_range = 0.1

,width_shift_range=0.1

,height_shift_range=0.1) testgen = ImageDataGenerator(rescale=(1./255)) #vgg16
base_model = tf.keras.applications.VGG16(input_shape=(128,128,3),

include_top=False, weights='imagenet')
for layer in base_model.layers: layer.trainable=False
x = tf.keras.layers.Flatten()(base_model.output)

x = tf.keras.layers.Dense(512,activation="relu")(x) x =
tf.keras.layers.Dense(128,activation="relu")(x)

x = tf.keras.layers.Dropout(0.2)(x)

prediction = tf.keras.layers.Dense(3, activation='softmax')(x) model =
Model(inputs=base_model.input, outputs=prediction)
model.compile(optimizer=RMSprop(learning_rate=0.001),
loss="categorical_crossentropy", metrics=["accuracy"])
history1 = model.fit(X_train ,
y_train , epochs=20 , batch_size=64,
validation_split = 0.33, callbacks=[learning_rate_reduction])
y_pred = model.predict(X_test).round() target_names = [f"{classes[i]}" for i in
range(3)]
print(classification_report(y_test, y_pred , target_names =target_names ))
plt.figure(figsize=(14,5))
plt.subplot(1,2,2) plt.plot(history1.history['accuracy'])
plt.plot(history1.history['val_accuracy']) plt.title('Model Accuracy')
plt.xlabel('Epochs') plt.ylabel('Accuracy')
plt.legend(['train', 'test'], loc='upper left')



plt.subplot(1,2,1) plt.plot(history1.history['loss'])
plt.plot(history1.history['val_loss']) plt.title('model Loss')
plt.xlabel('Epochs') plt.ylabel('Loss')
plt.legend(['train', 'test'], loc='upper left') plt.show()
model.save("vgg16.h5")

base_model = tf.keras.applications.VGG19(input_shape=(128,128,3),

include_top=False, weights='imagenet')
for layer in base_model.layers: layer.trainable=False
x = tf.keras.layers.Flatten()(base_model.output)

x = tf.keras.layers.Dense(512,activation="relu")(x) x =
tf.keras.layers.Dense(128,activation="relu")(x) x =
tf.keras.layers.Dropout(0.2)(x)
prediction = tf.keras.layers.Dense(3, activation='softmax')(x) model2 =
Model(inputs=base_model.input, outputs=prediction) from
tensorflow.keras.optimizers import RMSprop
model2.compile(optimizer=RMSprop(learning_rate=0.001),

loss="categorical_crossentropy", metrics=["accuracy"])
history2 = model2.fit(X_train , y_train , epochs=20 , batch_size=64,
validation_split = 0.33, callbacks=[learning_rate_reduction])
y_pred = model2.predict(X_test).round() target_names = [f"{classes[i]}" for i in
range(3)]
print(classification_report(y_test, y_pred , target_names =target_names ))
plt.figure(figsize=(14,5))
plt.subplot(1,2,2) plt.plot(history2.history['accuracy'])
plt.plot(history2.history['val_accuracy']) plt.title('Model Accuracy')
plt.xlabel('Epochs') plt.ylabel('Accuracy')
plt.legend(['train', 'test'], loc='upper left') plt.subplot(1,2,1)
plt.plot(history2.history['loss']) plt.plot(history2.history['val_loss'])
plt.title('model Loss')

plt.xlabel('Epochs') plt.ylabel('Loss')
plt.legend(['train', 'test'], loc='upper left') plt.show()
model2.save("vgg19.h5")

base_model = tf.keras.applications.ResNet50(input_shape=(128,128,3),

include_top=False, weights='imagenet')
for layer in base_model.layers: layer.trainable=False
x = tf.keras.layers.Flatten()(base_model.output)

x = tf.keras.layers.Dense(512,activation="relu")(x) x =
tf.keras.layers.Dense(128,activation="relu")(x) x =
tf.keras.layers.Dropout(0.2)(x)
prediction = tf.keras.layers.Dense(3, activation='softmax')(x) model3 =
Model(inputs=base_model.input, outputs=prediction) from
tensorflow.keras.optimizers import RMSprop
model3.compile(optimizer=RMSprop(learning_rate=0.001),
loss="categorical_crossentropy", metrics=["accuracy"])
history3 = model3.fit(X_train , y_train , epochs=20 ,

batch_size=64, validation_split = 0.33,
callbacks=[learning_rate_reduction]) y_pred = model3.predict(X_test).round()
target_names = [f"{classes[i]}" for i in range(3)]
print(classification_report(y_test, y_pred , target_names =target_names ))
plt.figure(figsize=(14,5))
plt.subplot(1,2,2) plt.plot(history3.history['accuracy'])
plt.plot(history3.history['val_accuracy']) plt.title('Model Accuracy')
plt.xlabel('Epochs') plt.ylabel('Accuracy')
plt.legend(['train', 'test'], loc='upper left') plt.subplot(1,2,1)
plt.plot(history3.history['loss']) plt.plot(history3.history['val_loss'])
plt.title('model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend(['train', 'test'], loc='upper left') plt.show()
base_model = tf.keras.applications.MobileNet(input_shape=(128,128,3),

include_top=False, weights='imagenet')


for layer in base_model.layers: layer.trainable=False
x = tf.keras.layers.Flatten()(base_model.output)

x = tf.keras.layers.Dense(512,activation="relu")(x) x =
tf.keras.layers.Dense(128,activation="relu")(x) x =
tf.keras.layers.Dropout(0.2)(x)
prediction = tf.keras.layers.Dense(3, activation='softmax')(x) model4 =
Model(inputs=base_model.input, outputs=prediction) from
tensorflow.keras.optimizers import RMSprop
model4.compile(optimizer=RMSprop(learning_rate=0.001),
loss="categorical_crossentropy", metrics=["accuracy"])
history4 = model4.fit(X_train , y_train , epochs=20 , batch_size=64,
validation_split = 0.33, callbacks=[learning_rate_reduction])
y_pred = model4.predict(X_test).round() target_names = [f"{classes[i]}" for i in
range(3)]

print(classification_report(y_test, y_pred , target_names =target_names ))
plt.figure(figsize=(14,5))
plt.subplot(1,2,2) plt.plot(history4.history['accuracy'])
plt.plot(history4.history['val_accuracy']) plt.title('Model Accuracy')
plt.xlabel('Epochs') plt.ylabel('Accuracy')
plt.legend(['train', 'test'], loc='upper left')



plt.subplot(1,2,1) plt.plot(history4.history['loss'])
plt.plot(history4.history['val_loss']) plt.title('model Loss')
plt.xlabel('Epochs') plt.ylabel('Loss')
plt.legend(['train', 'test'], loc='upper left') plt.show()
model4.save("mobilenet.h5") classes = {0 : "Healthy",
1: "Late Blight",

2 : "Early Blight"}
```

```
x =
"../input/potatofinal/potato/Test/Potato    healthy/5fcbde8f-52af-4963-b324-
ff7f4dd6bd 4c   RS_HL 1762.JPG"

im = Image.open(x)

im = im.resize((128,128)) pix_val = list(im.getdata())
pix_val_flat = [x for sets in pix_val for x in sets] Data =
np.array(pix_val_flat).reshape(-1,128,128,3) print('Shape of Data :',Data.shape)
print("PREDICTION\n\n")
ans = model3.predict(Data)[0] print("CONFIDENCE =",max(ans)*100, "%") index =
ans.argmax(axis = 0) print(classes[index])

'''
```

APP.PY

```
'''
import streamlit as st
import pickle
import pandas as pd
import numpy as np
import plotly.express as px
from PIL import Image
from tensorflow.keras.models import load_model

new_title = '<h1 style="font-family:sans-serif; color:NAVY; font-size: 50px;
align ="right">Potato Leaf Disease Classification</h1>'
st.markdown(new_title, unsafe_allow_html=True)

st.sidebar.subheader("SELECT A MODEL OF YOUR CHOICE")
x = st.sidebar.selectbox(label = 'MODEL',options =
["ResNet152","DenseNet","InceptionV3","EfficientNetB0","Proposed Model"])
st.write("\n\n\n\n\n")
m = '<h2 style="font-family:sans-serif; color: BROWN; font-size: 30px; align
="right">INTRODUCTION</h2>'
st.markdown(m , unsafe_allow_html = True)
st.markdown("The two most common leaf diseases found in potatoes are early
blight and late blight. A leaf that has been infected has no cure and the
disease is contagious across the plant, so the only option is to remove the
infected leaf.\n ")
```

```
m2 = '<h2 style="font-family:sans-serif; color: BROWN; font-size: 30px; align
="right">SAMPLE DATASET</h2>'
st.markdown(m2 , unsafe_allow_html = True)
imgsd = Image.open("sampledataset.png")
st.image(imgsd)
st.markdown("The dataset for the potato leaf diseases was obtained from the
PlantVillage dataset. The datasets include over 50,000 images of 14 crops such
as potatoes, grapes, tomato apples etc.")
st.markdown("From the 14 classes of data we have focused only the 3 Potato
leaf classes.")

m2 = '<h2 style="font-family:sans-serif; color: BROWN; font-size: 30px; align
="right">DATA PREPROCESSING</h2>'
st.markdown(m2 , unsafe_allow_html = True)
st.markdown("To balance the data RandomOverSampler was used and to avoid
overfitting of data data augmentation using Image Data Generator was used")
imgsd = Image.open("dataprep3.png")
st.image(imgsd)

m2 = '<h2 style="font-family:sans-serif; color: BLACK; font-size: 20px; align
="right">In the project 5 different models were compared namely
ResNet152,DenseNet,InceptionV3, EfficientNetB0 and a MULTI-INPUT Proposed
Model</h2>'
st.markdown(m2 , unsafe_allow_html = True)

if x == "ResNet152":

    m3 = '<h2 style="font-family:sans-serif; color: BROWN; font-size: 30px;
align ="right">MODEL DETAILS</h2>'
    st.markdown(m3 , unsafe_allow_html = True)
    if st.button("Click here to see model details"):
        df = pd.read_csv("classification_report_vgg16.csv")
        img1 = Image.open("Images/resnet_model.png")
        img2 = Image.open("Images/resnet152.jpg")
        st.subheader("Model architecture")
        st.image(img1)
        st.subheader("Model accuracy plots")
        st.image(img2)
        st.subheader("Classification Report")
        st.write(df)

if x == "DenseNet":
    m3 = '<h2 style="font-family:sans-serif; color: BROWN; font-size: 30px;
align ="right">MODEL DETAILS</h2>'
    st.markdown(m3 , unsafe_allow_html = True)
    if st.button("Click here to see model details"):
        df = pd.read_csv("classification_report_vgg19.csv")
        img1 = Image.open("Images/Densenet121_model.png")
        img2 = Image.open("Images/denset121.jpg")
        st.subheader("Model architecture")
        st.image(img1)
        st.subheader("Model accuracy plots")
        st.image(img2)
        st.subheader("Classification Report")
        st.write(df)

if x == "InceptionV3":
  #insert model comaprison df
    m3 = '<h2 style="font-family:sans-serif; color: BROWN; font-size: 30px;
align ="right">MODEL DETAILS</h2>'
    st.markdown(m3 , unsafe_allow_html = True)
    if st.button("Click here to see model details"):
        df = pd.read_csv("classification_report_resnet.csv")
        img1 = Image.open("Images/inceptionV3_model.png")
        img2 = Image.open("Images/inceptionv3.jpg")
        st.subheader("Model architecture")
        st.image(img1)
        st.subheader("Model accuracy plots")
        st.image(img2)
        st.subheader("Classification Report")
        st.write(df)

if x == "EfficientNetB0":
    m3 = '<h2 style="font-family:sans-serif; color: BROWN; font-size: 30px;
align ="right">MODEL DETAILS</h2>'
```

```
    st.markdown(m3 , unsafe_allow_html = True)
    if st.button("Click here to see model details"):
        df = pd.read_csv("classification_report_mobilenet.csv")
        img1 = Image.open("Images/EfficientNetB0_model.png")
        img2 = Image.open("Images/resnet152.jpg")
        st.subheader("Model architecture")
        st.image(img1)
        st.subheader("Model accuracy plots")
        st.image(img2)
        st.subheader("Classification Report")
        st.write(df)

if x == "Proposed Model":
    m3 = '<h2 style="font-family:sans-serif; color: BROWN; font-size: 30px;
align ="right">MODEL DETAILS</h2>'
    st.markdown(m3 , unsafe_allow_html = True)
    if st.button("Click here to see model details"):
        df = pd.read_csv("classification_report_mobilenet.csv")
        img1 = Image.open("Images/proposed_model_model.png")
        img2 = Image.open("images/proposed_model.jpg")
        st.subheader("Model architecture")
        st.image(img1)
        st.subheader("Model accuracy plots")
        st.image(img2)
        st.subheader("Classification Report")
        st.write(df)

classes = {0 : "Healthy",
           1: "Late Blight",
           2 : "Early Blight"}

m2 = '<h2 style="font-family:sans-serif; color: BROWN; font-size: 30px; align
="right">PREDICTION:</h2>'
st.markdown(m2 , unsafe_allow_html = True)
if x == "ResNet152":
    img = st.file_uploader(label = "")
    if st.button("Predict"):
        if img is not None:
            im = Image.open(img)
            im = im.resize((128,128))
            pix_val = list(im.getdata())
            pix_val_flat = [x for sets in pix_val for x in sets]
            Data = np.array(pix_val_flat).reshape(-1,128,128,3)
            model3 = load_model("vgg16.h5")
            ans = model3.predict(Data)[0]
            st.write("CONFIDENCE =",max(ans)*100, "%")
            index = ans.argmax(axis = 0)
            st.write(classes[index])
if x == "DenseNet":
    img = st.file_uploader(label = "")
    if st.button("Predict"):
        if img is not None:
            im = Image.open(img)
            im = im.resize((128,128))
            pix_val = list(im.getdata())
            pix_val_flat = [x for sets in pix_val for x in sets]
            Data = np.array(pix_val_flat).reshape(-1,128,128,3)
            model3 = load_model("vgg19.h5")
            ans = model3.predict(Data)[0]
            st.write("CONFIDENCE =",max(ans)*100, "%")
            index = ans.argmax(axis = 0)
            st.write(classes[index])
if x == "InceptionV3":
    img = st.file_uploader(label = "")
    if st.button("Predict"):
        if img is not None:
            im = Image.open(img)
            im = im.resize((128,128))
            pix_val = list(im.getdata())
            pix_val_flat = [x for sets in pix_val for x in sets]
            Data = np.array(pix_val_flat).reshape(-1,128,128,3)
            model3 = load_model("resnet50.h5")
            ans = model3.predict(Data)[0]
            st.write("CONFIDENCE =",max(ans)*100, "%")
            index = ans.argmax(axis = 0)
            st.write(classes[index])
```

```
if x == "EfficientNetB0":
    img = st.file_uploader(label = "")
    if st.button("Predict"):
        if img is not None:
            im = Image.open(img)
            im = im.resize((128,128))
            pix_val = list(im.getdata())
            pix_val_flat = [x for sets in pix_val for x in sets]
            Data = np.array(pix_val_flat).reshape(-1,128,128,3)
            model3 = load_model("mobilenet.h5")
            ans = model3.predict(Data)[0]
            st.write("CONFIDENCE =",max(ans)*100, "%")
            index = ans.argmax(axis = 0)
            st.write(classes[index])

if x == "Proposed Model":
    img = st.file_uploader(label = "")
    if st.button("Predict"):
        if img is not None:
            im = Image.open(img)
            im = im.resize((128,128))
            pix_val = list(im.getdata())
            pix_val_flat = [x for sets in pix_val for x in sets]
            Data = np.array(pix_val_flat).reshape(-1,128,128,3)
            model3 = load_model("mobilenet.h5")
            ans = model3.predict(Data)[0]
            st.write("CONFIDENCE =",max(ans)*100, "%")
            index = ans.argmax(axis = 0)
            st.write(classes[index])



st.sidebar.markdown("Developed by: ")
st.sidebar.markdown("Rakshit Acharya 123003203 ")
st.sidebar.markdown("Sathwik palakurty 123003175 ")
st.sidebar.markdown("Yashwanth Alapati 123003276 ")
```

CHAPTER 5

RESULTS AND SNAPSHOTS FOR CORN LEAF

5.1 RESNET152:

Validation accuracy :
~95.31%

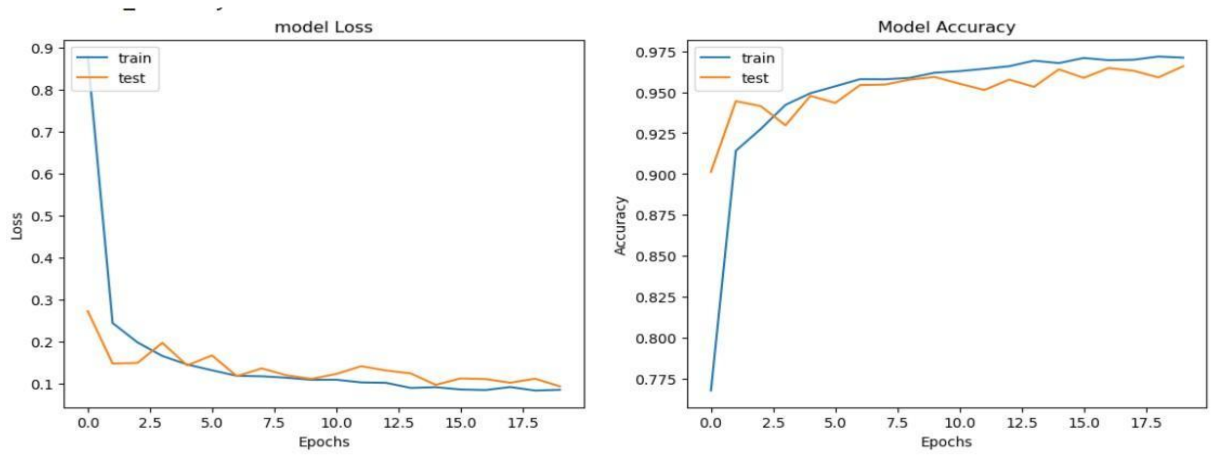
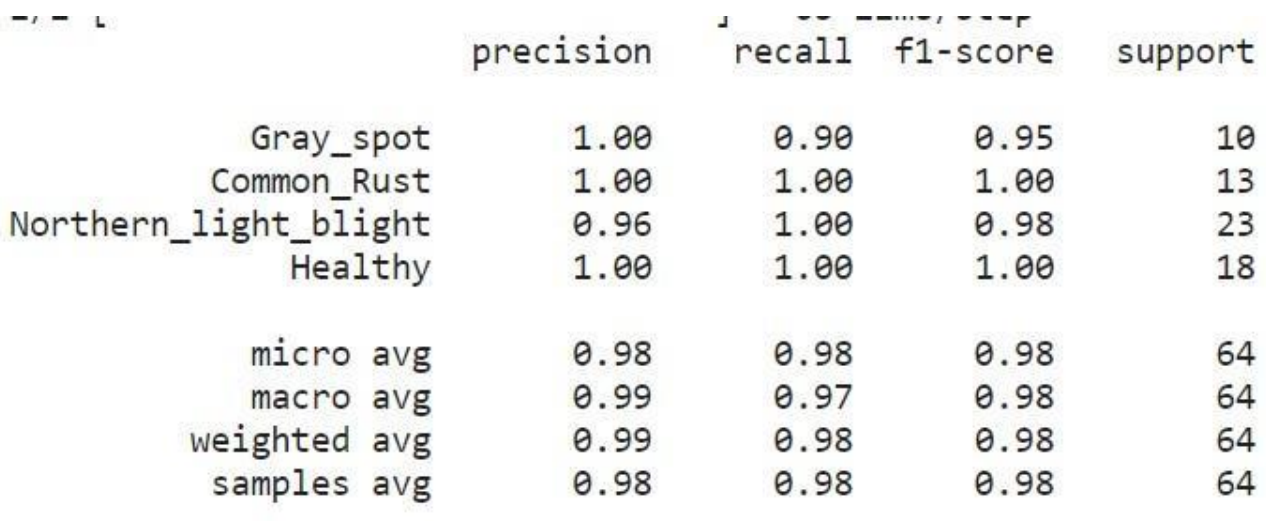


Figure 5.1.1 Accuracy and loss plots of RESNET152



```
                       precision    recall  f1-score   support

           Gray_spot       1.00      0.90      0.95        10
         Common_Rust       1.00      1.00      1.00        13
Northern_light_blight       0.96      1.00      0.98        23
             Healthy       1.00      1.00      1.00        18

           micro avg       0.98      0.98      0.98        64
           macro avg       0.99      0.97      0.98        64
        weighted avg       0.99      0.98      0.98        64
         samples avg       0.98      0.98      0.98        64
```

Figure 5.1.2 Classification Report of RESNET152

5.2 INCEPTION V3:

Validation accuracy :
~97.18%

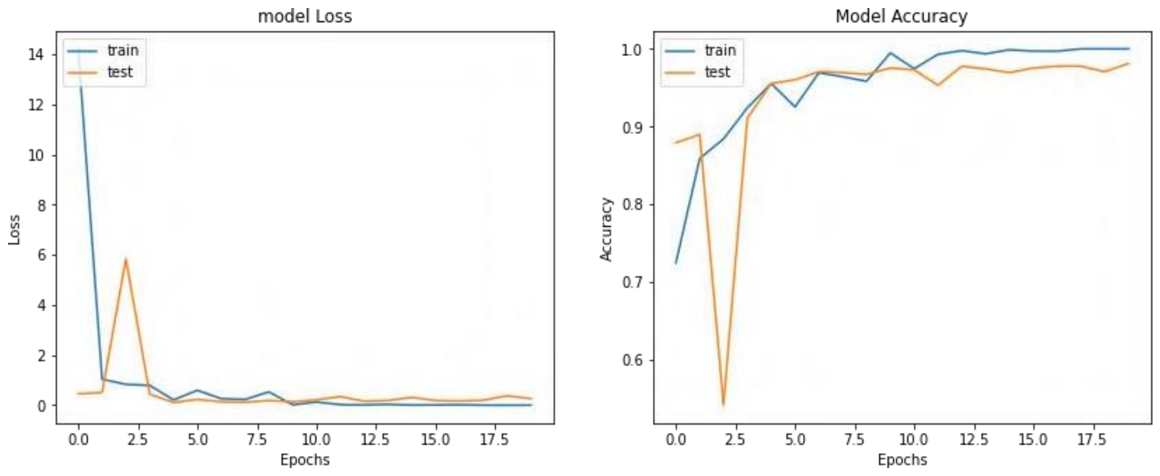
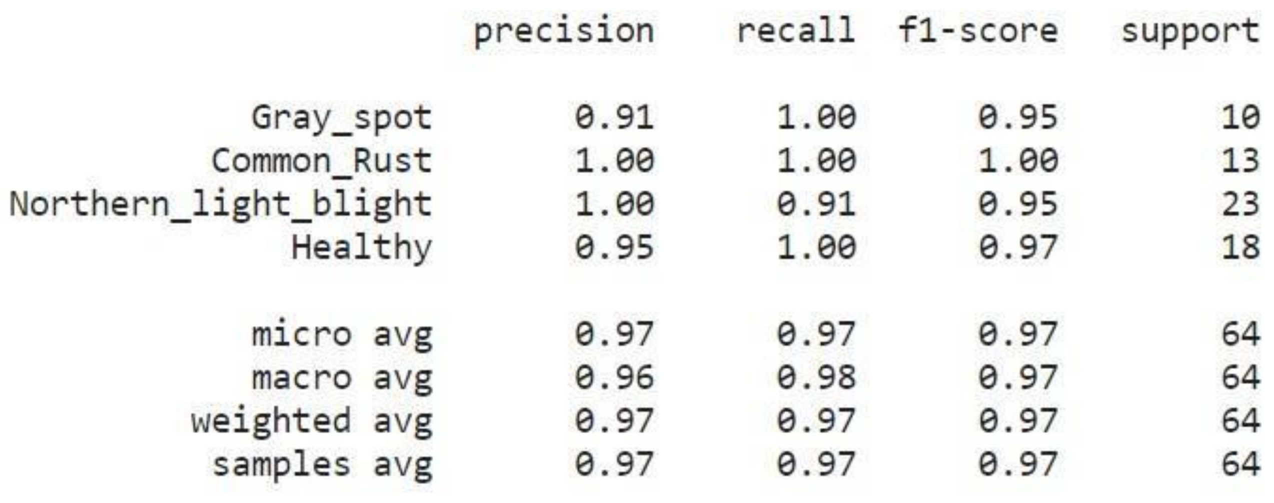


Figure 5.2.1 Accuracy and loss plots of INCEPTION V3



```
                    precision    recall  f1-score   support

          Gray_spot       0.91      1.00      0.95        10
        Common_Rust       1.00      1.00      1.00        13
Northern_light_blight       1.00      0.91      0.95        23
            Healthy       0.95      1.00      0.97        18

          micro avg       0.97      0.97      0.97        64
          macro avg       0.96      0.98      0.97        64
       weighted avg       0.97      0.97      0.97        64
        samples avg       0.97      0.97      0.97        64
```

Figure 5.2.2 Classification report of INCEPTION V3

5.3 DENSENET121 :

Validation accuracy :
~97.79%

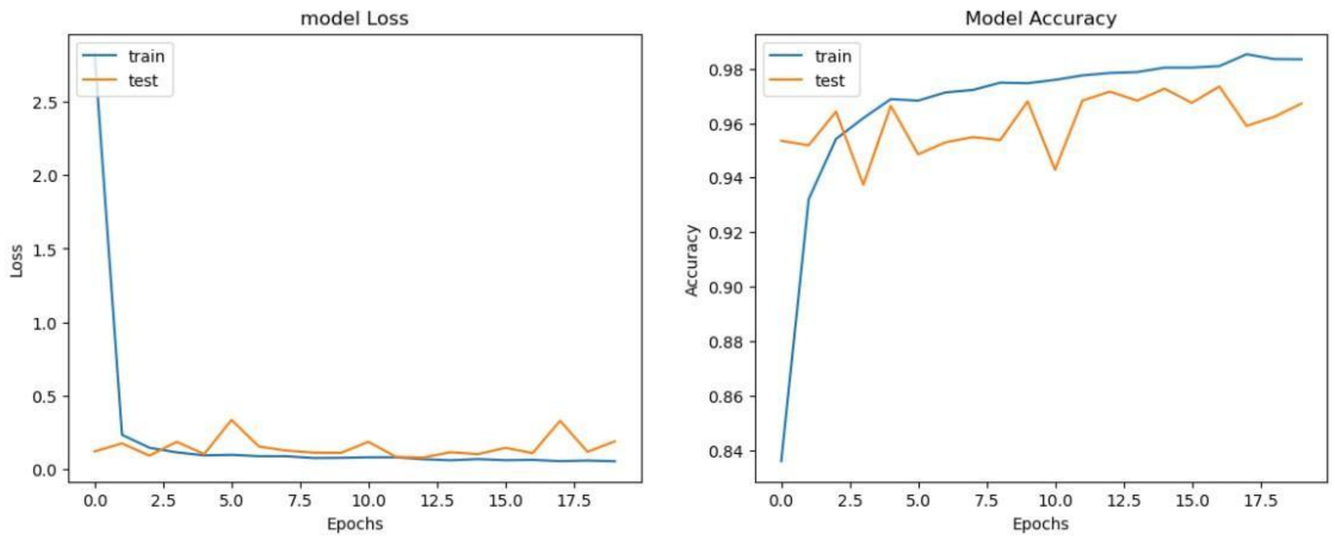
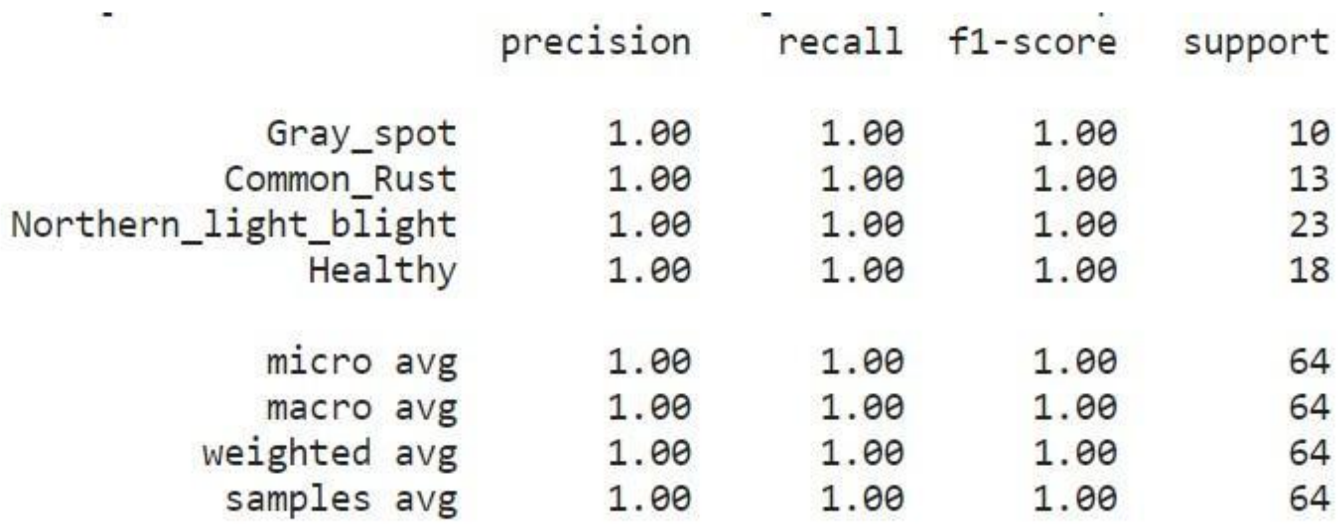


Figure 5.3.1 Accuracy and loss plots of DENSENET121



```
                      precision    recall  f1-score   support

           Gray_spot       1.00      1.00      1.00        10
         Common_Rust       1.00      1.00      1.00        13
Northern_light_blight       1.00      1.00      1.00        23
             Healthy       1.00      1.00      1.00        18

           micro avg       1.00      1.00      1.00        64
           macro avg       1.00      1.00      1.00        64
        weighted avg       1.00      1.00      1.00        64
         samples avg       1.00      1.00      1.00        64
```

Figure 5.3.2 Classification report of DENSENET121

5.4 EFFICIENT B0 :

Validation accuracy :
~97.10%

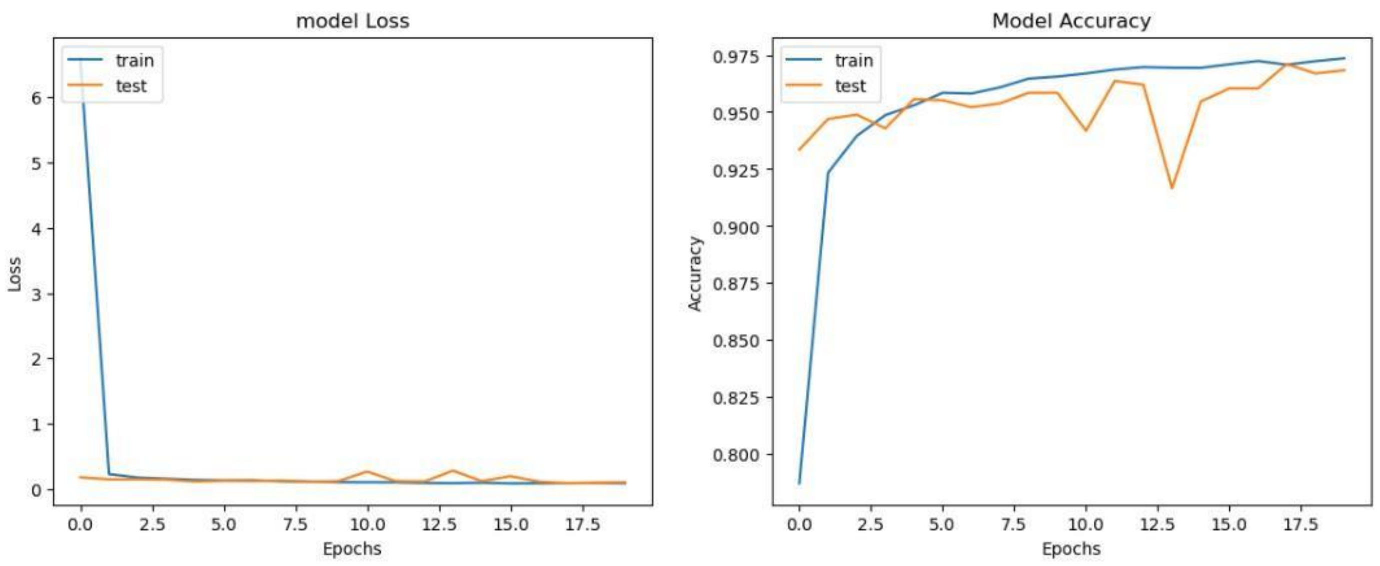
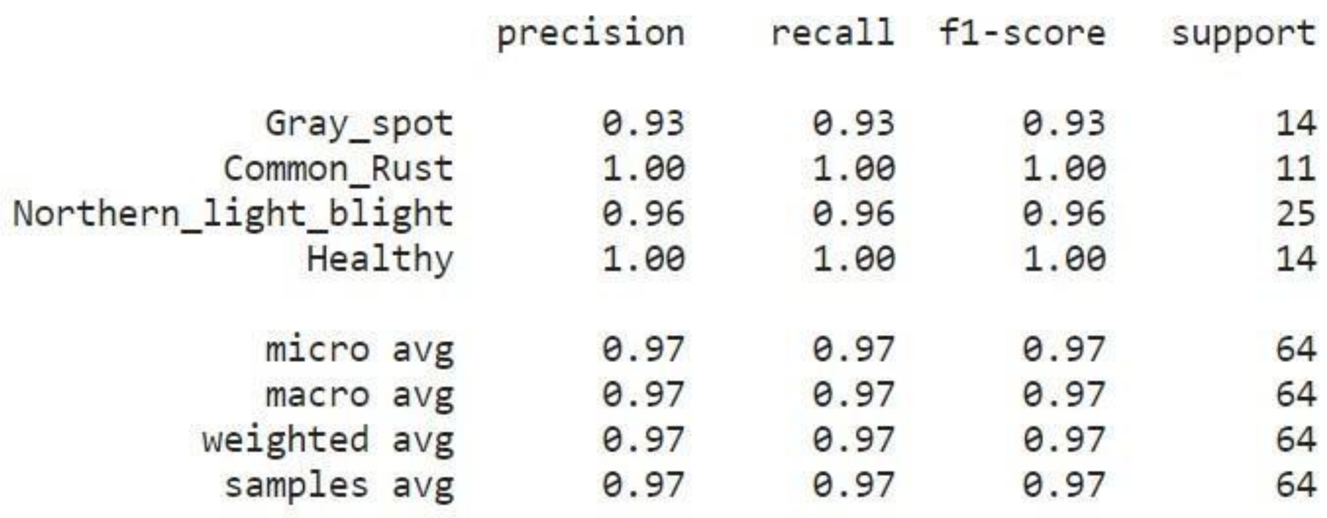


Figure 5.4.1 Accuracy and loss plots of EFFICIENTB0



```
                    precision    recall  f1-score   support

          Gray_spot      0.93      0.93      0.93        14
        Common_Rust      1.00      1.00      1.00        11
Northern_light_blight    0.96      0.96      0.96        25
            Healthy      1.00      1.00      1.00        14

          micro avg      0.97      0.97      0.97        64
          macro avg      0.97      0.97      0.97        64
       weighted avg      0.97      0.97      0.97        64
        samples avg      0.97      0.97      0.97        64
```

Figure 5.4.2 Classification report of EFFICIENT B0

5.5 PROPOSED MODEL(EFFICIENT B0 AND DENSENET121):

Validation Accuracy: 96.16%

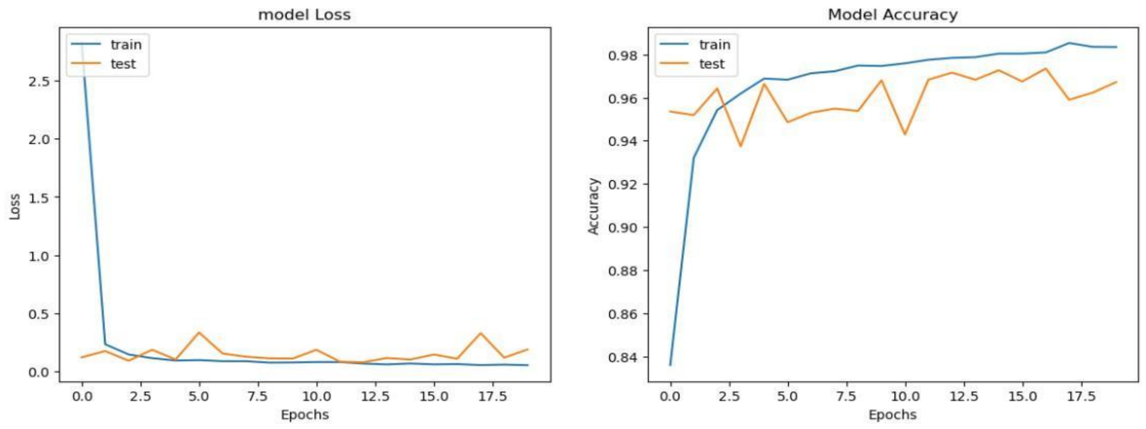
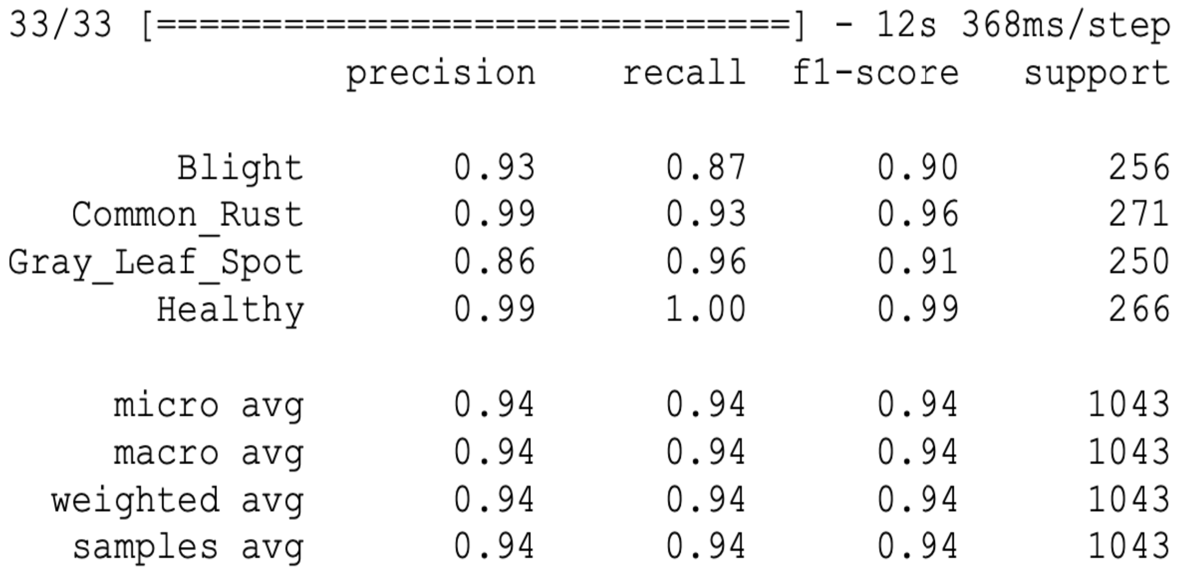


Figure 5.5.1 Accuracy and loss plots PROPOSED MODEL



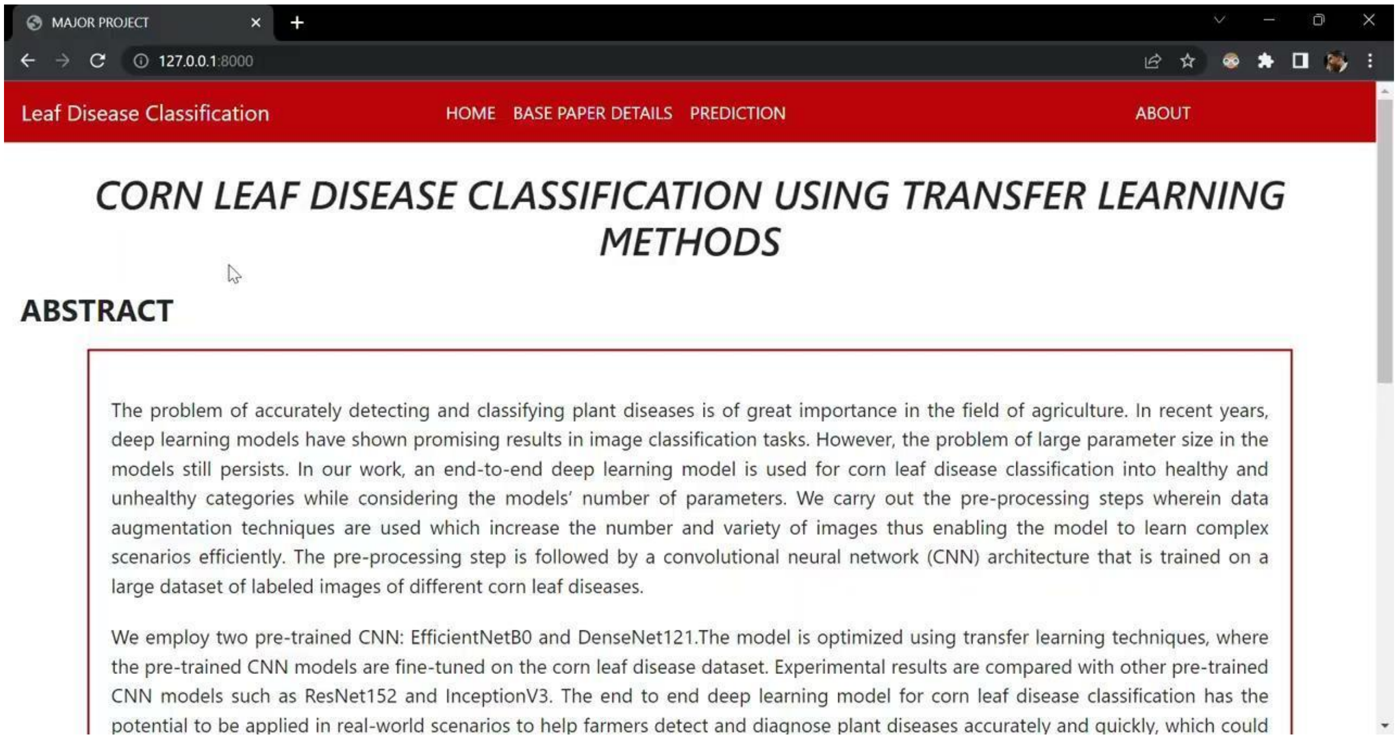
```
33/33 [==============================] - 12s 368ms/step
                precision    recall  f1-score   support

         Blight      0.93      0.87      0.90       256
    Common_Rust      0.99      0.93      0.96       271
 Gray_Leaf_Spot      0.86      0.96      0.91       250
        Healthy      0.99      1.00      0.99       266

      micro avg      0.94      0.94      0.94      1043
      macro avg      0.94      0.94      0.94      1043
   weighted avg      0.94      0.94      0.94      1043
    samples avg      0.94      0.94      0.94      1043
```

Figure 5.5.2 Classification report of PROPOSED MODEL

5.6 GUI OF CORN LEAF:



CORN LEAF DISEASE CLASSIFICATION USING TRANSFER LEARNING METHODS

ABSTRACT

The problem of accurately detecting and classifying plant diseases is of great importance in the field of agriculture. In recent years, deep learning models have shown promising results in image classification tasks. However, the problem of large parameter size in the models still persists. In our work, an end-to-end deep learning model is used for corn leaf disease classification into healthy and unhealthy categories while considering the models' number of parameters. We carry out the pre-processing steps wherein data augmentation techniques are used which increase the number and variety of images thus enabling the model to learn complex scenarios efficiently. The pre-processing step is followed by a convolutional neural network (CNN) architecture that is trained on a large dataset of labeled images of different corn leaf diseases.

We employ two pre-trained CNN: EfficientNetB0 and DenseNet121.The model is optimized using transfer learning techniques, where the pre-trained CNN models are fine-tuned on the corn leaf disease dataset. Experimental results are compared with other pre-trained CNN models such as ResNet152 and InceptionV3. The end to end deep learning model for corn leaf disease classification has the potential to be applied in real-world scenarios to help farmers detect and diagnose plant diseases accurately and quickly, which could

Figure 5.6.1 Graphical user interface

CHAPTER 6

RESULTS AND SNAPSHOTS OF POTATO LEAF

6.1 RESNET 152:

Validation Accuracy: 99.38%

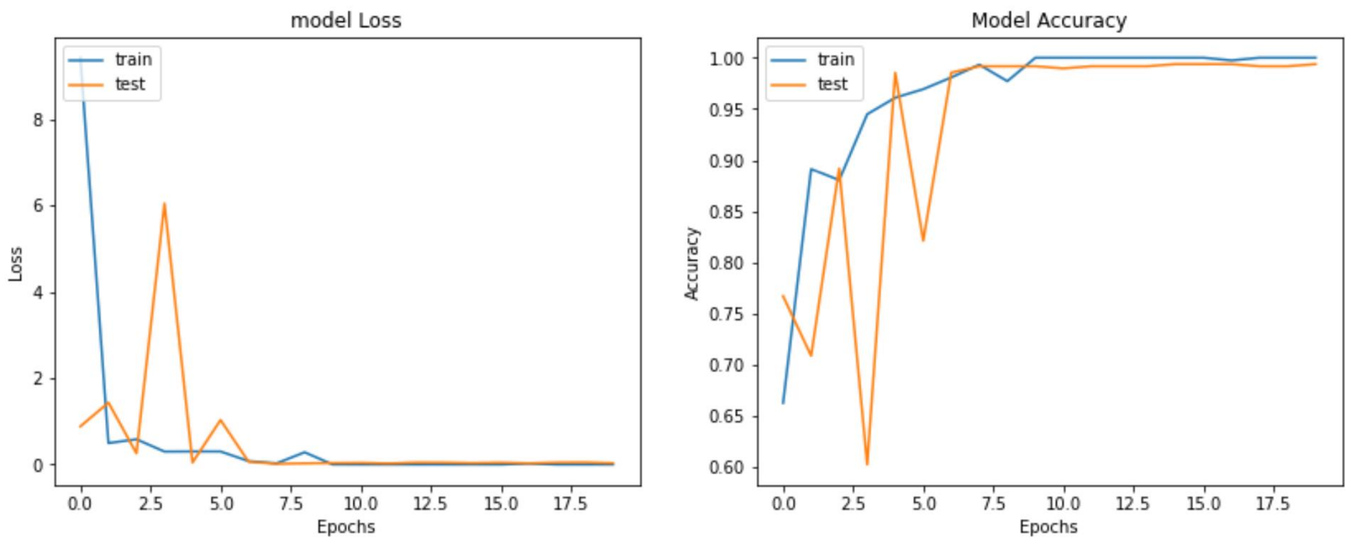
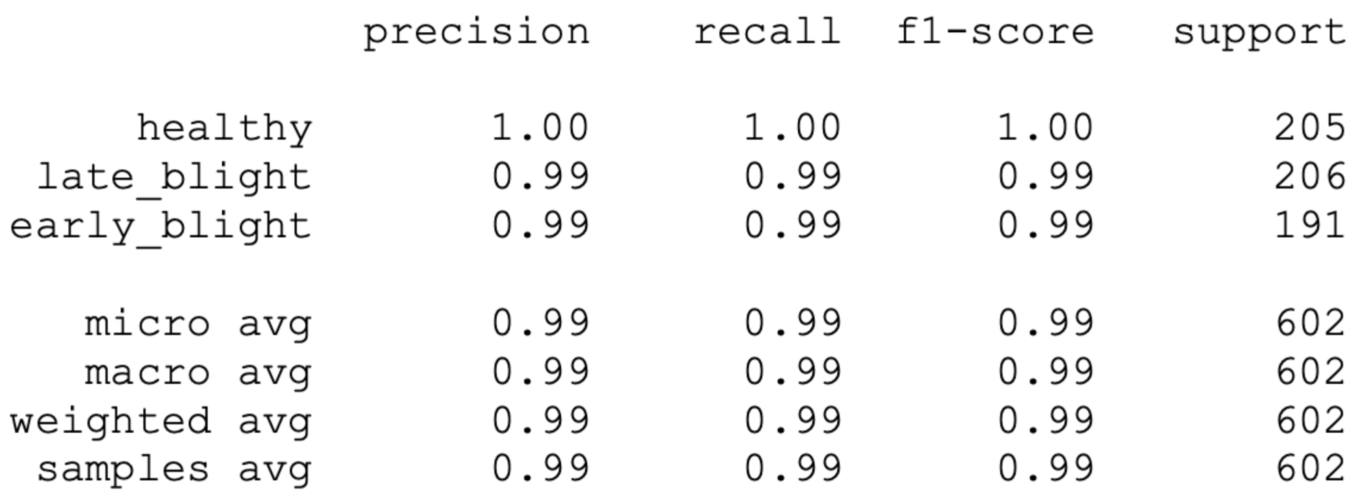


Figure 6.1.1 Accuracy and loss plots RESNET152 model



```
              precision    recall  f1-score   support

     healthy       1.00      1.00      1.00       205
 late_blight       0.99      0.99      0.99       206
early_blight       0.99      0.99      0.99       191

   micro avg       0.99      0.99      0.99       602
   macro avg       0.99      0.99      0.99       602
weighted avg       0.99      0.99      0.99       602
 samples avg       0.99      0.99      0.99       602
```

Figure 6.1.2 Classification report of RESNET152

6.2 DENSENET121:

Validation accuracy: 98.54%

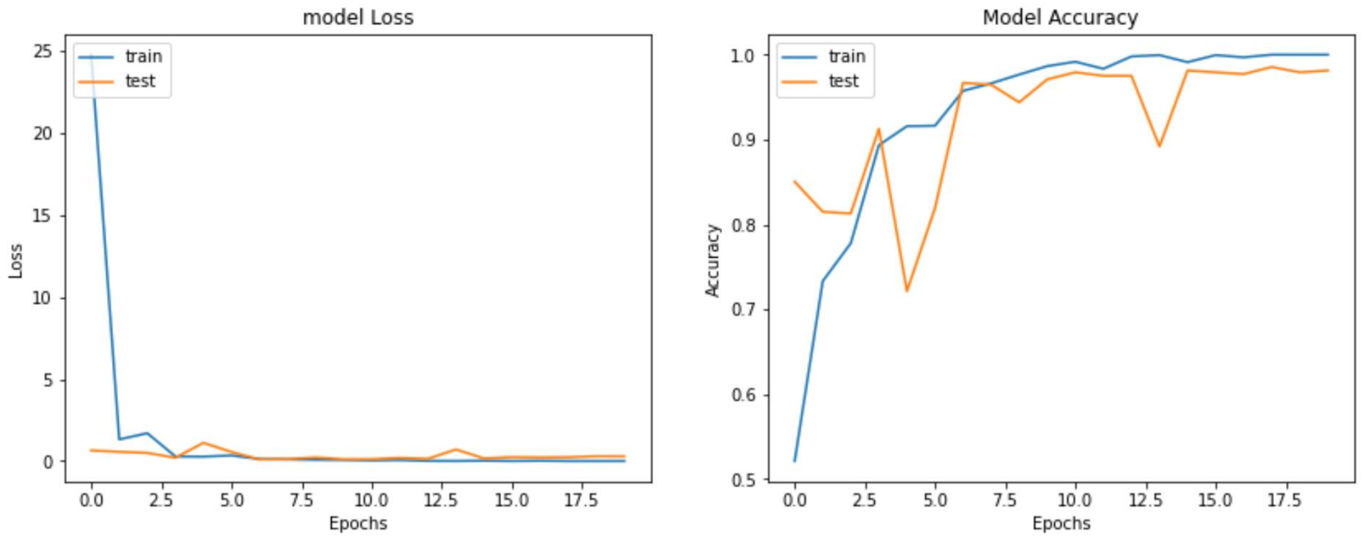
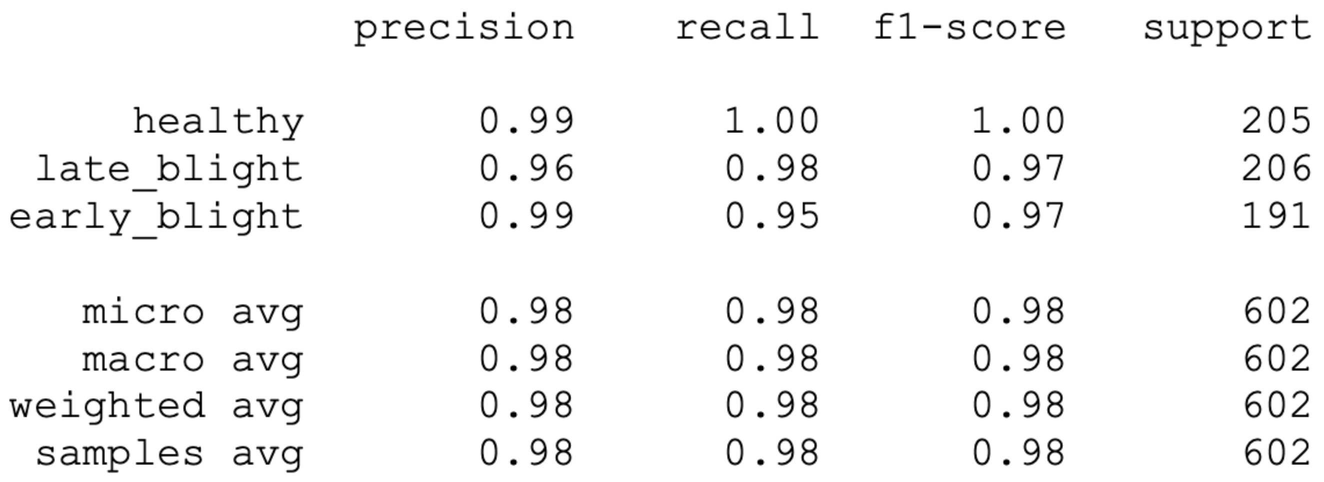


Figure 6.2.1 Accuracy and loss plots DENSENET121 model



```
              precision    recall  f1-score   support

     healthy       0.99      1.00      1.00       205
 late_blight       0.96      0.98      0.97       206
early_blight       0.99      0.95      0.97       191

   micro avg       0.98      0.98      0.98       602
   macro avg       0.98      0.98      0.98       602
weighted avg       0.98      0.98      0.98       602
 samples avg       0.98      0.98      0.98       602
```

Figure 6.2.2 Classification report of DENSENET121

6.3 EFFICIENTNET B0:

Validation accuracy: 99.50%

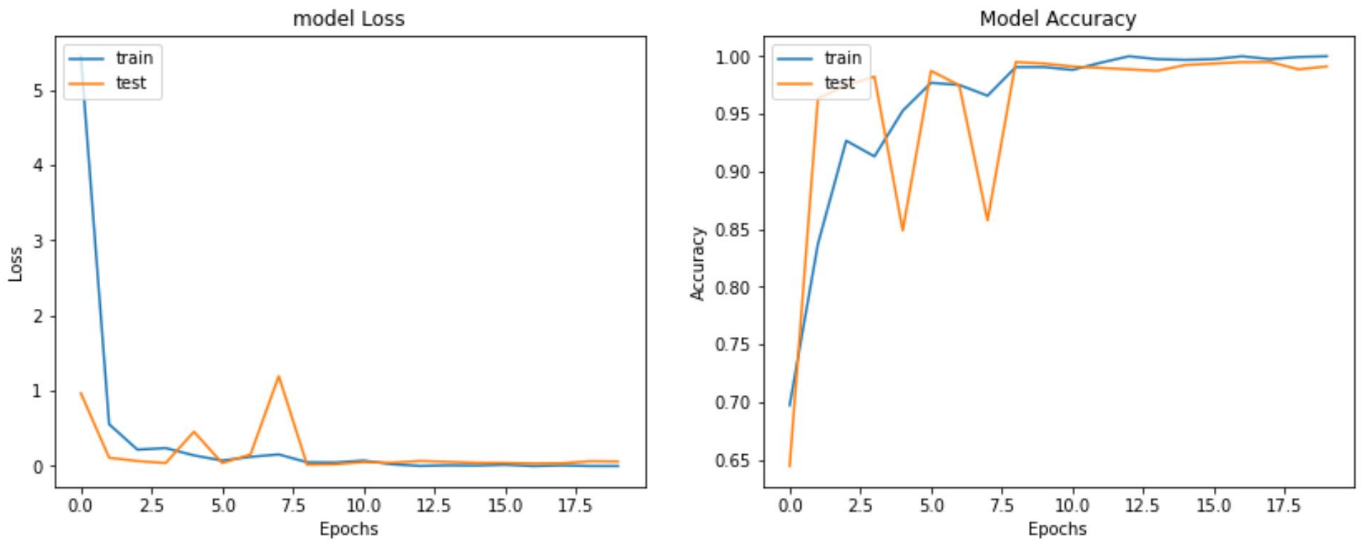
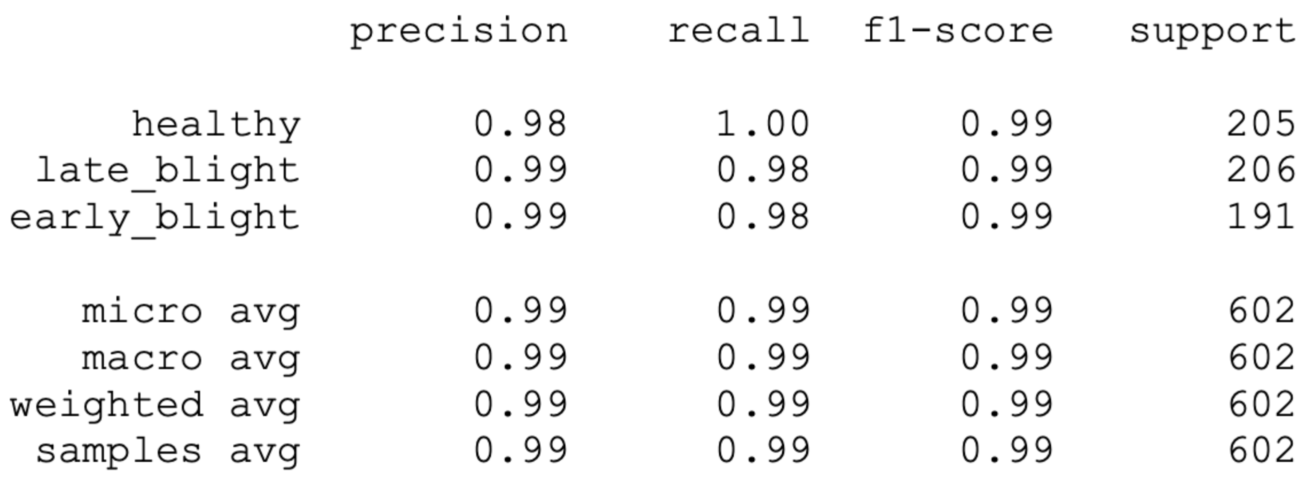


Figure 6.3.1 Accuracy and loss plots EFFICIENTB0 model



```
               precision    recall  f1-score   support

      healthy       0.98      1.00      0.99       205
  late_blight       0.99      0.98      0.99       206
 early_blight       0.99      0.98      0.99       191

    micro avg       0.99      0.99      0.99       602
    macro avg       0.99      0.99      0.99       602
 weighted avg       0.99      0.99      0.99       602
  samples avg       0.99      0.99      0.99       602
```

Figure 6.3.2 Classification report of EFFICIENT B0

6.4 INCEPTION V3

Validation accuracy: 97.30%

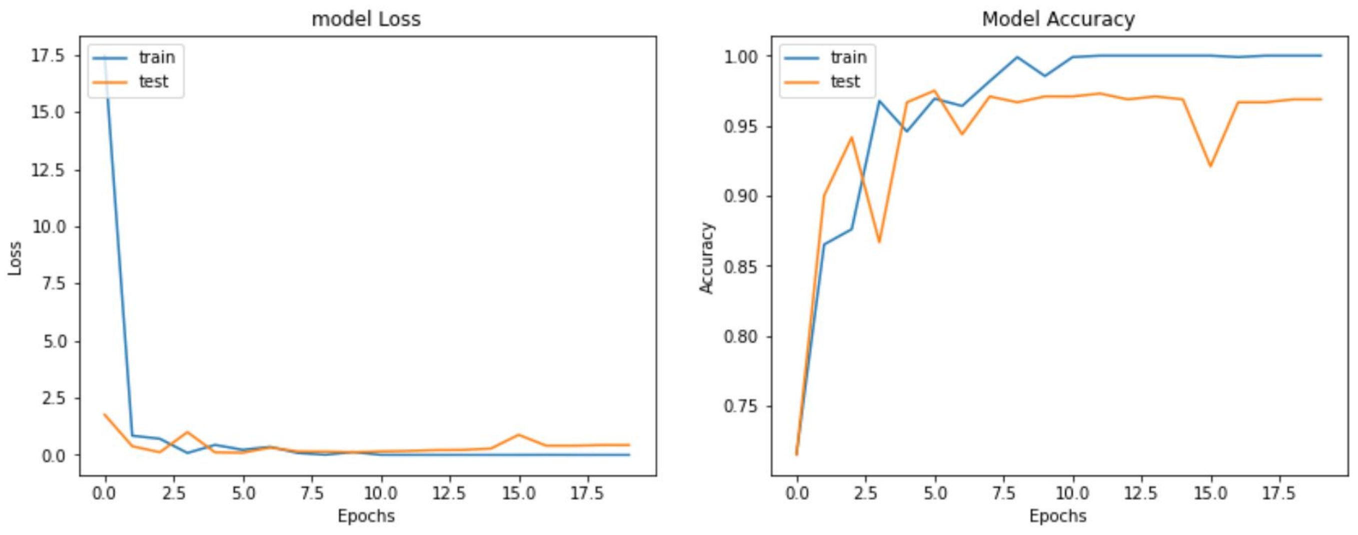
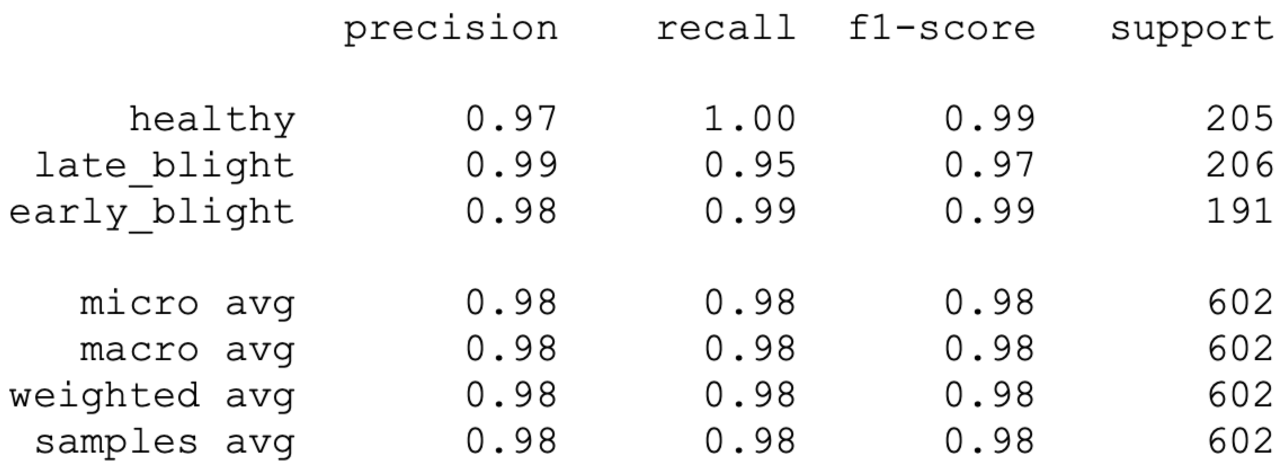


Figure 6.4.1 Accuracy and loss plots INCEPTIONV3 model



```
              precision    recall  f1-score   support

     healthy       0.97      1.00      0.99       205
 late_blight       0.99      0.95      0.97       206
early_blight       0.98      0.99      0.99       191

   micro avg       0.98      0.98      0.98       602
   macro avg       0.98      0.98      0.98       602
weighted avg       0.98      0.98      0.98       602
 samples avg       0.98      0.98      0.98       602
```

Figure 6.4.2 Classification report of INCEPTION V3

6.5 PROPOSED MODEL:

Validation accuracy: 98.13%

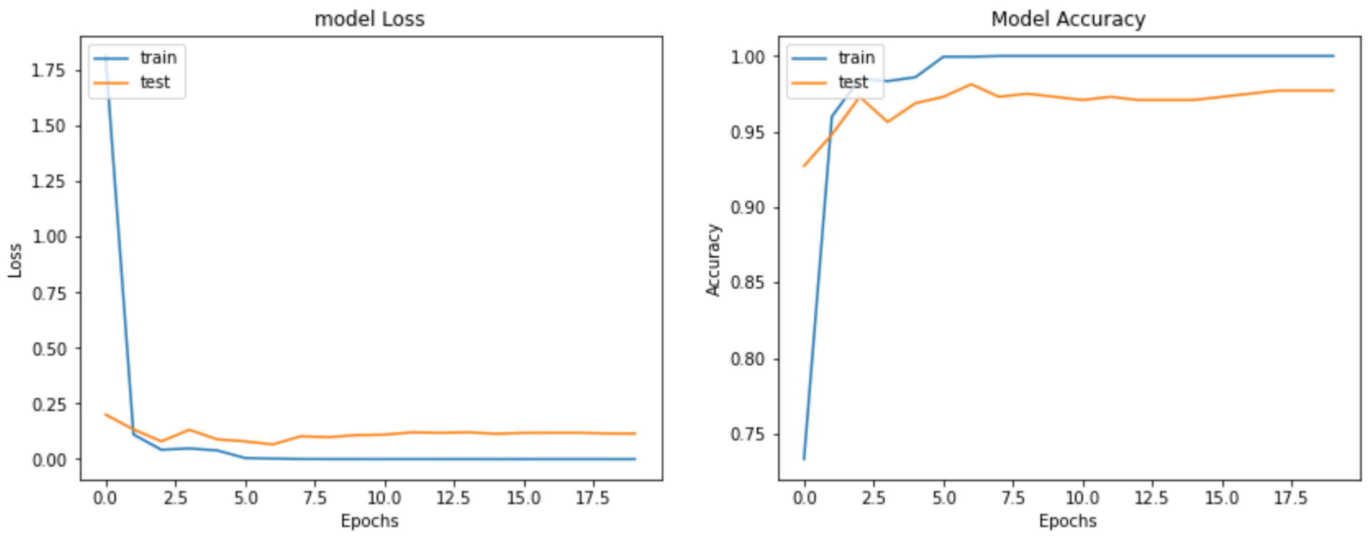
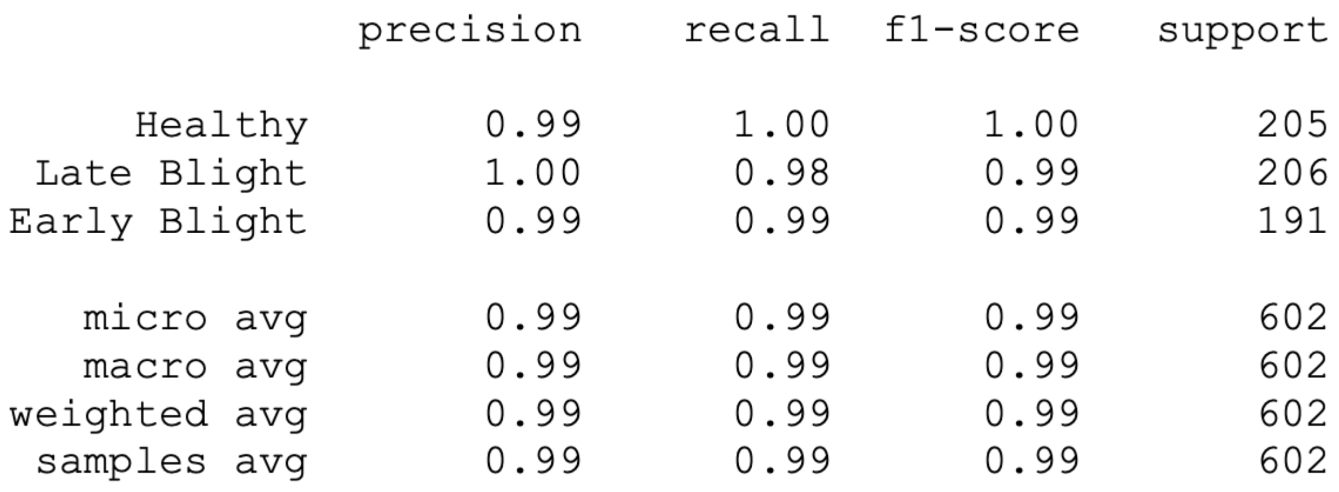


Figure 6.5.1 Accuracy and loss plots PROPOSED MODEL



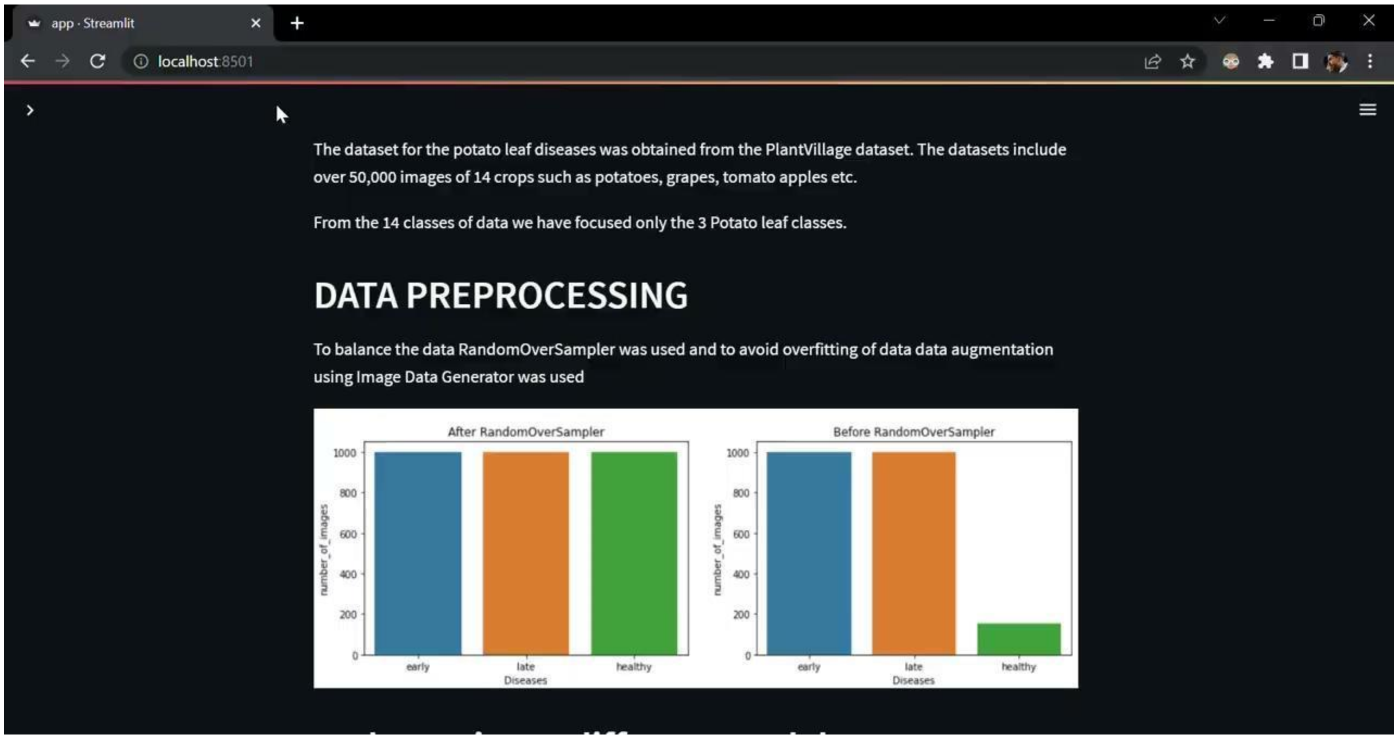
```
              precision    recall  f1-score   support

     Healthy       0.99      1.00      1.00       205
 Late Blight       1.00      0.98      0.99       206
Early Blight       0.99      0.99      0.99       191

   micro avg       0.99      0.99      0.99       602
   macro avg       0.99      0.99      0.99       602
weighted avg       0.99      0.99      0.99       602
 samples avg       0.99      0.99      0.99       602
```

Figure 6.5.2 Classification report of PROPOSED MODEL

6.6 GUI for potato leaf



The dataset for the potato leaf diseases was obtained from the PlantVillage dataset. The datasets include over 50,000 images of 14 crops such as potatoes, grapes, tomato apples etc.

From the 14 classes of data we have focused only the 3 Potato leaf classes.

DATA PREPROCESSING

To balance the data RandomOverSampler was used and to avoid overfitting of data data augmentation using Image Data Generator was used

CHAPTER 7

PROJECT TEAM AND INDIVIDUAL CONTRIBUTION

Register number	Name	Individual Contribution
123003203	Rakshit Acharya	● Data Collection and preprocessing (Week 1 - Week 2) ● Developed an interactive WebApp to implement the blight disease classifier model(Week 7- Week 8) ● Worked on Proposed model for corn and potato leaves.
123003276	A.Yashwanth	● Data Collection and preprocessing (Week 1 - Week 2) ● Worked on two models for Corn leaf. ● Worked on two models for Potato leaf.
123003175	Palakurty S Venkata Sai Sathwik	● Data Collection and preprocessing (Week 1 - Week 2) ● Worked on remaining two models for Corn leaf. ● Worked on remaining two models for Potato leaf.

Table 7.1 Individual Contributions

CHAPTER 8

CONCLUSION AND FUTURE PLANS

In this study, a classification problem is exhibited to detect infected leaves in the earlier stages of the infection so that it can be treated as soon as possible. Considerable data augmentation using ImageDataGenerator and RandomOverSampler is done on the data due to the imbalance in the dataset, and the data is then used to train various transfer learning CNN models, namely RESNET 152, DENSENET 121, EFFICIENTNET B0, INCEPTION V3, PROPOSED MODEL and the results obtained are evaluated and assessed.

Further, with these models, a WebApp GUI has been built to facilitate the use of these models by farmers and other relevant communities. We believe that this work will be a notable contribution to the sector of agriculture. We hope that this work proves useful to the research communities to get familiar with the AI-based detection of potato leaves diseases. The model developed not only works on the benchmark dataset but can also work on real-time images with decent accuracy.

In the future, the same work can be extended to include all classes of plants like tomatoes, bell pepper, etc., and the accuracy of classifying real-time images can be improved.

CHAPTER 9
REFERENCES

- **R. Hu, S. Zhang, P. Wang, G. Xu, D. Wang, and Y. Qian,** ‘‘The identification of corn leaf diseases based on transfer learning and data augmentation,’’ in *Proc. 3rd Int. Conf. Comput. Sci. Softw. Eng.*, May 2020, pp. 58–65.

- **E. L. da Rocha, L. Rodrigues, and J. F. Mari,** ‘‘Maize leaf disease classification using convolutional neural networks and hyperparameter optimization,’’ in *Proc. Anais do XVI Workshop Visão Computacional. (SBC)*, 2020, pp. 104–110.

- **Y. Zhao, L. Liu, C. Xie, R. Wang, F. Wang, Y. Bu, and S. Zhang**, ‘‘An effective automatic system deployed in agricultural Internet of Things using multi-context fusion network towards crop disease recognition in the wild,’’ *Appl. Soft Comput.*, vol. 89, Apr. 2020, Art. no. 106128.

- **F. Saeed, M. A. Khan, M. Sharif, M. Mittal, L. M. Goyal, and S. Roy**, ‘‘Deep neural network features fusion and selection based on PLS regression with an application for crops diseases classification,’’ *Appl. Soft Comput.*, vol. 103, May 2021, Art. no. 107164.