**LEAF DISEASE DETECTION**

**A CAPSTONE PROJECT REPORT**

*Submitted in partial fulfillment of the*

*requirement for the award of the*

*Degree of*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE ENGINEERING**

*by*

**MOPARTHI BAVITA(19BCN7016)**

**NITTOR VISHNU BHARADWAJ(19BCE7478)**

**POPURI HARSHA VARDHAN(19BCI7039)**

*Under the Guidance of*

**DR. M.PRIYA DARSHINI**



SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

VIT-AP UNIVERSITY

AMARAVATI- 522237

*DECEMBER 2022*

**CERTIFICATE**

This is to certify that the Capstone Project work titled “**LEAF DISEASE DETECTION**” that is being submitted by **MOPARTHI BAVITA(19BCN7016)**

**NITTOR VISHNU BHARADWAJ(19BCE7478) and POPURI HARSHA VARDHAN(19BCI7039)** is in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. M.PRIYADARSHINI

Guide

**The thesis is satisfactory / unsatisfactory**

**Internal Examiner External Examiner**

**Approved by**

**PROGRAM CHAIR DEAN**

B. Tech. ECE School Of Computer Science Engineering

**ACKNOWLEDGEMENTS**

I would like to express my gratitude and thanks to my guide Prof. Dr. M. Priyadharshini Mam, Department of Computer Science and Engineering, VIT-AP University, for all the time allotted to help us, her guidance and keeping us on the correct path that allowed us to carry out the project work deliberately that laid a strong foundation for our upcoming professional life. Even Though this project has been completed virtually, our guide Dr. M. Priyadharshini Mam was always ready to help with her valued suggestions. I truly appreciate and value her admired supervision and support from the start to the end of this project. We are grateful to her for having helped us shape the trouble and providing insights towards the way out.

**ABSTRACT**

In this project we have created an pretrained leaf disease detection system using deep learning techniques. Plant diseases pose a significant risk to agricultural productivity, and early detection remains a difficult task for farmers due to their similar look in colour, shape, and texture.  Transfer learning is used to fine-tune the previously learned models.

To overcome overfitting, data augmentation techniques like as picture enhancement, rotation, scaling, and translation are used. This research gives a detailed taxonomy of the performance of several pre-trained neural networks in plant leaf disease detection.

Furthermore, the suggested work's performance is evaluated using a publicly accessible plant village dataset, which consists of 38 classes collected from 14 crops.  The suggested model's performance evaluation shows that it is effective at categorizing various types of plant illnesses.

**TABLE OF CONTENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Chapter** | **Title** | **Page Number** |
| **0.** |  | Certificate | 1 |
| **1.** |  | Acknowledgements | 2 |
| **2.** |  | Abstract | 3 |
| **3.** |  | Table Of Contents | 4 |
| **4.** |  | List of Figures and Table | 5 |
| **5.** | 1 | Introduction | 7 |
|  | 1.1 | Objectives | 8 |
|  | 1.2 | Background and Literature Survey | 8 |
|  | 1.3 | Organization Of the Report | 9 |
| **6.** | 2 | Leaf Disease Detection | 10 |
|  | 2.1 | Proposed System | 10 |
|  | 2.2 | Working Methodology | 10 |
|  | 2.2.1 | Dataset | 10 |
|  | 2.2.2 | Models | 11 |
|  | 2.2.3 | Plant Leaf Disease detection using CNN algorithm | 12 |
|  | 2.3 | System Details | 13 |
|  | 2.3.1 | Software | 13 |
|  | 2.3.2 | Hardware | 13 |
| **7.** | 3 | Cost Analysis | 13 |
|  | 3.1 | List of components and their cost | 13 |
| **8.** | 4 | Results and Discussion | 14-16 |
| **9.** | 5 | Conclusion & Future Scope | 17 |
| **10.** | 6 | Appendix | 17-28 |
| **11.** | 7 | References | 28-30 |
| **12.** |  | Bio data | 31 |

**List of Tables**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Title** | **Page No** |
| 1 | List Of Components and The Cost Incurred | 13 |

## List of Figures

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Title** | **Page No** |
| 1 | Architecture Diagram | 9 |
| 2 | Inception V3 architecture | 10 |
| 3 | InceptionResNetV2 Basic architecture | 11 |
| 4 | MobileNet Layers | 12 |
| 5 | InceptionV3-output | 14 |
| 6 | InceptionResNetV2-output | 15 |
| 7 | MobileNet-output | 16 |

**CHAPTER 1**

**INTRODUCTION**

Automated plant disease identification has become one of the most important precision agricultural activities in recent years. Fungus, bacterial, and insect diseases impair agricultural production and lower productivity. The complexity of the pattern variations and the high levels of inter-class similarity make it difficult to classify plant leaf diseases. Climate changes can significantly accelerate the growth of plant diseases. One of the main obstacles to optimum output in the agricultural sector is the early detection of illnesses on plant leaves.

Diseases have already lowered plant output by more than 50%, thus early detection allows for earlier management, which lowers crop loss.In this work, we have used pre-trained deep learning models for the accurate identification of several plant leaf diseases in realistic circumstances, as opposed to typical shallow machine learning approaches. For reliable classification of plant diseases, researchers have proposed a wide range of computer vision-based machine learning and deep learning techniques. In this situation, CNN models have shown to be effective at conducting picture segmentation and classification when the number of data samples is large.

The classification of weeds, plant diseases, pests, damaged and good fruits, and weed infestations are all further applications of deep learning models. With the use of transfer learning, these deep learning models have high hopes for extracting discriminating features. The suggested model can accurately classify plant leaf diseases and derive distinguishing characteristics from photos of leaves.

The performance of the suggested strategy is assessed using the 54,305 photos from 38 classes that make up the plant village dataset. 80 percent of these photographs, or 43,456, are utilised to develop models, while the remaining 10,849 images are used to assess how well the suggested strategy is working.

* 1. **Objectives**

The objectives of our project are:

* + To Build a model that can help farmers for the early identification leads to quicker intervention that reduces crop loss .
  + To Build an Efficient Model, Which Can help farmers to increase the crop production.
  + Should be helpful for farmers in preventing the spread of disease to the entire crop.
  + Building the model using various Deep Learning Algorithms and give a decent accuracy.
  + Utilize the resources intelligently.
  1. **Background and Literature Survey**

Hyperspectral image benefits are described on various scales for plant protection and plant disease detection using deep learning models that are optimized for plant disease detection (Thomas et al. 2018). With the help of IPM and Bing test datasets, Lee et al. (2020) suggested a method for examining the most pertinent plant disease diagnosis. Fast-RCNN and PestNet comparisons are done using the VGG16 network, which is employed for pest detection. Similarly, GoogLeNet BN and InceptionV3 networks of 34 and 48 layers, respectively, are employed to evaluate the effectiveness in the detection of plant illnesses. Liu et al. (2019) proposed a PestNet that was primarily concerned with feature improvement and extraction and examined the identification of pests in plant leaves using a region-based technique.

Different deep learning architectures, including VGG16, Inception V4 101, ResNet 152, and DenseNet 121, are used to evaluate the efficacy of classifying plant diseases (Too et al. 2019). Compared to other deep neural network techniques, DenseNet 121 exhibits the highest accuracy. A reliable technique to identify and categorise the pest diseases that affected tomato plants was put forth by Fuentes et al. (2017). The classification of leaf diseases using deep convolutional neural networks as pre-trained models was proposed by Zhang et al. (2018). To identify tomato leaf diseases, CNN pre-trained models like AlexNet, GoogLeNet, and ResNet are employed. With stochastic gradient descent, the ResNet approach identifies tomato plant leaf diseases with the highest degree of accuracy (SGD). Sladojevic et al. (2016) proposed a novel technique to distinguish healthy plant leaves from their surroundings and identify 13 diseases that affected plants. A machine learning method was proposed by Ferentinos (2018), Sha- rif et al. (2018), and Arnal Barbedo (2019) to investigate the lesions and spots of the plants without taking into account the full leaf. Using CNN techniques, they developed a hybridization strategy to identify illnesses of citrus plants. Citrus fruit segmentation and multiple support vector machines are used for classification and lesion spot identification (M-SVM). The citrus plant is utilised to extract the colour, geometric, and textural elements, which are then combined to create a codebook. Barbedo (2018) conducted research on how deep learning influences disease identification in several plant species. A model to assess crop pest categorization utilising pre-trained models such as AlexNet, ResNet, VGGNet, and GoogLeNet was proposed by Thenmozhi and Srinivasulu Reddy in 2019. In the agricultural sector, pre-trained models have greater accuracy for crop protection. With the aid of deep learning models, Knoll et al. (2018) showed how to improve organic farming effectively.

**1.3 Organization of the Report**

The remaining chapters of the project report are described as follows:

* Chapter 2 contains the proposed system, methodology, hardware and software details.
* Chapter 3 gives the cost involved in the implementation of the project.
* Chapter 4 discusses the results obtained after the project was implemented.
* Chapter 5 concludes the report.
* Chapter 6 consists of codes.
* Chapter 7 gives references.

**CHAPTER 2**

**LEAF DISEASE DETECTION**

This Chapter describes the proposed system, working methodology, software details.

**2.1 Proposed System**

The following block diagram (figure 2) shows the system architecture of this project.

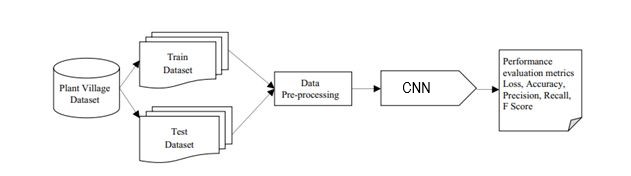


Figure Architecture Diagram

**2.2 Working Methodology**

**2.2.1:Dataset**

Plant village dataset is the largest publicly available dataset on plant diseases . The Plant Village dataset consists of 54303 healthy and unhealthy leaf images divided into 38 categories by species and disease This dataset comprises RGB images of Journal of Plant Diseases and Protection 13 both infected and healthy leaves collected uniformly from 14 diferent types of crops including Tomato, Blueberry, Cherry, Corn, Apple, Strawberry, Orange, Peach, etc . We have used plant village dataset for this model. The dataset is divided into training and testing sets. The training set contains of 80% of total images where the testing set contains 20% images of the total.

**2.2.2:Models**

**Inception V3**

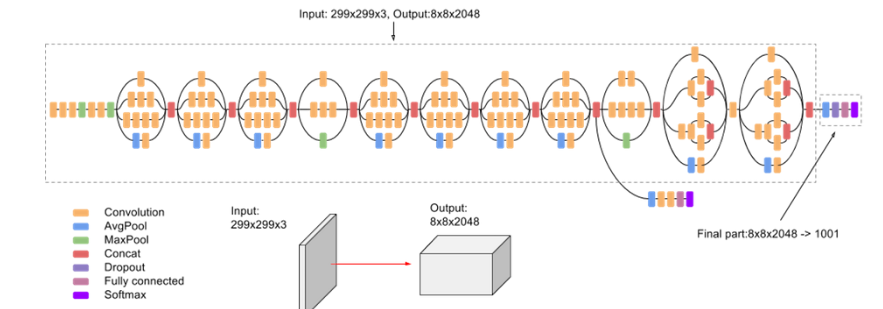
The Inception V3 model is simply the Inception V1 model upgraded and improved. Multiple methods were employed by the Inception V3 model to optimise the network for improved model adaption.

• It has higher efficiency

• It has a deeper network compared to the Inception V1 and V2 models, but its speed isn't compromised.

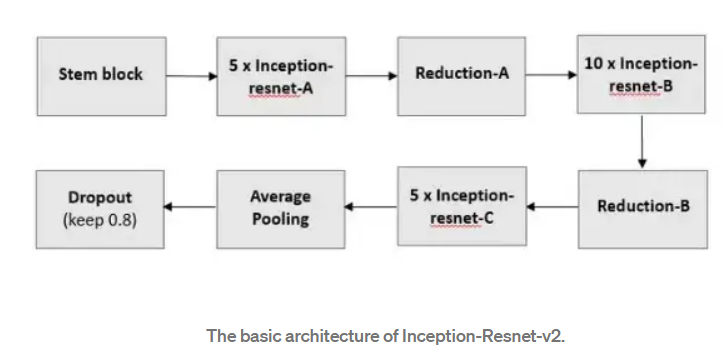
• It is computationally less expensive.

• It uses auxiliary Classifiers as regularizes.



**InceptionResNetV2:**

A convolutional neural network named Inception-ResNet-v2 was trained using more than a million photos from the ImageNet collection. The 164-layer network can categorize photos into 1000 different object categories, including the keyboard, mouse, pencil, and numerous animals. The network has therefore acquired rich feature representations for a variety of images. The network receives a 299 by 299 pixel picture as input, and it outputs a list of estimated class probabilities.



**MobileNet:**

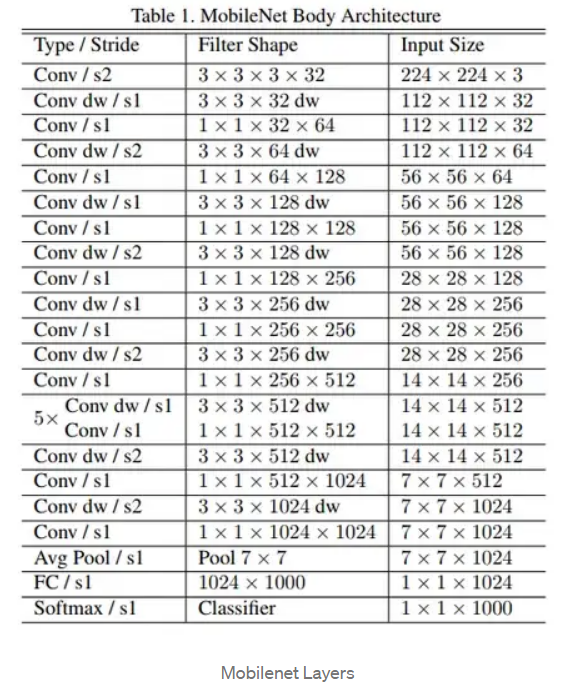
MobileNet uses **depthwise separable convolutions.**It significantly **reduces the number of parameters** when compared to the network with regular convolutions with the same depth in the nets. This results in lightweight deep neural networks.

A depthwise separable convolution is made from two operations.

1. **Depthwise convolution.**
2. **Pointwise convolution**.

This offers us a great place to start when training our classifiers, which are ridiculously small and unbelievably quick. MobileNet is a class of CNN that was open-sourced by Google.

MobileNets are a family of mobile-first computer vision models for tensor flow, designed to effectively maximize accuracy while being mindful of the restricted resources for an on-device or embedded application.

****

**2.2.3: Plant leaf disease detection using CNN algorithms**

In this study, we present a reliable and practical approach for identifying plant leaf diseases. We have employed deep learning techniques to reduce false positive and false negative with constrained computational resources. Fig. depicts the suggested methodology's architectural layout.

First, preprocessing is used to enhance the image's quality while giving it a quantitative focus on contrast and brightness. Image improvement, color space transformation, scaling, and noise removal are some of the preprocessing techniques. In this work, the performance of uni-modal pre-trained neural networks like InceptionResnetV2, InceptionV3, and MobileNet is being evaluated for the first time.

* 1. **System Details**

**2.3.1 Software Details**

* Python 3.7
* Google Colab

**2.3.2 Hardware Details**

* + 4GB RAM
  + WINDOWS 10

**CHAPTER 3**

**COST ANALYSIS**

**3.1 List of components and their cost**

The costs of the various components used in this project are given below in Table 3.1.

**Table 3.1 List of components and their costs**

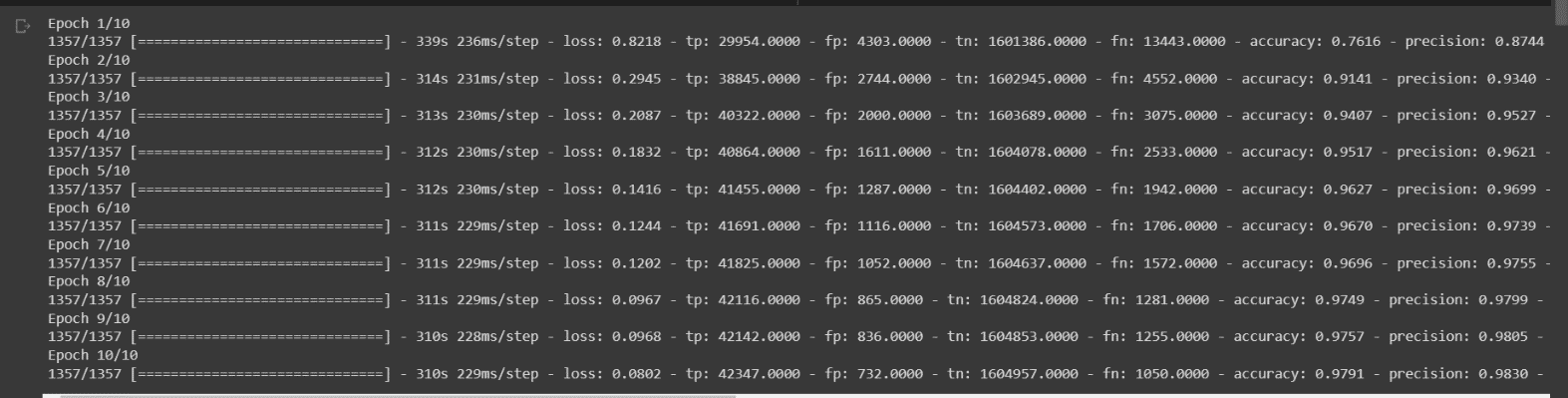
|  |  |
| --- | --- |
| **COMPONENT** | **COST** |
| Dataset - Kaggle | 740 Images |
| Google Colab | Free to Use for Non-Commercial Use |
| Miscellaneous | ₹ 0 |
| TOTAL | ₹ 0 |

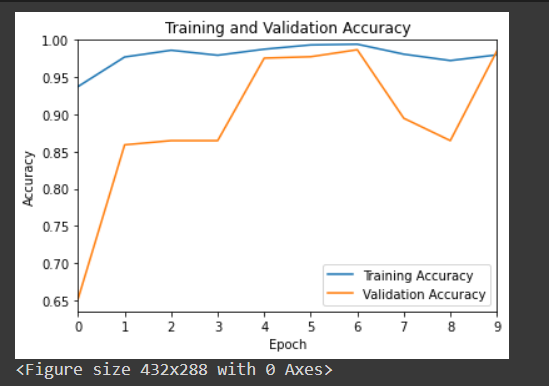
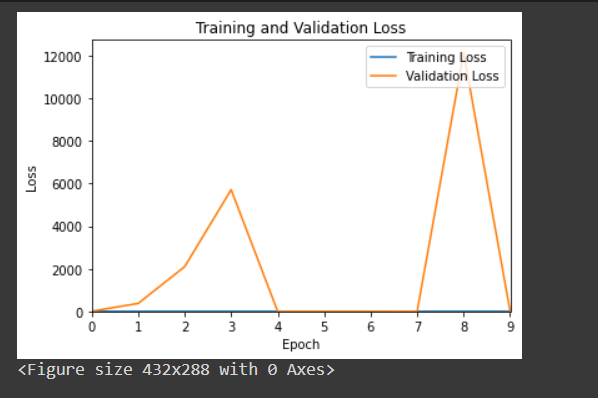
**CHAPTER 4**

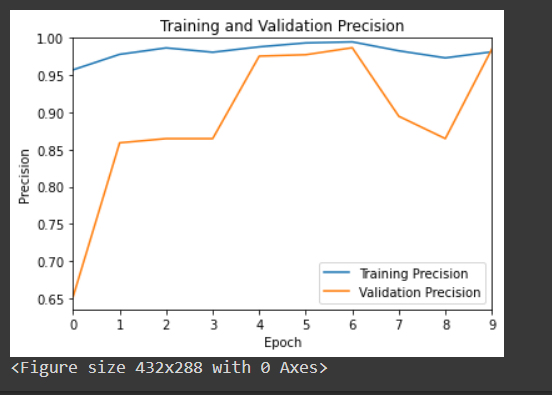
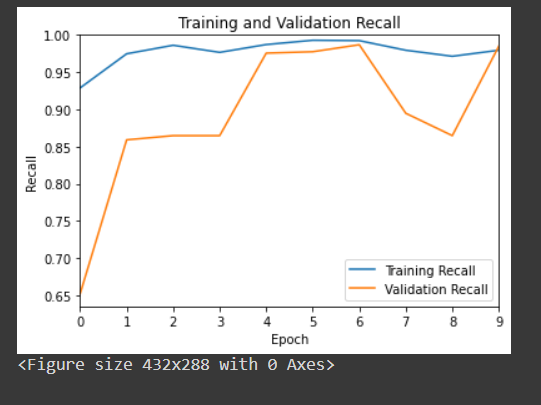
**RESULTS AND DISCUSSIONS**

**InceptionV3**

Here are the results of the models which are trained with the dataset. Inceptionv3 is giving the accuracy of up to 97.91% and with a good precision value of 98.3. The loss percentage of the model is also minimal with 0.0802%.

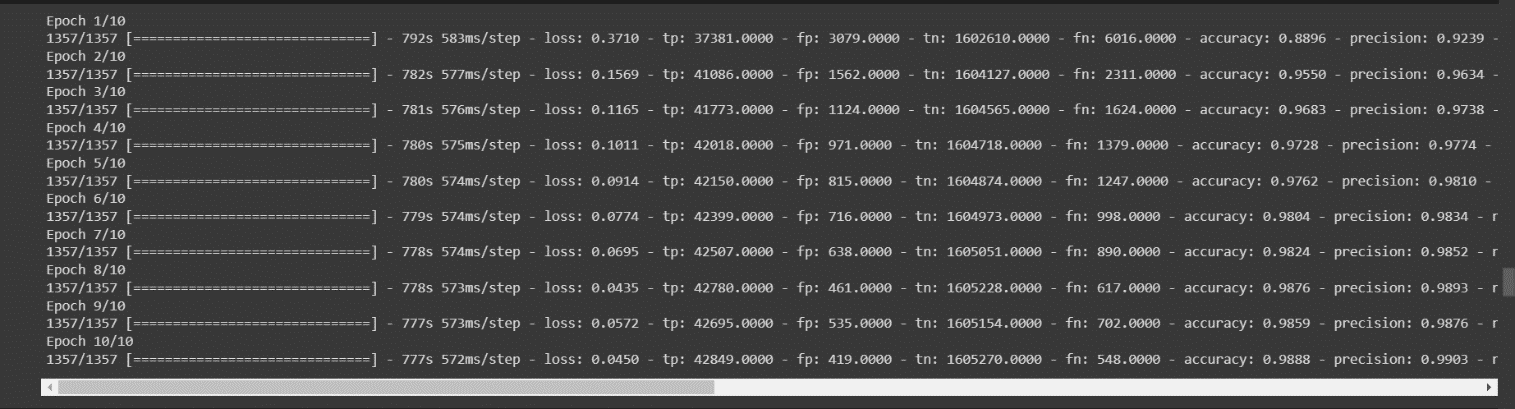
****

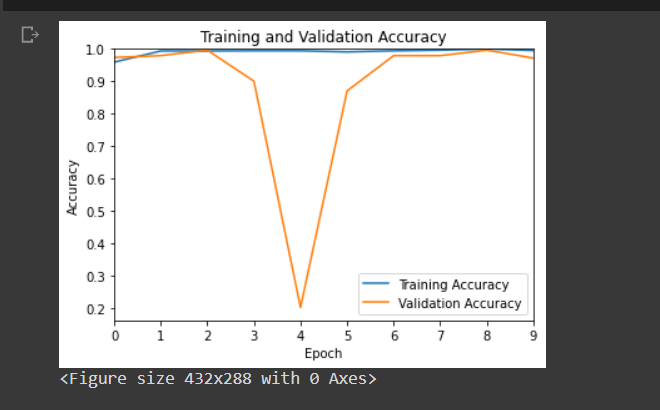
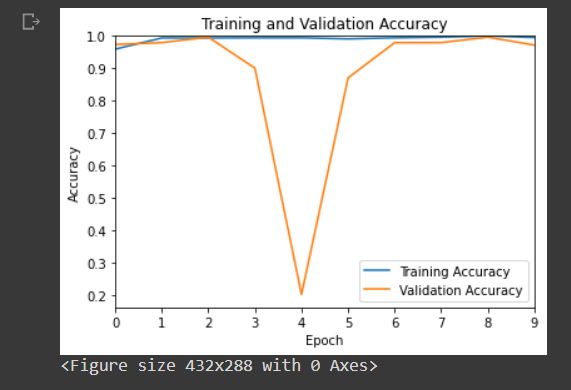
** **

** **

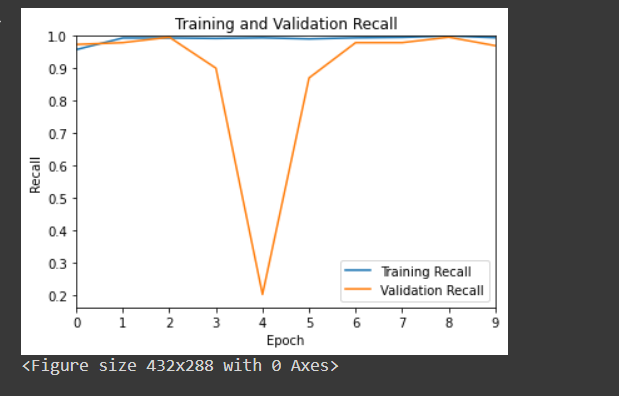
**InceptionResNetV2**

Here are the results of the models which are trained with the dataset. Inceptionv3 is giving the accuracy of upto 98.88 % and with a good precision value of 99.03. The loss percentage of the model is also minimal with 0.0450%.

****

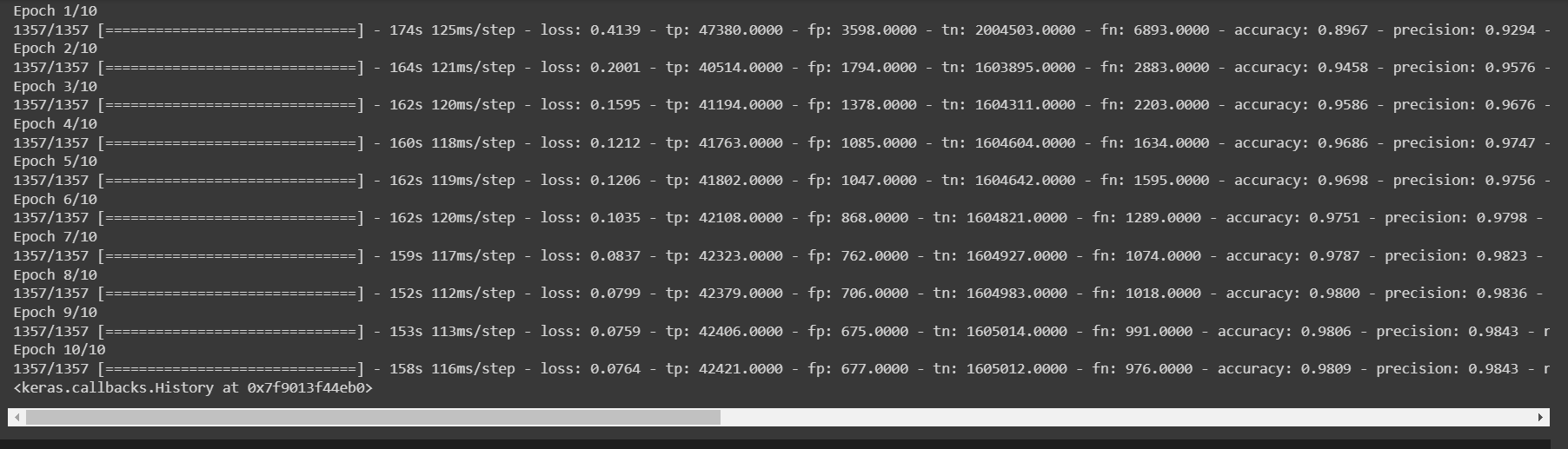
** **

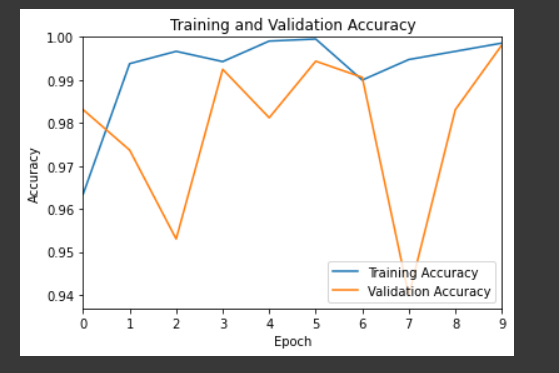
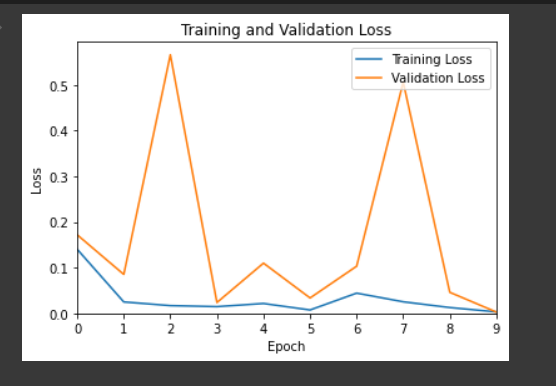
**Chart, line chart

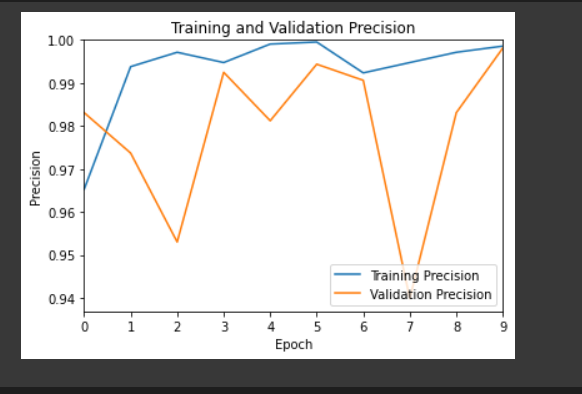
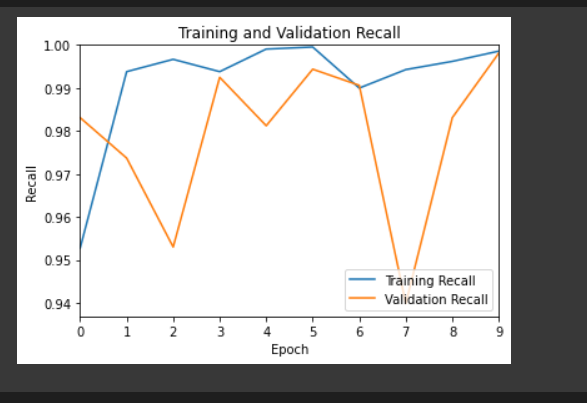
Description automatically generated **

**Mobilenet**

Here are the results of the models which are trained with the dataset. Inceptionv3 is giving the accuracy of upto 98.09 % and with a good precision value of 98.43. The loss percentage of the model is also minimal with 0.0764%.

****

** **

** **

**CHAPTER 5**

**CONCLUSION AND FUTURE WORK**

Early diagnosis of plant diseases is crucial since they have a negative impact on the development of the corresponding crops. The productivity and quality of crops are significantly impacted by several diseases, fungi, and insects. In this study, pre-trained models like Inception V3, InceptionResnetV2, and MobileNet are used to classify plant diseases based on visual cues.

Here, we have developed an effective deep learning system using a variety of pretrained models for classifying plant leaf diseases.

Additionally, the performance of the proposed approach is assessed using the 38 classes from 14 crops that make up the publicly available plant village dataset. The experimental results show that, when compared to pre-trained models, the proposed ensemble technique offers a high level of classification accuracy while using a minimal number of computations.when network are adjusted using data augmentation methods and transfer learning.

**CHAPTER 6**

**APPENDIX**

**CODE**

# -\*- coding: utf-8 -\*-

"""[CAP\_2022]Plants\_Disease.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1BZg2XgcqC0n58Tx\_QlLURnwo1sAj2VPF

"""

from google.colab import drive

drive.mount('/content/drive')

"""#### Files Copying from Zip to folder"""

!unzip /content/drive/MyDrive/plant.zip -d /content/drive/MyDrive/Extracted

"""#### Split Folders

"""

pip install split-folders

import splitfolders

splitfolders.ratio('/content/drive/MyDrive/Extracted/plantvillagedataset/color', output="output", seed=1337, ratio=(.8, .2), group\_prefix=None)

"""#### MAIN"""

import numpy as np

from re import sub

import tensorflow as tf

from tensorflow.keras import metrics

import matplotlib.colors as mcolors

from sklearn.metrics import confusion\_matrix

import seaborn as sns

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense,GlobalAveragePooling2D

from tensorflow.keras.layers import BatchNormalization

from tensorflow.keras.layers import Conv2D

from tensorflow.keras.layers import MaxPooling2D

from tensorflow.keras.layers import Activation, Flatten, Dropout, Dense

from tensorflow.keras.models import Model

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.preprocessing import image

import matplotlib.pyplot as plt

from tensorflow.keras import models

from tensorflow.keras.preprocessing.image import ImageDataGenerator

METRICS = [

metrics.TruePositives(name='tp'),

metrics.FalsePositives(name='fp'),

metrics.TrueNegatives(name='tn'),

metrics.FalseNegatives(name='fn'),

metrics.CategoricalAccuracy(name='accuracy'),

metrics.Precision(name='precision'),

metrics.Recall(name='recall'),

metrics.AUC(name='auc')

]

BATCH\_SIZE = 32

IMG\_SIZE = (224, 224)

IMG\_SHAPE = IMG\_SIZE + (3,)

"""Using InceptionV3 CNN """

from tensorflow.keras.applications.inception\_v3 import preprocess\_input

train\_datagen=ImageDataGenerator(preprocessing\_function=preprocess\_input)

train\_generator=train\_datagen.flow\_from\_directory('./output/train',

target\_size=IMG\_SIZE,

color\_mode='rgb',

batch\_size=BATCH\_SIZE,

class\_mode='categorical',

shuffle=True)

val\_datagen=ImageDataGenerator(preprocessing\_function=preprocess\_input)

val\_generator=train\_datagen.flow\_from\_directory('./output/val',

target\_size=IMG\_SIZE,

color\_mode='rgb',

batch\_size=BATCH\_SIZE,

class\_mode='categorical',

shuffle=True)

classes = train\_generator.class\_indices

class\_list = list(classes.keys())

base\_model=tf.keras.applications.InceptionV3(input\_shape=IMG\_SHAPE, weights='imagenet',include\_top=False)

x=base\_model.output

x=GlobalAveragePooling2D()(x)

x=Dense(1024,activation='relu')(x)

x=Dense(1024,activation='relu')(x)

x=Dense(512,activation='relu')(x)

x=Dense(256,activation='relu')(x)

preds=Dense(38,activation='softmax')(x)

model=Model(inputs=base\_model.input,outputs=preds)

for layer in model.layers[:20]:

layer.trainable=False

for layer in model.layers[20:]:

layer.trainable=True

model.compile(optimizer='Adam',loss='categorical\_crossentropy',metrics=METRICS)

step\_size\_train=train\_generator.n//train\_generator.batch\_size

model\_history=model.fit(train\_generator,validation\_data =val\_generator,steps\_per\_epoch=step\_size\_train,epochs=10)

#model=model\_history

def download\_and\_predict(filename):

# download and save

#os.system("curl -s {} -o {}".format(url, filename))

img = Image.open(filename)

img = img.convert('RGB')

img = img.resize((224, 224))

img.save(filename)

# show image

plt.figure(figsize=(4, 4))

plt.imshow(img)

plt.axis('off')

# predict

img = image.imread(filename)

img = preprocess\_input(img)

probs = model.predict(img)

for idx in probs.argsort()[0][::-1][:5]:

print("{:.2f}%".format(probs[0][idx]\*100), "\t", label\_maps\_rev[idx].split("-")[-1])

def download\_and\_predict(url, filename):

# download and save

os.system("curl -s {} -o {}".format(url, filename))

img = Image.open(filename)

img = img.convert('RGB')

img = img.resize((299, 299))

img.save(filename)

# show image

plt.figure(figsize=(4, 4))

plt.imshow(img)

plt.axis('off')

# predict

img = image.imread(filename)

img = preprocess\_input(img)

probs = model.predict(np.expand\_dims(img, axis=0))

for idx in probs.argsort()[0][::-1][:5]:

print("{:.2f}%".format(probs[0][idx]\*100), "\t", label\_maps\_rev[idx].split("-")[-1])

# Commented out IPython magic to ensure Python compatibility.

import os

import pandas as pd

import xml.etree.ElementTree as ET

#import gdown

import time

import math

import cv2

import numpy as np

from PIL import Image

import matplotlib.pyplot as plt

import matplotlib.image as image

from sklearn.model\_selection import train\_test\_split

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras import layers

from keras.applications.xception import Xception, preprocess\_input

from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlyStopping

from keras.layers import Conv2D, MaxPooling2D, GlobalAveragePooling2D

from keras.layers import Dropout, Flatten, Dense

from keras.models import Sequential

from keras.utils import np\_utils

from tensorflow.keras.utils import Sequence

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.applications.xception import decode\_predictions

# %matplotlib inline

from sklearn.metrics import classification\_report

import sklearn.metrics

from PIL import Image

import matplotlib.image as mpimg

#from imgaug import augmenters as iaa

download\_and\_predict("/content/test1.png","test1.png")

test,test\_acc = model.evaluate\_generator(generator = val\_generator,steps=int(20))

acc = model\_history.history['accuracy']

val\_acc = model\_history.history['val\_accuracy']

plt.plot(acc, label='Training Accuracy')

plt.plot(val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.ylabel('Accuracy')

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation Accuracy')

plt.xlabel('Epoch')

plt.show()

plt.savefig("model\_Accuracy.png")

loss = model\_history.history['loss']

val\_loss = model\_history.history['val\_loss']

plt.plot(loss, label='Training Loss')

plt.plot(val\_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.ylabel('Loss')

plt.xlim([0,9])

plt.ylim([0,max(plt.ylim())])

plt.title('Training and Validation Loss')

plt.xlabel('Epoch')

plt.show()

plt.savefig("model\_Loss.png")

pre = model\_history.history['precision']

val\_pre = model\_history.history['val\_precision']

plt.plot(pre, label='Training Precision')

plt.plot(val\_pre, label='Validation Precision')

plt.legend(loc='lower right')

plt.ylabel("Precision")

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation Precision')

plt.xlabel('Epoch')

plt.show()

plt.savefig("model\_Precision.png")

rec = model\_history.history['recall']

val\_rec = model\_history.history['val\_recall']

plt.plot(rec, label='Training Recall')

plt.plot(val\_rec, label='Validation Recall')

plt.legend(loc='lower right')

plt.ylabel("Recall")

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation Recall')

plt.xlabel('Epoch')

plt.show()

plt.savefig("model\_Recall.png")

"""# \*\*InceptionResnetV2\*\*

"""

from tensorflow.keras.applications.inception\_resnet\_v2 import preprocess\_input

train\_datagen=ImageDataGenerator(preprocessing\_function=preprocess\_input)

train\_generator=train\_datagen.flow\_from\_directory('./output/train',

target\_size=IMG\_SIZE,

color\_mode='rgb',

batch\_size=BATCH\_SIZE,

class\_mode='categorical',

shuffle=True)

val\_datagen=ImageDataGenerator(preprocessing\_function=preprocess\_input)

val\_generator=train\_datagen.flow\_from\_directory('./output/val',

target\_size=IMG\_SIZE,

color\_mode='rgb',

batch\_size=BATCH\_SIZE,

class\_mode='categorical',

shuffle=True)

base\_model=tf.keras.applications.InceptionResNetV2(input\_shape=IMG\_SHAPE, weights='imagenet',include\_top=False)

x=base\_model.output

x=GlobalAveragePooling2D()(x)

x=Dense(1024,activation='relu')(x)

x=Dense(1024,activation='relu')(x)

x=Dense(512,activation='relu')(x)

preds=Dense(38,activation='softmax')(x)

model1=Model(inputs=base\_model.input,outputs=preds)

for layer in model1.layers[:20]:

layer.trainable=False

for layer in model1.layers[20:]:

layer.trainable=True

model1.compile(optimizer='Adam',loss='categorical\_crossentropy',metrics=METRICS)

step\_size\_train=train\_generator.n//train\_generator.batch\_size

model1\_history=model1.fit(train\_generator,

validation\_data =val\_generator,

steps\_per\_epoch=step\_size\_train,

epochs=10)

acc = model1\_history.history['accuracy']

val\_acc = model1\_history.history['val\_accuracy']

plt.plot(acc, label='Training Accuracy')

plt.plot(val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.ylabel('Accuracy')

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation Accuracy')

plt.xlabel('Epoch')

plt.show()

plt.savefig("model1\_Accuracy.png")

acc = model1\_history.history['accuracy']

val\_acc = model1\_history.history['val\_accuracy']

plt.plot(acc, label='Training Accuracy')

plt.plot(val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.ylabel('Accuracy')

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation Accuracy')

plt.xlabel('Epoch')

plt.show()

plt.savefig("model1\_Accuracy.png")

pre = model1\_history.history['precision']

val\_pre = model1\_history.history['val\_precision']

plt.plot(pre, label='Training Precision')

plt.plot(val\_pre, label='Validation Precision')

plt.legend(loc='lower right')

plt.ylabel("Precision")

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation Precision')

plt.xlabel('Epoch')

plt.show()

plt.savefig("model1\_Precision.png")

rec = model1\_history.history['recall']

val\_rec = model1\_history.history['val\_recall']

plt.plot(rec, label='Training Recall')

plt.plot(val\_rec, label='Validation Recall')

plt.legend(loc='lower right')

plt.ylabel("Recall")

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation Recall')

plt.xlabel('Epoch')

plt.show()

plt.savefig("model1\_Recall.png")

rec = model1\_history.history['recall']

val\_rec = model1\_history.history['val\_recall']

plt.plot(rec, label='Training Recall')

plt.plot(val\_rec, label='Validation Recall')

plt.legend(loc='lower right')

plt.ylabel("Recall")

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation Recall')

plt.xlabel('Epoch')

plt.show()

plt.savefig("model1\_Recall.png")

model1.save("InceptionResNetV211.h5")

"""# MobileNet"""

from tensorflow.keras.applications.mobilenet import preprocess\_input

train\_datagen=ImageDataGenerator(preprocessing\_function=preprocess\_input)

train\_generator=train\_datagen.flow\_from\_directory('./output/train',

target\_size=IMG\_SIZE,

color\_mode='rgb',

batch\_size=BATCH\_SIZE,

class\_mode='categorical',

shuffle=True)

val\_datagen=ImageDataGenerator(preprocessing\_function=preprocess\_input)

val\_generator=train\_datagen.flow\_from\_directory('./output/val',

target\_size=IMG\_SIZE,

color\_mode='rgb',

batch\_size=BATCH\_SIZE,

class\_mode='categorical',

shuffle=True)

base\_model=tf.keras.applications.MobileNet(input\_shape=IMG\_SHAPE, weights='imagenet',include\_top=False)

x=base\_model.output

x=GlobalAveragePooling2D()(x)

x=Dense(1024,activation='relu')(x)

x=Dense(1024,activation='relu')(x)

x=Dense(512,activation='relu')(x)

preds=Dense(38,activation='softmax')(x)

model2=Model(inputs=base\_model.input,outputs=preds)

for layer in model2.layers[:20]:

layer.trainable=False

for layer in model2.layers[20:]:

layer.trainable=True

model2.compile(optimizer='Adam',loss='categorical\_crossentropy',metrics=METRICS)

step\_size\_train=train\_generator.n//train\_generator.batch\_size

model2.fit(train\_generator,

validation\_data =val\_generator,

steps\_per\_epoch=step\_size\_train,

epochs=10)

acc = model2\_history.history['accuracy']

val\_acc = model2\_history.history['val\_accuracy']

plt.plot(acc, label='Training Accuracy')

plt.plot(val\_acc, label='Validation Accuracy')

plt.legend(loc='lower right')

plt.ylabel('Accuracy')

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation Accuracy')

plt.xlabel('Epoch')

plt.show()

loss = model2\_history.history['loss']

val\_loss = model2\_history.history['val\_loss']

plt.plot(loss, label='Training Loss')

plt.plot(val\_loss, label='Validation Loss')

plt.legend(loc='upper right')

plt.ylabel('Loss')

plt.xlim([0,9])

plt.ylim([0,max(plt.ylim())])

plt.title('Training and Validation Loss')

plt.xlabel('Epoch')

plt.show()

pre = model2\_history.history['precision']

val\_pre = model2\_history.history['val\_precision']

plt.plot(pre, label='Training Precision')

plt.plot(val\_pre, label='Validation Precision')

plt.legend(loc='lower right')

plt.ylabel("Precision")

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation Precision')

plt.xlabel('Epoch')

plt.show()

rec = model2\_history.history['recall']

val\_rec = model2\_history.history['val\_recall']

plt.plot(rec, label='Training Recall')

plt.plot(val\_rec, label='Validation Recall')

plt.legend(loc='lower right')

plt.ylabel("Recall")

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation Recall')

plt.xlabel('Epoch')

plt.show()

auc = model2\_history.history['auc']

val\_auc = model2\_history.history['val\_auc']

plt.plot(auc, label='Training AUC')

plt.plot(val\_auc, label='Validation AUC')

plt.legend(loc='lower right')

plt.ylabel("AUC")

plt.xlim([0,9])

plt.ylim([min(plt.ylim()),1])

plt.title('Training and Validation AUC')

plt.xlabel('Epoch')

plt.show()

**REFERENCES**

L. Han, M. S. Haleem, and M. Taylor, "A novel

computer vision-based approach to automatic detection

and severity assessment of crop diseases," in 2015

Science and Information Conference (SAl), 2015, pp.

638-644.

[2] FAO, "Declaration of the world summit on food

security," ed: World Summit on Food Security Rome,

2009, pp. 1-7.

[3] A. Cruz, Y. Ampatzidis, R. Pierro, A. Materazzi, A.

Panattoni, L. De Bellis, et aI., "Detection of grapevine

yellows symptoms in Vitis vinifera L. with artificial

intelligence," Computers and electronics in agriculture,

vol. 157, pp. 63-76, 2019.

[4] X. E. Pantazi, D. Moshou, and A. A. Tamouridou,

"Automated leaf disease detection in different crop

species through image features analysis and One Class

Classifiers," Computers and electronics in agriculture,

vol. 156, pp. 96-104,2019.

[5] S. Zhang, W. Huang, and C. Zhang, "Three-channel

convolutional neural networks for vegetable leaf

disease recognition," Cognitive Systems Research, vol.

53, pp. 31-41,2019.

[6] S. Zhang, X. Wu, Z. You, and L. Zhang, "Leaf image

based cucumber disease recognition using sparse

representation classification," Computers and

electronics in agriculture, vol. 134, pp. 135-141,2017.

[7] X. Zhang, Y. Qiao, F. Meng, C. Fan, and M. Zhang,

"Identification of maize leaf diseases using improved

deep convolutional neural networks," IEEEAccess, vol.

6,pp.30370-30377,2018.

[8] K. Singh, S. Kumar, and P. Kaur, "Support vector

machine classifier based detection offungal rust disease

in Pea Plant (Pisam sativam)," International Journal of

Information Technology, vol. 11, pp. 485-492, 2019.

[9] S. P. Mohanty, D. P. Hughes, and M. Salathe, "Using

deep learning for image-based plant disease detection,"

Frontiers in plantscience, vol. 7, p. 1419,2016.

[10] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, "A

comparative study offine-tuning deep learning models

for plant disease identification," Computers and

Electronics in Agriculture, vol. 161, pp. 272-279, 2019.

[11] 1. Lu, 1. Hu, G. Zhao, F. Mei, and C. Zhang, "An in-

field automatic wheat disease diagnosis system,"

Computers and electronics in agriculture, vol. 142, pp.

369-379,2017.

[12] H. Durmu~, E. O. Giine~, and M. KIfCl, "Disease

detection on the leaves of the tomato plants by using

deep learning," in 2017 6th International Conference on

Agro-Geoinformatics, 2017, pp. 1-5.

[13] 1. Shijie, 1. Peiyi, and H. Siping, "Automatic detection

of tomato diseases and pests based on leaf images," in

2017 Chinese Automation Congress (CAC), 2017, pp.

2537-2510.

[14] R. Zhou, S. i. Kaneko, F. Tanaka, M. Kayamori, andM.

Shimizu, "Disease detection of Cercospora Leaf Spot in

sugar beet by robust template matching," Computers

and electronics in agriculture, vol. 108, pp. 58-70,

2014.

[15] V. Singh and A. K. Misra, "Detection of plant leaf

diseases using image segmentation and soft computing

techniques," Information processing in Agriculture,

vol. 4, pp. 41-49, 2017.

[16] P. Revathi and M. Hemalatha, "Classification of cotton

leaf spot diseases using image processing edge

detection techniques," in 2012 International

Conference on Emerging Trends in Science,

Engineering and Technology (INCaSET), 2012, pp.

169-173.

[17] B. Liu, Y. Zhang, D. He, and Y. Li, "Identification of

apple leaf diseases based on deep convolutional neural

networks," Symmetry, vol. 10, p. 11,2018.

[18] G. Wang, Y. Sun, and 1. Wang, "Automatic image-

based plant disease severity estimation using deep

learning," Computational intelligence and

neuroscience, vol. 2017, 2017.

[19] A. Fuentes, S. Yoon, S. C. Kim, and D. S. Park, "A

robust deep-learning-based detector for real-time

tomato plant diseases and pests recognition," Sensors,

vol. 17,p.2022,2017.

[20] G. Hu, H. Wu, Y. Zhang, and M. Wan, "A low shot

learning method for tea leaf's disease identification,"

Computers and Electronics in Agriculture, vol. 163, p.

104852,2019.

[21] D. Hughes and M. Salathe, "An open access repository

of images on plant health to enable the development of

mobile disease diagnostics," arXiv preprint

arXiv:1511.08060, 2015.

[22] M. Brahimi, K. Boukhalfa, and A. Moussaoui, "Deep

learning for tomato diseases: classification and

symptoms visualization," Applied Artificial

Intelligence, vol. 31, pp. 299-315, 2017.

[23] K. P. Ferentinos, "Deep learning models for plant

disease detection and diagnosis," Computers and

Electronics in Agriculture, vol. 145, pp. 311-318,2018.

[24] Y. Li, H. Wang, L. M. Dang, A. Sadeghi-Niaraki, and

H. Moon, "Crop pest recognition in natural scenes using

convolutional neural networks," Computers and

Electronics in Agriculture, vol. 169, p. 105174,2020.

**BIODATA**

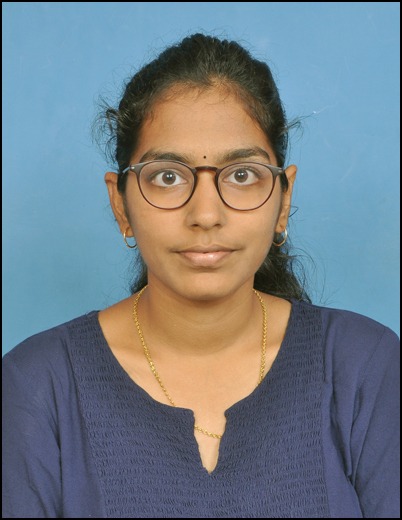


Name : Nittoor Vishnu Bharadwaj

Mobile Number : 9032220257

E-mail : vishnu.19bce7478@vitap.ac.in

Permanaent Address : 12-3-542 Jesus Nagar , Anantapur



Name : Bavita Moparthi

Mobile Number : 8106670359

E-mail : bavita.19bcn7016@vitap.ac.in

Permanaent Address : Guntur



Name : Popuri Harsha Vardhan

Mobile Number : 9100624101

E-mail : harsha.19bci7039@vitap.ac.in

Permanaent Address :Guntur