# Import Libraries

#Importing and Installing Required Modules and Libraries  
!pip install tensorflow  
!pip install basic\_image\_eda  
import os  
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib.colors as mcolors  
import seaborn as sns  
sns.set\_theme(style="whitegrid")  
import cv2  
import sys  
from re import sub  
import tensorflow as tf  
from tensorflow.keras import metrics  
from tensorflow.keras.models import Sequential  
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.optimizers import Adam  
from tensorflow.keras.preprocessing import image  
from tensorflow.keras import models  
from tensorflow.keras.preprocessing import image\_dataset\_from\_directory

OUTPUT

Requirement already satisfied: tensorflow in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (2.6.0)  
Requirement already satisfied: opt-einsum~=3.3.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (3.3.0)  
Requirement already satisfied: astunparse~=1.6.3 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (1.6.3)  
Requirement already satisfied: typing-extensions~=3.7.4 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (3.7.4.3)  
Requirement already satisfied: grpcio<2.0,>=1.37.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (1.39.0)  
Requirement already satisfied: keras-preprocessing~=1.1.2 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (1.1.2)  
Requirement already satisfied: numpy~=1.19.2 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (1.19.5)  
Requirement already satisfied: tensorflow-estimator~=2.6 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (2.6.0)  
Requirement already satisfied: h5py~=3.1.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (3.1.0)  
Requirement already satisfied: six~=1.15.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (1.15.0)  
Requirement already satisfied: clang~=5.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (5.0)  
Requirement already satisfied: wrapt~=1.12.1 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (1.12.1)  
Requirement already satisfied: wheel~=0.35 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (0.37.0)  
Requirement already satisfied: flatbuffers~=1.12.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (1.12)  
Requirement already satisfied: google-pasta~=0.2 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (0.2.0)  
Requirement already satisfied: gast==0.4.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (0.4.0)  
Requirement already satisfied: termcolor~=1.1.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (1.1.0)  
Requirement already satisfied: protobuf>=3.9.2 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (3.17.3)  
Requirement already satisfied: absl-py~=0.10 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (0.13.0)  
Requirement already satisfied: tensorboard~=2.6 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (2.6.0)  
Requirement already satisfied: keras~=2.6 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorflow) (2.6.0)  
Requirement already satisfied: google-auth<2,>=1.6.3 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorboard~=2.6->tensorflow) (1.35.0)  
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorboard~=2.6->tensorflow) (0.6.1)  
Requirement already satisfied: markdown>=2.6.8 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorboard~=2.6->tensorflow) (3.3.4)  
Requirement already satisfied: werkzeug>=0.11.15 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorboard~=2.6->tensorflow) (2.0.1)  
Requirement already satisfied: setuptools>=41.0.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorboard~=2.6->tensorflow) (56.0.0)  
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorboard~=2.6->tensorflow) (1.8.0)  
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorboard~=2.6->tensorflow) (0.4.5)  
Requirement already satisfied: requests<3,>=2.21.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from tensorboard~=2.6->tensorflow) (2.26.0)  
Requirement already satisfied: pyasn1-modules>=0.2.1 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from google-auth<2,>=1.6.3->tensorboard~=2.6->tensorflow) (0.2.8)  
Requirement already satisfied: rsa<5,>=3.1.4 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from google-auth<2,>=1.6.3->tensorboard~=2.6->tensorflow) (4.7.2)  
Requirement already satisfied: cachetools<5.0,>=2.0.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from google-auth<2,>=1.6.3->tensorboard~=2.6->tensorflow) (4.2.2)  
Requirement already satisfied: requests-oauthlib>=0.7.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow) (1.3.0)  
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from pyasn1-modules>=0.2.1->google-auth<2,>=1.6.3->tensorboard~=2.6->tensorflow) (0.4.8)  
Requirement already satisfied: certifi>=2017.4.17 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (2021.5.30)  
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (1.26.6)  
Requirement already satisfied: idna<4,>=2.5 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (3.2)  
Requirement already satisfied: charset-normalizer~=2.0.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from requests<3,>=2.21.0->tensorboard~=2.6->tensorflow) (2.0.4)  
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow) (3.1.1)  
Requirement already satisfied: basic\_image\_eda in c:\users\Abhishek\appdata\local\programs\python\python39\lib\site-packages (0.0.3)

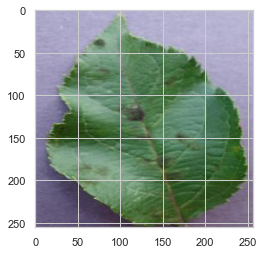
# Load Dataset

data\_dir = r"Downloads/plantvillage-dataset/color/"

# Sample Image

img = plt.imread(data\_dir+"Apple\_\_\_Apple\_scab/00075aa8-d81a-4184-8541-b692b78d398a\_\_\_FREC\_Scab 3335.JPG")  
plt.imshow(img)

<matplotlib.image.AxesImage at 0x1d4240aba00>



# Exploratory Data Analysis

category\_count = []  
  
for root, dirs, files in os.walk(data\_dir):  
 for dir\_path in dirs:  
 category\_count.append((dir\_path, len(os.listdir(root+os.sep+dir\_path))))  
count\_df = pd.DataFrame(category\_count, columns=['Category', 'Count'])  
count\_df.head(10)

Category Count  
0 Apple\_\_\_Apple\_scab 630  
1 Apple\_\_\_Black\_rot 621  
2 Apple\_\_\_Cedar\_apple\_rust 275  
3 Apple\_\_\_healthy 1645  
4 Blueberry\_\_\_healthy 1502  
5 Cherry\_(including\_sour)\_\_\_healthy 854  
6 Cherry\_(including\_sour)\_\_\_Powdery\_mildew 1052  
7 Corn\_(maize)\_\_\_Cercospora\_leaf\_spot Gray\_leaf\_... 513  
8 Corn\_(maize)\_\_\_Common\_rust\_ 1192  
9 Corn\_(maize)\_\_\_healthy 1162

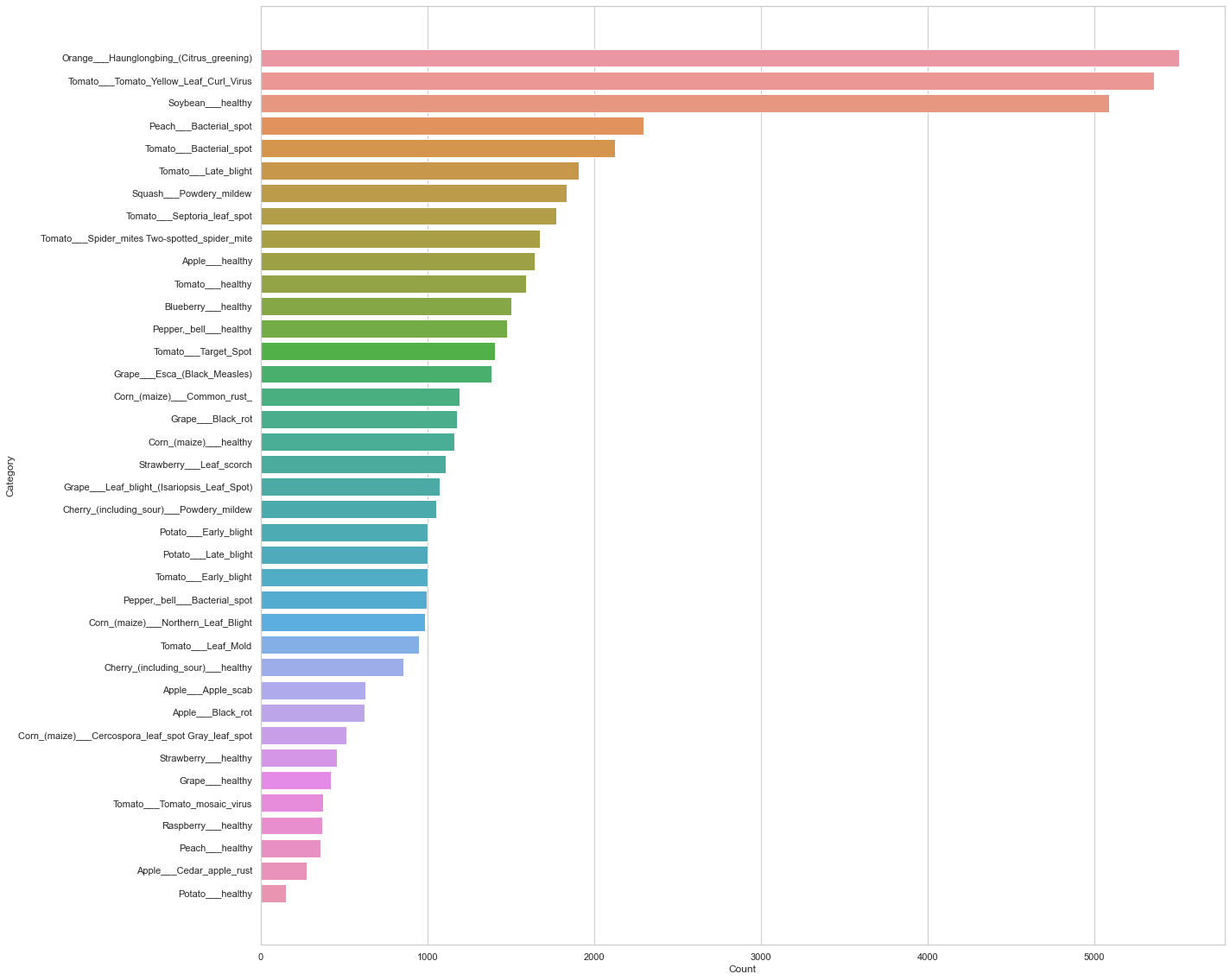
total\_images = count\_df['Count'].sum()  
total\_images

54305

# Data Visualization by Category

count\_df = count\_df.sort\_values(by='Count', ascending=False)  
plt.figure(figsize=(20,20))  
sns.barplot(x="Count", y="Category", data=count\_df)  
plt.plot()

[]



# Counting the Healthy Plants

healthy\_images\_count = count\_df[count\_df['Category'].str.endswith("healthy")]['Count'].sum()  
healthy\_images\_count

15084

# Counting he Unhealthy Plant by Substracting Healthy from Total Datasets

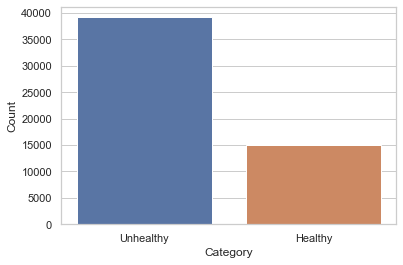
disease\_images\_count = total\_images - healthy\_images\_count  
disease\_images\_count

39221

# Ploting Graph of Healthy and Unhealthy Plants

temp\_df = pd.DataFrame(data=[("Unhealthy", disease\_images\_count), ("Healthy", healthy\_images\_count)], columns=['Category', 'Count'])  
sns.barplot(y="Count",x="Category", data=temp\_df)  
plt.plot()

[]



# Data Augmentation and Pre-processing

BATCH\_SIZE = 32  
IMG\_SIZE = (240, 240)

train\_dataset = image\_dataset\_from\_directory(data\_dir,  
 shuffle=True,  
 label\_mode = 'categorical',  
 validation\_split = 0.2,  
 batch\_size=BATCH\_SIZE,  
 seed = 42,  
 subset = "training",  
 image\_size=IMG\_SIZE)  
  
validation\_dataset = image\_dataset\_from\_directory(data\_dir,  
 shuffle=True,  
 label\_mode = 'categorical',  
 validation\_split = 0.2,  
 batch\_size=BATCH\_SIZE,  
 seed = 42,  
 subset = "validation",  
 image\_size=IMG\_SIZE)

Found 54305 files belonging to 38 classes.  
Using 43444 files for training.  
Found 54305 files belonging to 38 classes.  
Using 10861 files for validation.

class\_names = train\_dataset.class\_names  
num\_classes = len(class\_names)  
for i in range(1, num\_classes + 1):  
 print(str(i) + ". ", class\_names[i - 1])

1. Apple\_\_\_Apple\_scab  
2. Apple\_\_\_Black\_rot  
3. Apple\_\_\_Cedar\_apple\_rust  
4. Apple\_\_\_healthy  
5. Blueberry\_\_\_healthy  
6. Cherry\_(including\_sour)\_\_\_Powdery\_mildew  
7. Cherry\_(including\_sour)\_\_\_healthy  
8. Corn\_(maize)\_\_\_Cercospora\_leaf\_spot Gray\_leaf\_spot  
9. Corn\_(maize)\_\_\_Common\_rust\_  
10. Corn\_(maize)\_\_\_Northern\_Leaf\_Blight  
11. Corn\_(maize)\_\_\_healthy  
12. Grape\_\_\_Black\_rot  
13. Grape\_\_\_Esca\_(Black\_Measles)  
14. Grape\_\_\_Leaf\_blight\_(Isariopsis\_Leaf\_Spot)  
15. Grape\_\_\_healthy  
16. Orange\_\_\_Haunglongbing\_(Citrus\_greening)  
17. Peach\_\_\_Bacterial\_spot  
18. Peach\_\_\_healthy  
19. Pepper,\_bell\_\_\_Bacterial\_spot  
20. Pepper,\_bell\_\_\_healthy  
21. Potato\_\_\_Early\_blight  
22. Potato\_\_\_Late\_blight  
23. Potato\_\_\_healthy  
24. Raspberry\_\_\_healthy  
25. Soybean\_\_\_healthy  
26. Squash\_\_\_Powdery\_mildew  
27. Strawberry\_\_\_Leaf\_scorch  
28. Strawberry\_\_\_healthy  
29. Tomato\_\_\_Bacterial\_spot  
30. Tomato\_\_\_Early\_blight  
31. Tomato\_\_\_Late\_blight  
32. Tomato\_\_\_Leaf\_Mold  
33. Tomato\_\_\_Septoria\_leaf\_spot  
34. Tomato\_\_\_Spider\_mites Two-spotted\_spider\_mite  
35. Tomato\_\_\_Target\_Spot  
36. Tomato\_\_\_Tomato\_Yellow\_Leaf\_Curl\_Virus  
37. Tomato\_\_\_Tomato\_mosaic\_virus  
38. Tomato\_\_\_healthy

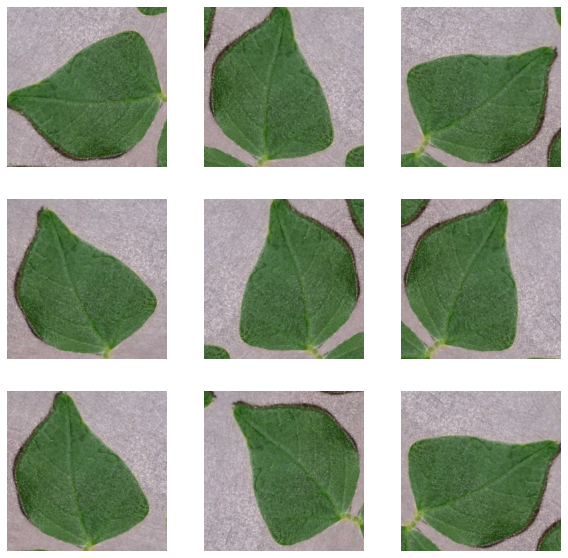
val\_batches = tf.data.experimental.cardinality(validation\_dataset)  
test\_dataset = validation\_dataset.take(val\_batches // 5)  
validation\_dataset = validation\_dataset.skip(val\_batches // 5)  
  
print('Number of validation batches: %d' % tf.data.experimental.cardinality(validation\_dataset))  
print('Number of test batches: %d' % tf.data.experimental.cardinality(test\_dataset))

Number of validation batches: 272  
Number of test batches: 68

AUTOTUNE = tf.data.AUTOTUNE  
train\_dataset = train\_dataset.prefetch(buffer\_size=AUTOTUNE)  
validation\_dataset = validation\_dataset.prefetch(buffer\_size=AUTOTUNE)  
test\_dataset = test\_dataset.prefetch(buffer\_size=AUTOTUNE)

# added augmentations  
data\_augmentation = tf.keras.Sequential([  
 tf.keras.layers.experimental.preprocessing.RandomFlip('horizontal'),  
 tf.keras.layers.experimental.preprocessing.RandomRotation(0.2),  
])

for image, \_ in train\_dataset.take(1):  
 plt.figure(figsize=(10, 10))  
 first\_image = image[0]  
 for i in range(9):  
 ax = plt.subplot(3, 3, i + 1)  
 augmented\_image = data\_augmentation(tf.expand\_dims(first\_image, 0))  
 plt.imshow(augmented\_image[0] / 255)  
 plt.axis('off')



## Metrics and Plotting functions

def plot\_metrics(history):  
 colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k', 'w']  
 metrics = ['loss', 'auc', 'precision', 'recall']  
 plt.figure(figsize=(20,10))  
 for n, metric in enumerate(metrics):  
 name = metric.replace("\_"," ").capitalize()  
 plt.subplot(2,2,n+1)  
 plt.plot(history.epoch, history.history[metric], color=colors[0], label='Train')  
 plt.plot(history.epoch, history.history['val\_'+metric],  
 color=colors[0], linestyle="--", label='Val')  
 plt.xlabel('Epoch')  
 plt.ylabel(name)  
 if metric == 'loss':  
 plt.ylim([0, plt.ylim()[1]])  
 elif metric == 'auc':  
 plt.ylim([0.8,1])  
 else:  
 plt.ylim([0,1])  
  
 plt.legend()

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')  
]

## Load and compile model

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

preprocess\_input = tf.keras.applications.inception\_resnet\_v2.preprocess\_input

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

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception\_resnet\_v2/inception\_resnet\_v2\_weights\_tf\_dim\_ordering\_tf\_kernels\_notop.h5  
219062272/219055592 [==============================] - 1s 0us/step  
219070464/219055592 [==============================] - 1s 0us/step

image\_batch, label\_batch = next(iter(train\_dataset))  
feature\_batch = base\_model(image\_batch)  
print(feature\_batch.shape)

(32, 6, 6, 1536)

base\_model.trainable = False

global\_average\_layer = tf.keras.layers.GlobalAveragePooling2D()  
feature\_batch\_average = global\_average\_layer(feature\_batch)  
  
print(feature\_batch\_average.shape)

(32, 1536)

prediction\_layer = tf.keras.layers.Dense(num\_classes, activation="softmax")  
prediction\_batch = prediction\_layer(feature\_batch\_average)  
print(prediction\_batch.shape)

(32, 38)

inputs = tf.keras.Input(shape=(240, 240, 3))  
x = data\_augmentation(inputs)  
x = preprocess\_input(x)  
x = base\_model(x, training=False)  
x = global\_average\_layer(x)  
x = tf.keras.layers.Dropout(0.2)(x)  
outputs = prediction\_layer(x)  
model = tf.keras.Model(inputs, outputs)

base\_learning\_rate = 0.001  
model.compile(optimizer=tf.keras.optimizers.Adam(lr=base\_learning\_rate),  
 loss=tf.keras.losses.CategoricalCrossentropy(from\_logits=True),  
 metrics=METRICS)

C:\Users\Abhishek\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\optimizer\_v2\optimizer\_v2.py:355: UserWarning: The `lr` argument is deprecated, use `learning\_rate` instead.  
 warnings.warn(

model.summary()

Model: "model"  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Layer (type) Output Shape Param #   
=================================================================  
input\_2 (InputLayer) [(None, 240, 240, 3)] 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
sequential (Sequential) (None, 240, 240, 3) 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
tf.math.truediv (TFOpLambda) (None, 240, 240, 3) 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
tf.math.subtract (TFOpLambda (None, 240, 240, 3) 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
inception\_resnet\_v2 (Functio (None, 6, 6, 1536) 54336736   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
global\_average\_pooling2d (Gl (None, 1536) 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dropout (Dropout) (None, 1536) 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense (Dense) (None, 38) 58406   
=================================================================  
Total params: 54,395,142  
Trainable params: 58,406  
Non-trainable params: 54,336,736  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Training, Validation and Testing

Before tuning

initial\_epochs = 10

history = model.fit(train\_dataset,  
 epochs=initial\_epochs,  
 validation\_data=validation\_dataset)

Epoch 1/10

C:\Users\Abhishek\AppData\Local\Programs\Python\Python39\lib\site-packages\keras\backend.py:4846: UserWarning: "`categorical\_crossentropy` received `from\_logits=True`, but the `output` argument was produced by a sigmoid or softmax activation and thus does not represent logits. Was this intended?"  
 warnings.warn(

1358/1358 [==============================] - 3620s 3s/step - loss: 0.8491 - tp: 27133.0000 - fp: 2299.0000 - tn: 1605129.0000 - fn: 16311.0000 - accuracy: 0.7646 - precision: 0.9219 - recall: 0.6246 - auc: 0.9872 - val\_loss: 0.4727 - val\_tp: 6783.0000 - val\_fp: 552.0000 - val\_tn: 320793.0000 - val\_fn: 1902.0000 - val\_accuracy: 0.8558 - val\_precision: 0.9247 - val\_recall: 0.7810 - val\_auc: 0.9963  
Epoch 2/10  
1358/1358 [==============================] - 3638s 3s/step - loss: 0.4329 - tp: 35232.0000 - fp: 2742.0000 - tn: 1604686.0000 - fn: 8212.0000 - accuracy: 0.8704 - precision: 0.9278 - recall: 0.8110 - auc: 0.9959 - val\_loss: 0.3538 - val\_tp: 7359.0000 - val\_fp: 482.0000 - val\_tn: 320863.0000 - val\_fn: 1326.0000 - val\_accuracy: 0.8940 - val\_precision: 0.9385 - val\_recall: 0.8473 - val\_auc: 0.9975  
Epoch 3/10  
1358/1358 [==============================] - 3650s 3s/step - loss: 0.3622 - tp: 36849.0000 - fp: 2775.0000 - tn: 1604653.0000 - fn: 6595.0000 - accuracy: 0.8884 - precision: 0.9300 - recall: 0.8482 - auc: 0.9967 - val\_loss: 0.2919 - val\_tp: 7650.0000 - val\_fp: 448.0000 - val\_tn: 320897.0000 - val\_fn: 1035.0000 - val\_accuracy: 0.9108 - val\_precision: 0.9447 - val\_recall: 0.8808 - val\_auc: 0.9979  
Epoch 4/10  
1358/1358 [==============================] - 3660s 3s/step - loss: 0.3282 - tp: 37482.0000 - fp: 2765.0000 - tn: 1604663.0000 - fn: 5962.0000 - accuracy: 0.8968 - precision: 0.9313 - recall: 0.8628 - auc: 0.9969 - val\_loss: 0.2575 - val\_tp: 7722.0000 - val\_fp: 445.0000 - val\_tn: 320900.0000 - val\_fn: 963.0000 - val\_accuracy: 0.9162 - val\_precision: 0.9455 - val\_recall: 0.8891 - val\_auc: 0.9983  
Epoch 5/10  
1358/1358 [==============================] - 3684s 3s/step - loss: 0.3023 - tp: 38110.0000 - fp: 2666.0000 - tn: 1604762.0000 - fn: 5334.0000 - accuracy: 0.9038 - precision: 0.9346 - recall: 0.8772 - auc: 0.9969 - val\_loss: 0.2475 - val\_tp: 7813.0000 - val\_fp: 414.0000 - val\_tn: 320931.0000 - val\_fn: 872.0000 - val\_accuracy: 0.9239 - val\_precision: 0.9497 - val\_recall: 0.8996 - val\_auc: 0.9979  
Epoch 6/10  
1358/1358 [==============================] - 3653s 3s/step - loss: 0.2900 - tp: 38290.0000 - fp: 2682.0000 - tn: 1604746.0000 - fn: 5154.0000 - accuracy: 0.9066 - precision: 0.9345 - recall: 0.8814 - auc: 0.9972 - val\_loss: 0.2138 - val\_tp: 7911.0000 - val\_fp: 395.0000 - val\_tn: 320950.0000 - val\_fn: 774.0000 - val\_accuracy: 0.9308 - val\_precision: 0.9524 - val\_recall: 0.9109 - val\_auc: 0.9985  
Epoch 7/10  
1358/1358 [==============================] - 3658s 3s/step - loss: 0.2796 - tp: 38590.0000 - fp: 2675.0000 - tn: 1604753.0000 - fn: 4854.0000 - accuracy: 0.9100 - precision: 0.9352 - recall: 0.8883 - auc: 0.9969 - val\_loss: 0.2114 - val\_tp: 7898.0000 - val\_fp: 402.0000 - val\_tn: 320943.0000 - val\_fn: 787.0000 - val\_accuracy: 0.9308 - val\_precision: 0.9516 - val\_recall: 0.9094 - val\_auc: 0.9988  
Epoch 8/10  
1358/1358 [==============================] - 3640s 3s/step - loss: 0.2718 - tp: 38742.0000 - fp: 2683.0000 - tn: 1604745.0000 - fn: 4702.0000 - accuracy: 0.9112 - precision: 0.9352 - recall: 0.8918 - auc: 0.9971 - val\_loss: 0.2089 - val\_tp: 7940.0000 - val\_fp: 390.0000 - val\_tn: 320955.0000 - val\_fn: 745.0000 - val\_accuracy: 0.9337 - val\_precision: 0.9532 - val\_recall: 0.9142 - val\_auc: 0.9984  
Epoch 9/10  
1358/1358 [==============================] - 3630s 3s/step - loss: 0.2665 - tp: 38808.0000 - fp: 2756.0000 - tn: 1604672.0000 - fn: 4636.0000 - accuracy: 0.9123 - precision: 0.9337 - recall: 0.8933 - auc: 0.9972 - val\_loss: 0.2334 - val\_tp: 7872.0000 - val\_fp: 467.0000 - val\_tn: 320878.0000 - val\_fn: 813.0000 - val\_accuracy: 0.9247 - val\_precision: 0.9440 - val\_recall: 0.9064 - val\_auc: 0.9977  
Epoch 10/10  
1358/1358 [==============================] - 3655s 3s/step - loss: 0.2687 - tp: 38918.0000 - fp: 2738.0000 - tn: 1604690.0000 - fn: 4526.0000 - accuracy: 0.9139 - precision: 0.9343 - recall: 0.8958 - auc: 0.9968 - val\_loss: 0.2161 - val\_tp: 7949.0000 - val\_fp: 425.0000 - val\_tn: 320920.0000 - val\_fn: 736.0000 - val\_accuracy: 0.9313 - val\_precision: 0.9492 - val\_recall: 0.9153 - val\_auc: 0.9974

After tuning

base\_model.trainable = True

# Let's take a look to see how many layers are in the base model  
print("Number of layers in the base model: ", len(base\_model.layers))  
  
# Fine-tune from this layer onwards  
fine\_tune\_at = 700  
  
# Freeze all the layers before the `fine\_tune\_at` layer  
for layer in base\_model.layers[:fine\_tune\_at]:  
 layer.trainable = False

Number of layers in the base model: 780

fine\_tuning\_learning\_rate = 1e-5  
model.compile(optimizer=tf.keras.optimizers.Adam(lr=fine\_tuning\_learning\_rate),  
 loss=tf.keras.losses.CategoricalCrossentropy(from\_logits=True),  
 metrics=METRICS)

model.summary()

Model: "model"  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Layer (type) Output Shape Param #   
=================================================================  
input\_2 (InputLayer) [(None, 240, 240, 3)] 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
sequential (Sequential) (None, 240, 240, 3) 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
tf.math.truediv (TFOpLambda) (None, 240, 240, 3) 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
tf.math.subtract (TFOpLambda (None, 240, 240, 3) 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
inception\_resnet\_v2 (Functio (None, 6, 6, 1536) 54336736   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
global\_average\_pooling2d (Gl (None, 1536) 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dropout (Dropout) (None, 1536) 0   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
dense (Dense) (None, 38) 58406   
=================================================================  
Total params: 54,395,142  
Trainable params: 13,028,070  
Non-trainable params: 41,367,072  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

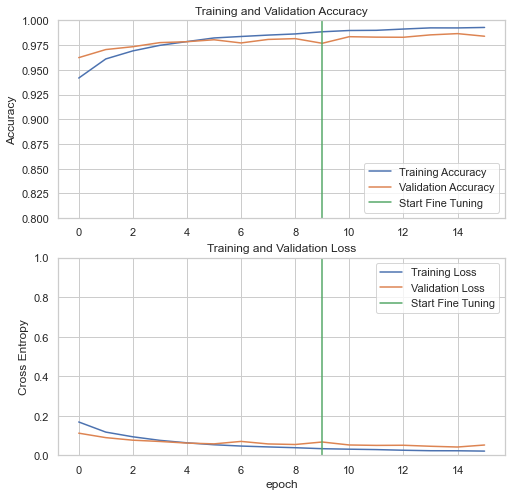
len(model.trainable\_variables)

52

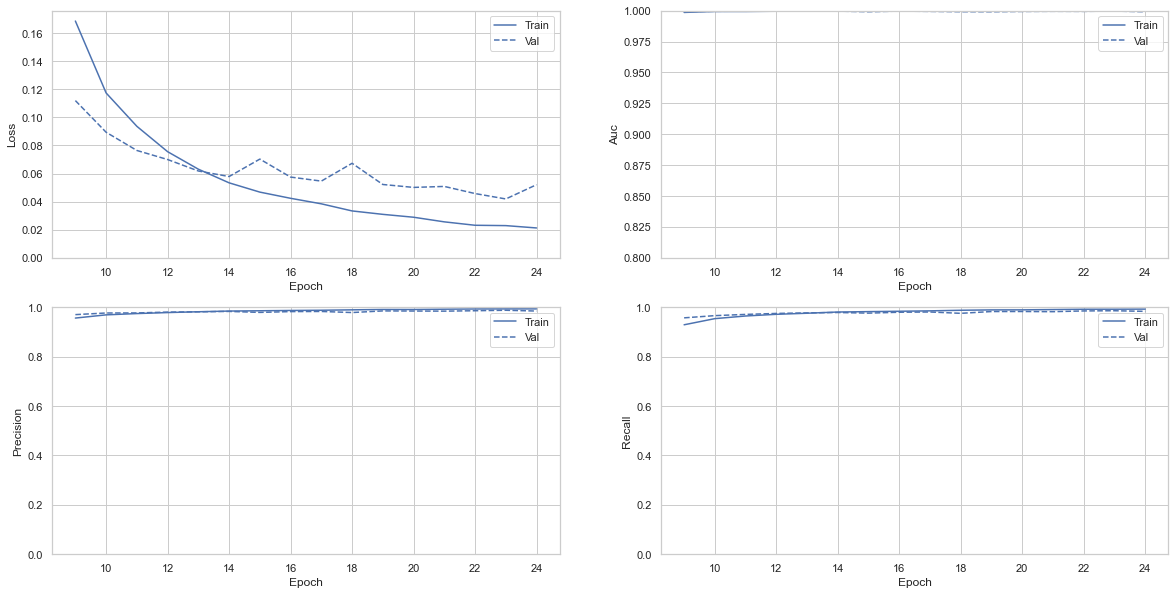
fine\_tune\_epochs = 15  
total\_epochs = initial\_epochs + fine\_tune\_epochs  
  
history\_fine = model.fit(train\_dataset,  
 epochs=total\_epochs,  
 initial\_epoch=history.epoch[-1],  
 validation\_data=validation\_dataset)

Epoch 10/25  
1358/1358 [==============================] - 4612s 3s/step - loss: 0.1686 - tp: 48421.0000 - fp: 2225.0000 - tn: 1926548.0000 - fn: 3708.0000 - accuracy: 0.9417 - precision: 0.9561 - recall: 0.9289 - auc: 0.9986 - val\_loss: 0.1119 - val\_tp: 8311.0000 - val\_fp: 257.0000 - val\_tn: 321088.0000 - val\_fn: 374.0000 - val\_accuracy: 0.9623 - val\_precision: 0.9700 - val\_recall: 0.9569 - val\_auc: 0.9994  
Epoch 11/25  
1358/1358 [==============================] - 4582s 3s/step - loss: 0.1174 - tp: 41435.0000 - fp: 1324.0000 - tn: 1606104.0000 - fn: 2009.0000 - accuracy: 0.9610 - precision: 0.9690 - recall: 0.9538 - auc: 0.9994 - val\_loss: 0.0894 - val\_tp: 8389.0000 - val\_fp: 201.0000 - val\_tn: 321144.0000 - val\_fn: 296.0000 - val\_accuracy: 0.9705 - val\_precision: 0.9766 - val\_recall: 0.9659 - val\_auc: 0.9995  
Epoch 12/25  
1358/1358 [==============================] - 4581s 3s/step - loss: 0.0936 - tp: 41898.0000 - fp: 1106.0000 - tn: 1606322.0000 - fn: 1546.0000 - accuracy: 0.9692 - precision: 0.9743 - recall: 0.9644 - auc: 0.9994 - val\_loss: 0.0764 - val\_tp: 8432.0000 - val\_fp: 200.0000 - val\_tn: 321145.0000 - val\_fn: 253.0000 - val\_accuracy: 0.9734 - val\_precision: 0.9768 - val\_recall: 0.9709 - val\_auc: 0.9996  
Epoch 13/25  
1358/1358 [==============================] - 4591s 3s/step - loss: 0.0756 - tp: 42196.0000 - fp: 936.0000 - tn: 1606492.0000 - fn: 1248.0000 - accuracy: 0.9748 - precision: 0.9783 - recall: 0.9713 - auc: 0.9996 - val\_loss: 0.0700 - val\_tp: 8466.0000 - val\_fp: 171.0000 - val\_tn: 321174.0000 - val\_fn: 219.0000 - val\_accuracy: 0.9775 - val\_precision: 0.9802 - val\_recall: 0.9748 - val\_auc: 0.9996  
Epoch 14/25  
1358/1358 [==============================] - 4612s 3s/step - loss: 0.0631 - tp: 42385.0000 - fp: 794.0000 - tn: 1606634.0000 - fn: 1059.0000 - accuracy: 0.9786 - precision: 0.9816 - recall: 0.9756 - auc: 0.9997 - val\_loss: 0.0618 - val\_tp: 8486.0000 - val\_fp: 161.0000 - val\_tn: 321184.0000 - val\_fn: 199.0000 - val\_accuracy: 0.9785 - val\_precision: 0.9814 - val\_recall: 0.9771 - val\_auc: 0.9997  
Epoch 15/25  
1358/1358 [==============================] - 4636s 3s/step - loss: 0.0534 - tp: 42598.0000 - fp: 675.0000 - tn: 1606753.0000 - fn: 846.0000 - accuracy: 0.9823 - precision: 0.9844 - recall: 0.9805 - auc: 0.9998 - val\_loss: 0.0579 - val\_tp: 8499.0000 - val\_fp: 147.0000 - val\_tn: 321198.0000 - val\_fn: 186.0000 - val\_accuracy: 0.9804 - val\_precision: 0.9830 - val\_recall: 0.9786 - val\_auc: 0.9997  
Epoch 16/25  
1358/1358 [==============================] - 4634s 3s/step - loss: 0.0468 - tp: 42669.0000 - fp: 622.0000 - tn: 1606806.0000 - fn: 775.0000 - accuracy: 0.9837 - precision: 0.9856 - recall: 0.9822 - auc: 0.9998 - val\_loss: 0.0703 - val\_tp: 8477.0000 - val\_fp: 182.0000 - val\_tn: 321163.0000 - val\_fn: 208.0000 - val\_accuracy: 0.9772 - val\_precision: 0.9790 - val\_recall: 0.9761 - val\_auc: 0.9992  
Epoch 17/25  
1358/1358 [==============================] - 4655s 3s/step - loss: 0.0424 - tp: 42733.0000 - fp: 570.0000 - tn: 1606858.0000 - fn: 711.0000 - accuracy: 0.9852 - precision: 0.9868 - recall: 0.9836 - auc: 0.9998 - val\_loss: 0.0575 - val\_tp: 8508.0000 - val\_fp: 154.0000 - val\_tn: 321191.0000 - val\_fn: 177.0000 - val\_accuracy: 0.9808 - val\_precision: 0.9822 - val\_recall: 0.9796 - val\_auc: 0.9997  
Epoch 18/25  
1358/1358 [==============================] - 4669s 3s/step - loss: 0.0384 - tp: 42804.0000 - fp: 530.0000 - tn: 1606898.0000 - fn: 640.0000 - accuracy: 0.9863 - precision: 0.9878 - recall: 0.9853 - auc: 0.9999 - val\_loss: 0.0546 - val\_tp: 8517.0000 - val\_fp: 149.0000 - val\_tn: 321196.0000 - val\_fn: 168.0000 - val\_accuracy: 0.9816 - val\_precision: 0.9828 - val\_recall: 0.9807 - val\_auc: 0.9994  
Epoch 19/25  
1358/1358 [==============================] - 4668s 3s/step - loss: 0.0334 - tp: 42908.0000 - fp: 461.0000 - tn: 1606967.0000 - fn: 536.0000 - accuracy: 0.9885 - precision: 0.9894 - recall: 0.9877 - auc: 0.9999 - val\_loss: 0.0673 - val\_tp: 8470.0000 - val\_fp: 187.0000 - val\_tn: 321158.0000 - val\_fn: 215.0000 - val\_accuracy: 0.9769 - val\_precision: 0.9784 - val\_recall: 0.9752 - val\_auc: 0.9992  
Epoch 20/25  
1358/1358 [==============================] - 4661s 3s/step - loss: 0.0309 - tp: 42971.0000 - fp: 395.0000 - tn: 1607033.0000 - fn: 473.0000 - accuracy: 0.9898 - precision: 0.9909 - recall: 0.9891 - auc: 0.9998 - val\_loss: 0.0522 - val\_tp: 8533.0000 - val\_fp: 130.0000 - val\_tn: 321215.0000 - val\_fn: 152.0000 - val\_accuracy: 0.9835 - val\_precision: 0.9850 - val\_recall: 0.9825 - val\_auc: 0.9991  
Epoch 21/25  
1358/1358 [==============================] - 4654s 3s/step - loss: 0.0289 - tp: 42979.0000 - fp: 401.0000 - tn: 1607027.0000 - fn: 465.0000 - accuracy: 0.9900 - precision: 0.9908 - recall: 0.9893 - auc: 0.9999 - val\_loss: 0.0501 - val\_tp: 8534.0000 - val\_fp: 134.0000 - val\_tn: 321211.0000 - val\_fn: 151.0000 - val\_accuracy: 0.9831 - val\_precision: 0.9845 - val\_recall: 0.9826 - val\_auc: 0.9994  
Epoch 22/25  
1358/1358 [==============================] - 4656s 3s/step - loss: 0.0256 - tp: 43035.0000 - fp: 350.0000 - tn: 1607078.0000 - fn: 409.0000 - accuracy: 0.9913 - precision: 0.9919 - recall: 0.9906 - auc: 0.9999 - val\_loss: 0.0508 - val\_tp: 8528.0000 - val\_fp: 138.0000 - val\_tn: 321207.0000 - val\_fn: 157.0000 - val\_accuracy: 0.9830 - val\_precision: 0.9841 - val\_recall: 0.9819 - val\_auc: 0.9995  
Epoch 23/25  
1358/1358 [==============================] - 4664s 3s/step - loss: 0.0232 - tp: 43090.0000 - fp: 310.0000 - tn: 1607118.0000 - fn: 354.0000 - accuracy: 0.9925 - precision: 0.9929 - recall: 0.9919 - auc: 0.9999 - val\_loss: 0.0457 - val\_tp: 8554.0000 - val\_fp: 124.0000 - val\_tn: 321221.0000 - val\_fn: 131.0000 - val\_accuracy: 0.9854 - val\_precision: 0.9857 - val\_recall: 0.9849 - val\_auc: 0.9994  
Epoch 24/25  
1358/1358 [==============================] - 4670s 3s/step - loss: 0.0229 - tp: 43093.0000 - fp: 299.0000 - tn: 1607129.0000 - fn: 351.0000 - accuracy: 0.9924 - precision: 0.9931 - recall: 0.9919 - auc: 0.9999 - val\_loss: 0.0419 - val\_tp: 8561.0000 - val\_fp: 107.0000 - val\_tn: 321238.0000 - val\_fn: 124.0000 - val\_accuracy: 0.9866 - val\_precision: 0.9877 - val\_recall: 0.9857 - val\_auc: 0.9997  
Epoch 25/25  
1358/1358 [==============================] - 4667s 3s/step - loss: 0.0212 - tp: 43111.0000 - fp: 295.0000 - tn: 1607133.0000 - fn: 333.0000 - accuracy: 0.9929 - precision: 0.9932 - recall: 0.9923 - auc: 0.9999 - val\_loss: 0.0521 - val\_tp: 8539.0000 - val\_fp: 135.0000 - val\_tn: 321210.0000 - val\_fn: 146.0000 - val\_accuracy: 0.9840 - val\_precision: 0.9844 - val\_recall: 0.9832 - val\_auc: 0.9992

acc = []   
val\_acc = []  
loss = []   
val\_loss = []  
  
acc += history\_fine.history['accuracy']  
val\_acc += history\_fine.history['val\_accuracy']  
  
loss += history\_fine.history['loss']  
val\_loss += history\_fine.history['val\_loss']  
  
  
  
plt.figure(figsize=(8, 8))  
plt.subplot(2, 1, 1)  
plt.plot(acc, label='Training Accuracy')  
plt.plot(val\_acc, label='Validation Accuracy')  
plt.ylim([0.8, 1])  
plt.plot([initial\_epochs-1,initial\_epochs-1],  
 plt.ylim(), label='Start Fine Tuning')  
plt.legend(loc='lower right')  
plt.title('Training and Validation Accuracy')  
plt.ylabel('Accuracy')  
  
plt.subplot(2, 1, 2)  
plt.plot(loss, label='Training Loss')  
plt.plot(val\_loss, label='Validation Loss')  
plt.ylim([0, 1.0])  
plt.plot([initial\_epochs-1,initial\_epochs-1],  
 plt.ylim(), label='Start Fine Tuning')  
plt.legend(loc='upper right')  
plt.title('Training and Validation Loss')  
plt.xlabel('epoch')  
plt.ylabel('Cross Entropy')  
plt.show()



plot\_metrics(history\_fine)



result = model.evaluate(test\_dataset)

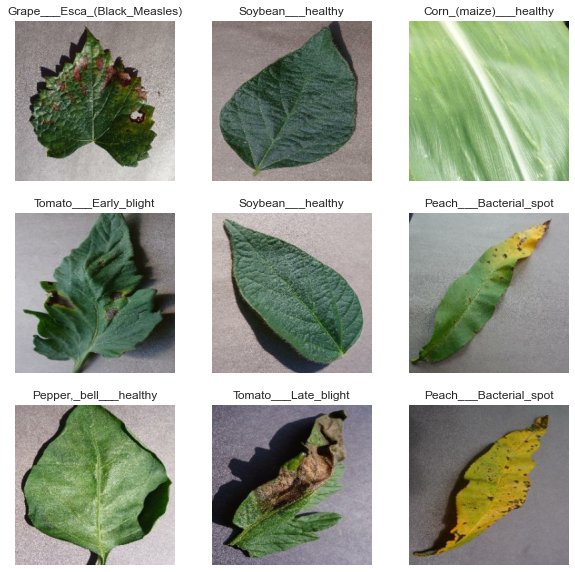
68/68 [==============================] - 151s 2s/step - loss: 0.0483 - tp: 2143.0000 - fp: 30.0000 - tn: 80482.0000 - fn: 33.0000 - accuracy: 0.9853 - precision: 0.9862 - recall: 0.9848 - auc: 0.9995

metrics = ["loss", "tp", "fp", "tn", "fn", "accuracy", "precision", "recall", "auc"]  
for i in range(len(result)):  
 print("{} : {}".format(metrics[i],round(result[i], 3)))

loss : 0.048  
tp : 2143.0  
fp : 30.0  
tn : 80482.0  
fn : 33.0  
accuracy : 0.985  
precision : 0.986  
recall : 0.985  
auc : 1.0

#Retrieve a batch of images from the test set  
image\_batch, label\_batch = test\_dataset.as\_numpy\_iterator().next()  
predictions = model.predict\_on\_batch(image\_batch)  
  
predictions = tf.nn.softmax(predictions)  
predictions = list(np.argmax(x) for x in predictions.numpy())  
  
print('Predictions:\n', predictions)  
print('Labels:\n', list(np.argmax(x) for x in label\_batch))  
  
plt.figure(figsize=(10, 10))  
for i in range(9):  
 ax = plt.subplot(3, 3, i + 1)  
 plt.imshow(image\_batch[i].astype("uint8"))  
 plt.title(class\_names[predictions[i]])  
 plt.axis("off")

Predictions:  
 [12, 24, 10, 29, 24, 16, 19, 30, 16, 29, 35, 11, 31, 35, 11, 4, 24, 16, 11, 21, 35, 16, 16, 16, 35, 0, 4, 5, 4, 13, 35, 33]  
Labels:  
 [12, 24, 10, 29, 24, 16, 19, 30, 16, 29, 35, 11, 31, 35, 11, 4, 24, 16, 11, 21, 35, 16, 16, 16, 35, 0, 4, 5, 4, 13, 35, 33]



# Save Model

# save model in JSON format  
model\_json = model.to\_json()  
json\_file = open("model\_weights.json", "w")  
json\_file.write(model\_json)  
print("Model saved in JSON format!")  
   
# save training weights in h5 file  
model.save\_weights("model\_weights.h5")  
print("\nModel weights saved!")

Model saved in JSON format!  
  
Model weights saved!

model.save("inception\_V3.0\_fineTuning.h5")