POTATO CROP DISEASE CLASSIFICATION USING TRANSFER LEARNING METHODS

Report submitted to the SASTRA Deemed to be University

as the requirement for the course

CSE300 MINI PROJECT

Submitted by

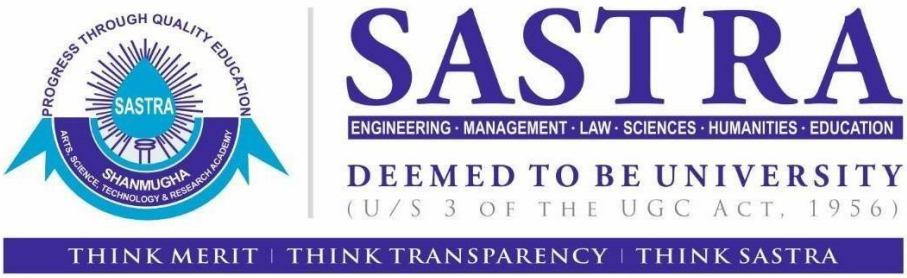
Adhitya Narayan P A

(Reg. No.: 123003004)

Rakshit Acharya

(Reg. No.: 123003203)

JUNE 2022



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

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled "**Potato Crop Disease Classification using Transfer Learning Methods**" submitted as a requirement for the course, CSE300 **Mini-project** for B.Tech. is a bonafide record of the work done by **Mr. Adhitya Narayan P A (Reg. No. 123003004, CSE), Mr. Rakshit Acharya (Reg. No. 123003203, CSE)** during the academic year 2021-22, in the School of Computing, under my supervision.

Signature of Project Supervisor : L Gowri

Name with Affiliation : L. GOWRI, AP-II, SOC

Date : 25/06/2022

Mini-project Viva-voce held on 29/06/2022.

Examiner 1

Examiner 2

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

THINK MERIT THINK TRANSPARENCY THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Declaration

We declare that the report titled "**Potato Crop Disease Classification using Transfer Learning Methods**" submitted by us is an original work done by us under the guidance of, **Mrs. Gowri L, School of Computing, SASTRA Deemed to be University** during the sixth semester of the academic year 2021-22, 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) : 1. P.A. Adhitya Narayan

2. Rakshit

Name of the candidate(s) : **1. Mr. Adhitya Narayan P A**

2. Mr. Rakshit Acharya

Date : **25 June 2022**

ACKNOWLEDGEMENT

First and foremost, I praise and thank God for having made everything possible.

It is a great honour to express my gratefulness to our beloved Vice Chancellor **Prof. Dr. S. VAIDHYASUBRAMANIAM**, SASTRA Deemed to be University, the paramount of the institution, for his patronage and **Dr. R. CHANDRAMOULI**, Registrar, SASTRA Deemed to be University for providing the necessary research facilities in the campus.

I am extremely thankful to **Dr. UMAMAKESWARI A**, Dean, School of Computing, SASTRA Deemed to be University for giving me this opportunity and setting a standard for the project work.

I am extremely grateful to **Dr. SHANKAR SRIRAM,** Associate Dean, School of Computing for his constant support, motivation and academic help extended for the past three years of my life in School of Computing.

I would like to acknowledge with much appreciation the crucial role of my project guide, **Mrs. GOWRI L**, Assistant Professor, for her continuous efforts in guiding me throughout the semester. Her valuable inputs made me keep the progress of the work in track. Her timely help can never be penned with words.

I express my deepest thanks to all the faculty members and technical assistants of the School of Computing, SASTRA Deemed to be University for their timely assistance in the work.

I am grateful to all my well-wishers, who helped me in facing the difficulties in the successful completion of project work.

LIST OF FIGURES

LIST OF TABLES

TABLE OF CONTENTS

SYNOPSIS

Project Team Number: 22SOCU1048

Register No: **Name:**

123003004 Adhitya Narayan P A

Abstract

Potatoes are the 4th staple food consumed in the world. After rice and wheat it is the most cultivated and in-demand crop. With the enhancement in technology and artificial intelligence in diagnosing crop diseases, it has become a relevant and applicable area of research for agricultural development. Potato plants' diseases are mostly categorised into 2 classes namely Early blight and late blight. Late detection and inappropriate classification of diseases can lead to worsening of the plants' health and quality of potatoes. Fortunately, the diseases in potato plants can be found and identified based on the condition of the leaves. Previously, various deep learning models have been proposed for classification of plant diseases. Plant-Village Dataset is used which has 1000 images for both the diseases class, and 152 images for the healthy leaves. A deep learning approach would be proposed to detect the early and late blight diseases in potato plants by analysing the visual representation of various crop leaves images.

Specific Contribution

- Data Pre-processing and Data collection.
- Developed a Responsive Web Page for the Project where users can easily upload the image file and get it classified as Early blight , Late blight or healthy.

Specific Learning

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

Project Team Number: 22SOCU1048

Register No: **Name:**

123003203 Rakshit Acharya

Abstract

Potatoes are the 4th staple food consumed in the world. After rice and wheat it is the most cultivated and in-demand crop. With the enhancement in technology and artificial intelligence in diagnosing crop diseases, it has become a relevant and applicable area of research for agricultural development. Potato plants' diseases are mostly categorised into 2 classes namely Early blight and late blight. Late detection and inappropriate classification of diseases can lead to worsening of the plants' health and quality of potatoes. Fortunately, the diseases in potato plants can be found and identified based on the condition of the leaves. Previously, various deep learning models have been proposed for classification of plant diseases. Plant-Village Dataset is used which has 1000 images for both the diseases class, and 152 images for the healthy leaves. A deep learning approach would be proposed to detect the early and late blight diseases in potato plants by analysing the visual representation of various crop leaves images.

Specific Contribution

- Data Pre-processing - RandomOverSampler (**Novelty** from the base paper).
- Implement the transfer learning techniques as given in the base paper. Compare the models based on evaluation metrics and suggest the best model.

Specific Learning

- Learned working of various transfer learning models (VGG16, VGG19, Resnet50, Mobilenet) and RandomOverSampler technique to balance the data, and basics of streamlit for webapp development.

ABBREVIATIONS

GUI	Graphical User Interface
CNN	Convolutional Neural Network
VGG	Visual Geometry Group
RESNET	Residual Network
WebApp	Web Application

CHAPTER 1

INTRODUCTION

Solanum tuberosum, the commonly cultivated potato crop is one of the most grown agricultural crops in the world with a production of almost 400 million tonnes worldwide every year, out of which China and India alone amount for a whopping 150 million tonnes every year. Given its importance in the global agriculture industry, it becomes relevant to work and research on the various diseases and pathogens it's affected by, and provide a solution. Potato plants are every now and then affected by fungal pathogens which reduce the yield and affect the livelihood of the farmers. Among the diseases, the blight diseases, namely, early blight and late blight, are the two major causes for notable losses in yield.

Late blight is one of the most callous leaves diseases for the crop of potato. It causes spore formation, which is indicated by tiny moist spots on the tips of leaves. The spots enlarge as time progresses, and a yellow/brown pigment formation is seen in low temperatures and moist conditions. Sometimes, a white mould formation occurs at the bottom part of the leaves. These spread through petioles, and affect the stem and eventually the whole plant, making the stem dark brown in colour.

Fig.1.1 Late blight

Fig.1.2 Early Blight

Early blight is another vicious fungal disease that affects the potato crop. The symptoms for this appear on older leaves, which are soon going to wither. Dark brown/ Black spots appear on these leaves surrounded by concentric rings. These spots enlarge with time, and eventually lead to the deterioration of the leaf . The stems also get affected by these spots, which become firm and very deep with time. The pathogen causing this, is also harmful to animals and humans on contact, and may cause minor dermatological conditions.

Both these diseases 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

very difficult to rehabilitate the crop, so these diseases need to be identified at very early stages to take adequate measures against it.

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 - VGG16, VGG19, ResNet50 and MobileNet to classify the potato leaf blight diseases. Then the optimal-performing model is identified and is linked with the WebApp GUI for use.

1.1. Summary of Base Paper

Title: Automated recognition of optical image based potato leaf blight diseases using deep learning

Author: Kulendu Kashyap Chakraborty, Rashmi Mukherjee, Chandan Chakraborty, Kangkana Bora

Journal Name: Physiological and molecular plant pathology

Volume: 117

Publisher: Elsevier

Year: 2022

URL: https://doi.org/10.1016/j.pmpp.2021.101781

CHAPTER 2

LITERATURE SURVEY

Paper Title	**Pub-lished year**	**Authors' names**	**Elaboration**
Roots and tuber crops as functional foods: a review on phytochemical constituents and their potential health benefits	2016	A.Chandrasekaran, T.J. Kumar	Talks about the health benefits of starchy crops like common potatoes, sweet potatoes, and explains its nutritional and even some medicinal benefits.
Management of early blight of potato by using different bioagents as tuber dressing and its effect on germination and growth	2019	A.Shukla, V. Ratan	Discusses various biological solutions like seed dressing and bio-formulation to cure or prevent early blight condition in the potato crop.
Threats to global food security from emerging fungal and oomycete crop pathogens	2020	H.N. Fones, D.P. Bebber, T.M. Chaloner	Reviews the parameters influencing the spread of pathogens and disease control techniques, quoting with a few historic examples like the Irish potato famine.
Investigating potato late blight physiological differences across potato cultivars with spectroscopy and machine learning	2020	Kaitlin M. Gold, Philip A. Townsend, Ittai Herrman, Amanda J. Gevens	Explains about various types of late blight diseases found in potato plants across the world and a few instantaneous disease-detection methods used in the industry.

CHAPTER 3

PROPOSED WORK

The objective of this project is to present a system to classify the leaves into the three classes i.e. two types of diseases and healthy categories based on the leaf condition by utilising 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 VGG16, VGG19, RESNET50, and MOBILENET. An interactive GUI was built for predicting if the newly uploaded image of a potato leaf is early blight, late blight, or healthy.

3.1 Methodology:

The proposed methodology for classifying the 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.

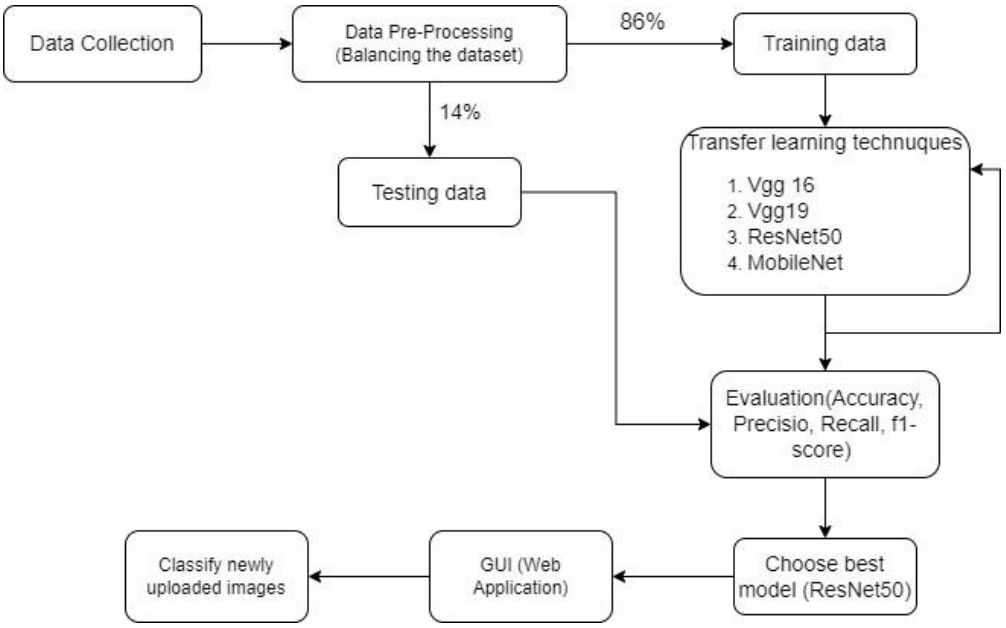


Figure 3.1 Work flow diagram

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

Total number of images = 2152

Number of images in Early_blight = 1000

Number of images in Late_blight = 1000

Number of images in healthy = 152

3.1.2 Data Preprocessing:

The number of images in the healthy category is 152 and the other categories have 1000 each. The imbalance in the data is of the ratio 1 : 6.5. Imbalanced classifications pose a challenge for predictive modelling 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.

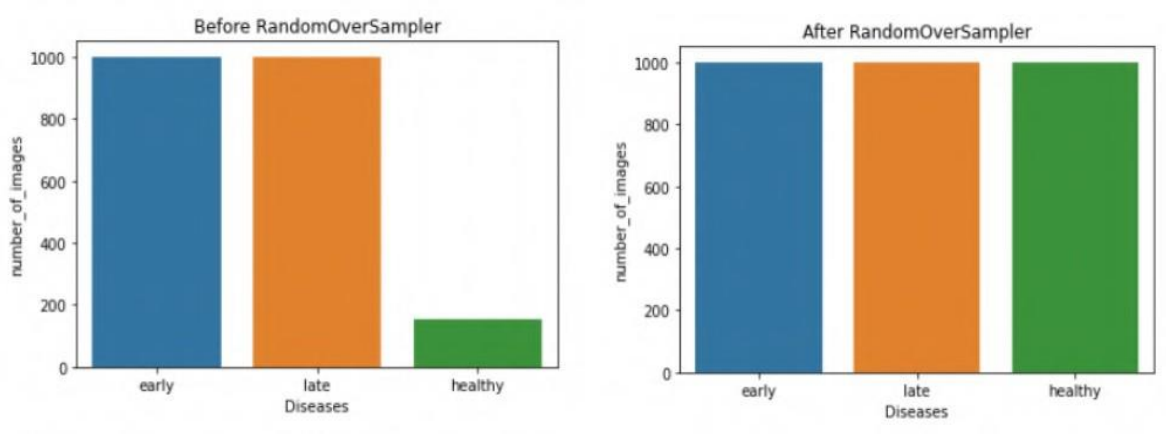


Figure 3.2 Dataset before and after RandomOverSampler

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 VGG16, VGG19, MobileNet, and ResNet50 were trained to classify. 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 ResNet50 gave the best results with around **99.77%** validation accuracy.

The models were run for 20 epochs; batch size = 64,optimizer used - RMSprop, test size = 14%, validation split = 33% i.e. number of images in training dataset 2585, testing dataset = 421, validation = 853 images.

3.1.4 Choosing the best performing model

Among all the four given models, ResNet50 outperforms the other models. The model after training for 20 epochs gave ~99.77% validation accuracy and 99%, 98%, 99% precision on the 3 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 initialised 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 the three classes.

3.1.5 Interactive WebApp for prediction(GUI):

An interactive web application was developed which uses the ResNet50 model trained on the given dataset for classifying newly uploaded images.

The 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

FINAL_PROJECT.ipynb

```
import matplotlib.pyplot as plt

from imblearn.over_sampling import RandomOverSampler

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

import os, cv2

from tensorflow.keras.optimizers import RMSprop

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D

from sklearn.metrics import confusion_matrix , classification_report

import os

import tensorflow as tf

from tensorflow.keras.optimizers import RMSprop, SGD, Adam

from tensorflow.keras.preprocessing.image import ImageDataGenerator

import numpy as np

import tensorflow as tf

import keras_tuner as kt

from keras_tuner.tuners import RandomSearch

from tensorflow.keras.models import Sequential,Model
```

```
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 =
["VGG16","VGG19","RESNET50","MOBILENET"])

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 4 different models were compared namely VGG16 , VGG19, RESNET50,
MOBILENET</h2>'

st.markdown(m2 , unsafe_allow_html = True)
```

```
m2 = '<h2 style="font-family:sans-serif; color: BLACK; font-size: 20px; align ="right">The proposed model is RESNET50</h2>'

st.markdown(m2 , unsafe_allow_html = True)

if x == "VGG16":

    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("vgg16.png")

        img2 = Image.open("vgg16results.png")

        st.subheader("Model architecture")

        st.image(img1)

        st.subheader("Model accuracy plots")

        st.image(img2)

        st.subheader("Classification Report")

        st.write(df)

if x == "VGG19":

    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("vgg19.png")

        img2 = Image.open("vgg19results.png")

        st.subheader("Model architecture")

        st.image(img1)

        st.subheader("Model accuracy plots")
```

```
        st.image(img2)
        st.subheader("Classification Report")
        st.write(df)
if x == "RESNET50":
  #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("resnet_model.png")
        img2 = Image.open("resnetresults.png")
        st.subheader("Model architecture")
        st.image(img1)
        st.subheader("Model accuracy plots")
        st.image(img2)
        st.subheader("Classification Report")
        st.write(df)
if x == "MOBILENET":
    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("mobilenet.png")
        img2 = Image.open("mobilenetresults.png")
        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 == "VGG16":
    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]
            index = ans.argmax(axis = 0)
            st.write("The uploaded leaf images is predicted to be of the class : " + classes[index])
            st.write("CONFIDENCE =",max(ans)*100, "%")
if x == "VGG19":
    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]
            index = ans.argmax(axis = 0)
            st.write("The uploaded leaf images is predicted to be of the class" + classes[index])
            st.write("CONFIDENCE =",max(ans)*100, "%")
if x == "RESNET50":
    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]
            index = ans.argmax(axis = 0)
            st.write("The uploaded leaf images is predicted to be of the class : " + classes[index])
            st.write("CONFIDENCE =",max(ans)*100, "%")
if x == "MOBILENET":
    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]
            index = ans.argmax(axis = 0)
            st.write("The uploaded leaf images is predicted to be of the class : " + classes[index])
            st.write("CONFIDENCE =",max(ans)*100, "%")
st.sidebar.markdown("Developed by: ")
st.sidebar.markdown("Adhitya Narayan 123003004 ")
st.sidebar.markdown("Rakshit Acharya 123003203 ")
```

CHAPTER 5

RESULTS AND SNAPSHOTS

5.1 VGG16:

Validation accuracy : ~98.65%
Validation loss : ~0.1421

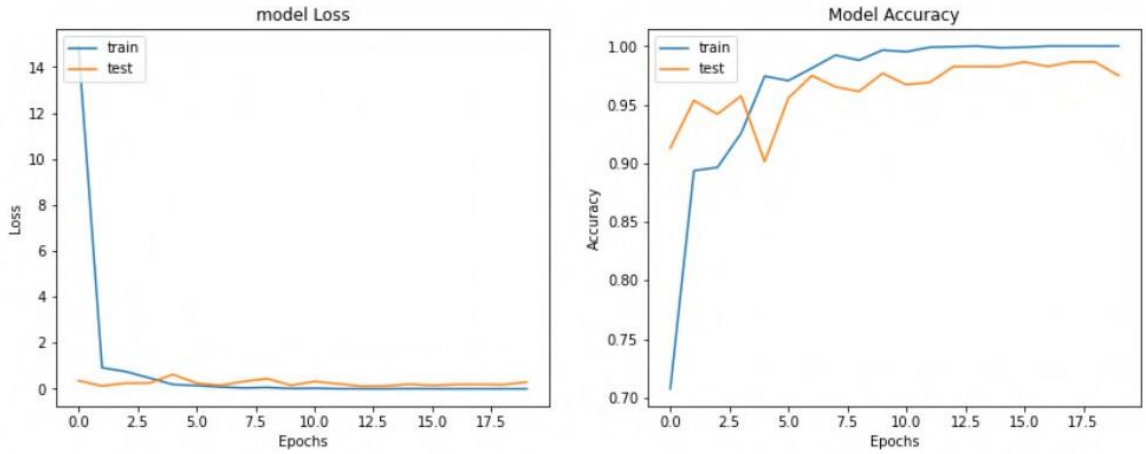


Figure 5.1.1 Accuracy and loss plots of VGG16

	precision	recall	f1-score	support
healthy	0.97	1.00	0.98	143
late_blight	0.98	0.93	0.96	153
early_blight	0.96	0.98	0.97	125
micro avg	0.97	0.97	0.97	421
macro avg	0.97	0.97	0.97	421
weighted avg	0.97	0.97	0.97	421
samples avg	0.97	0.97	0.97	421

Figure 5.1.2 Classification Report of VGG16

5.2 VGG19:

Validation accuracy : ~98.13%
Validation loss : ~0.1959

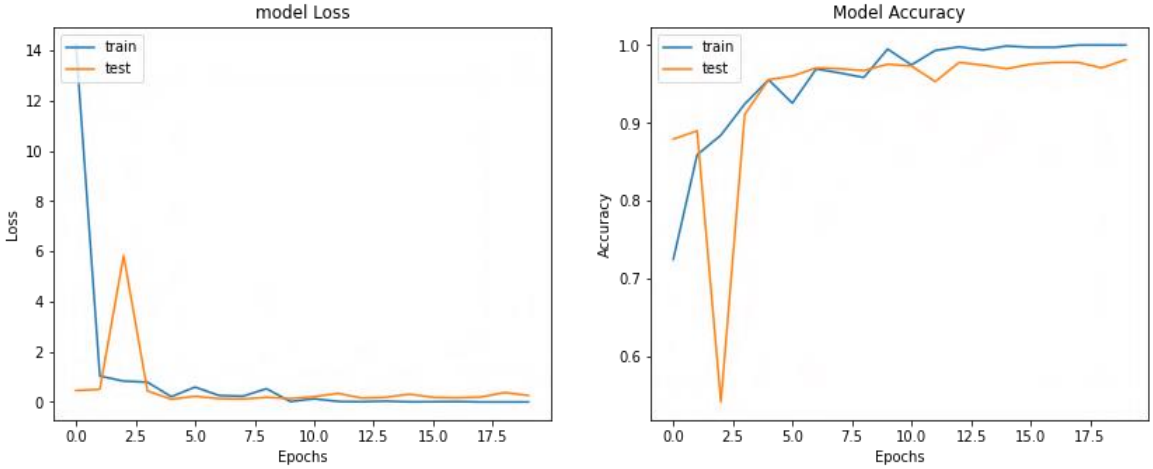


Figure 5.2.1 Accuracy and loss plots of VGG19

	precision	recall	f1-score	support
healthy	0.98	1.00	0.99	143
late_blight	0.99	0.97	0.98	153
early_blight	0.98	0.98	0.98	125
micro avg	0.98	0.98	0.98	421
macro avg	0.98	0.98	0.98	421
weighted avg	0.98	0.98	0.98	421
samples avg	0.98	0.98	0.98	421

Figure 5.2.2 Classification report of VGG19

5.3 MOBILENET :

Validation accuracy : ~98.59%
Validation loss : ~0.0489

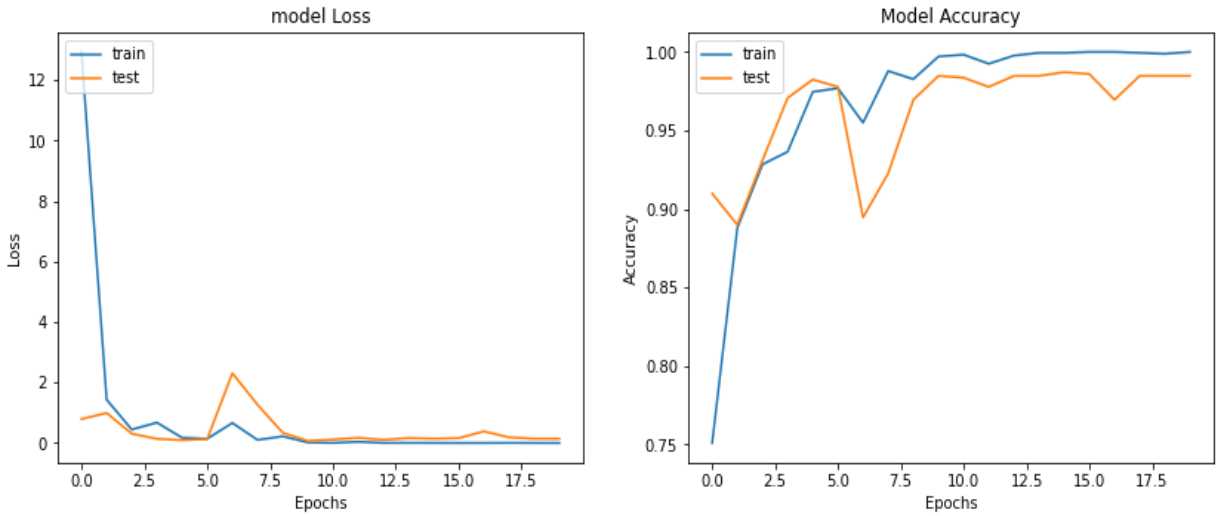


Figure 5.3.1 Accuracy and loss plots of Mobilenet

	precision	recall	f1-score	support
healthy	0.97	1.00	0.98	143
late_blight	0.98	0.93	0.95	153
early_blight	0.95	0.98	0.96	125
micro avg	0.97	0.97	0.97	421
macro avg	0.97	0.97	0.97	421
weighted avg	0.97	0.97	0.97	421
samples avg	0.97	0.97	0.97	421

Figure 5.3.2 Classification report of Mobilenet

5.4 RESNET50 :

Validation accuracy : ~99.77%
Validation loss : ~0.0055

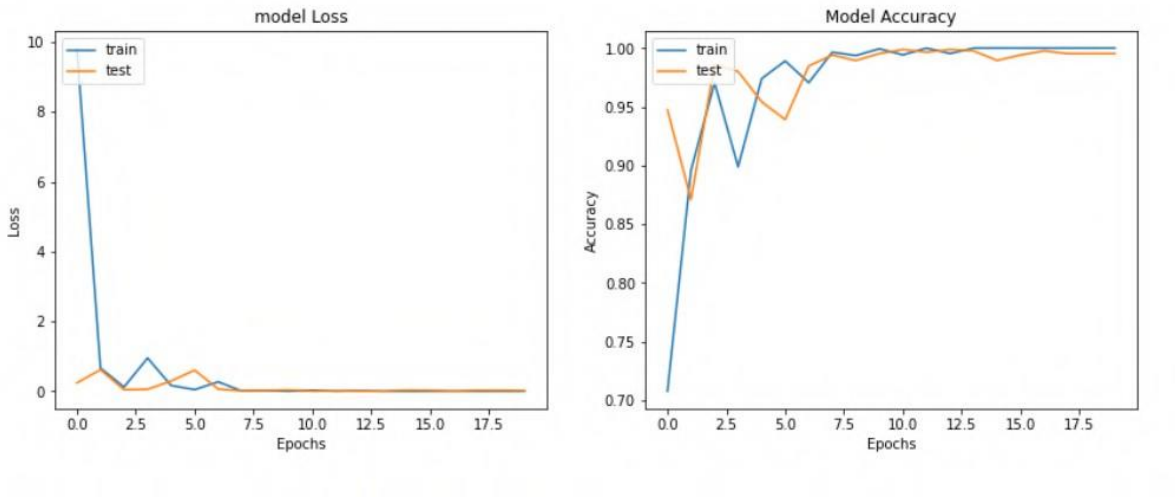


Figure 5.4.1 Accuracy and loss plots of RESNET50

	precision	recall	f1-score	support
healthy	0.98	1.00	0.99	143
late_blight	0.99	0.97	0.98	153
early_blight	0.98	0.99	0.99	125
micro avg	0.99	0.99	0.99	421
macro avg	0.99	0.99	0.99	421
weighted avg	0.99	0.99	0.99	421
samples avg	0.99	0.99	0.99	421

Figure 5.4.2 Classification report of RESNET50

5.5 Interactive Web App (using streamlit):

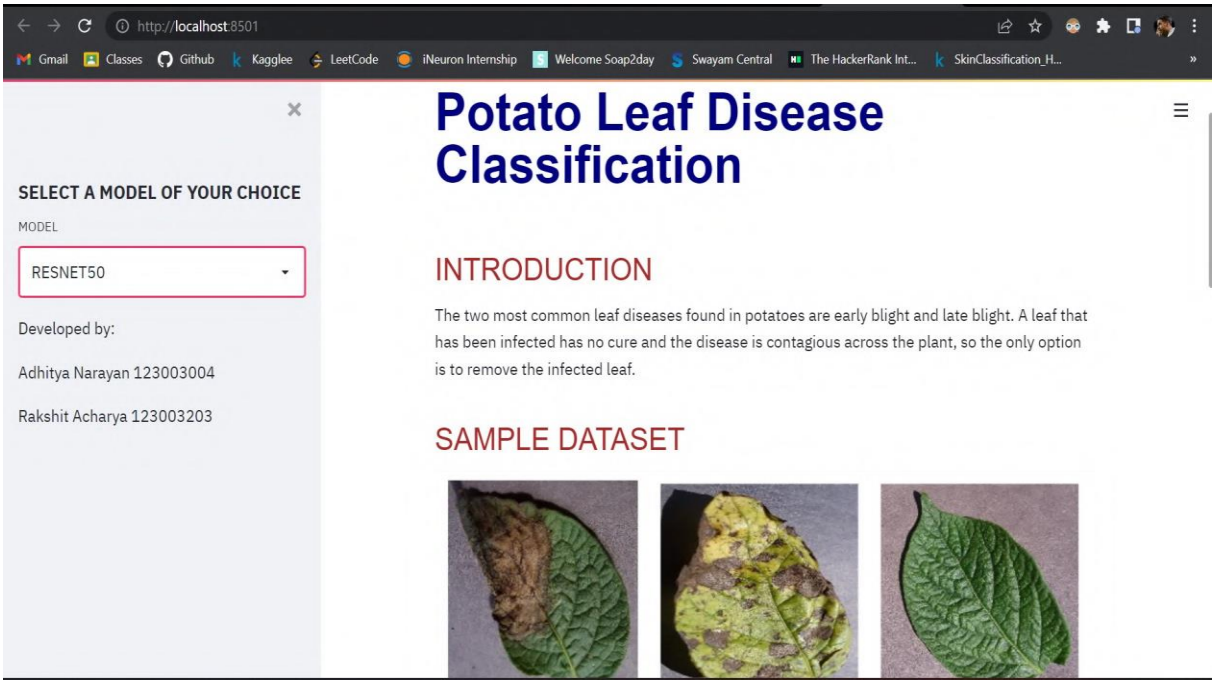


Figure 5.5.1 Graphical user interface

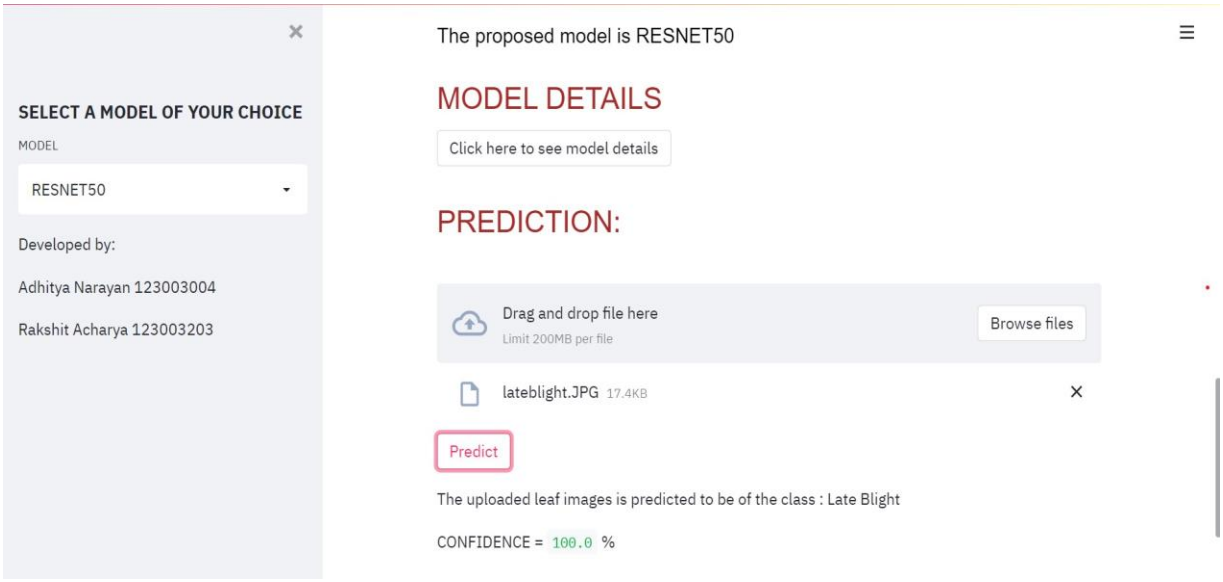


Figure 5.5.2 Graphical user interface for prediction

CHAPTER 6

PROJECT TEAM AND INDIVIDUAL CONTRIBUTION

Register number	Name	Individual Contribution
123003004	Adhitya Narayan P A	● Data Collection and preprocessing (Week 1 - Week 2) ● Developed an interactive WebApp to implement the blight disease classifier model(Week 7- Week 8)
123003203	Rakshit Acharya	● Data Collection and preprocessing (Week 1 - Week 2) ● Implement the transfer learning techniques(Week 3 - 5) ● Compare the models based on evaluation metrics and suggest the model (Week 6)

Table 6.1 Individual Contributions

CHAPTER 7

CONCLUSION AND FUTURE PLANS

In this study, a three-class classification problem is exhibited to detect infected potato leaves as early blight or late blight or healthy, in the earlier stages of the infection so that it can be treated as soon as possible. Considerable data augmentation using ImageDataGenerator and RandomOverSample 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 VGG16, VGG19, Mobilenet and ResNet50, and the results obtained are evaluated and assessed.

The results show that ResNet50 performed the best among these four, achieving an validation accuracy of ~99.77% and further has been optimised to provide better results. Further, with these models, a WebApp GUI has been built to facilitate the use of these models by farmers and other relevant communities. We beleive 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 a decent accuracy.

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

CHAPTER 8

REFERENCES

D. Tiwari, M. Ashish, N. Gangwar, A. Sharma, S. Patel and S. Bhardwaj, "Potato Leaf Diseases Detection Using Deep Learning," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), 2020, pp. 461-466, doi: 10.1109/ICICCS48265.2020.9121067.

Hughes, D.P., Salathe, M.: An open access repository of images on plant health to enable the development of mobile disease diagnostics through machine learning and crowdsourcing. CoRR, abs/1511.08060

R. A. Sholihati, I. A. Sulistijono, A. Risnumawan and E. Kusumawati, "Potato Leaf Disease Classification Using Deep Learning Approach," 2020 International Electronics Symposium (IES), 2020, pp. 392-397, doi: 10.1109/IES50839.2020.9231784.

P. Patil, N. Yaligar and S. M. Meena, "Comparison of Performance of Classifiers - SVM, RF and ANN in Potato Blight Disease Detection Using Leaf Images," 2017 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), 2017, pp. 1-5, doi: 10.1109/ICCIC.2017.8524301.

Z. Saeed, M. Urooj Khan, A. Raza, N. Sajjad, S. Naz and A. Salal, "Identification of Leaf Diseases in Potato Crop Using Deep Convolutional Neural Networks (DCNNs)," 2021 16th International Conference on Emerging Technologies (ICET), 2021, pp. 1-6, doi: 10.1109/ICET54505.2021.9689807.