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)
```

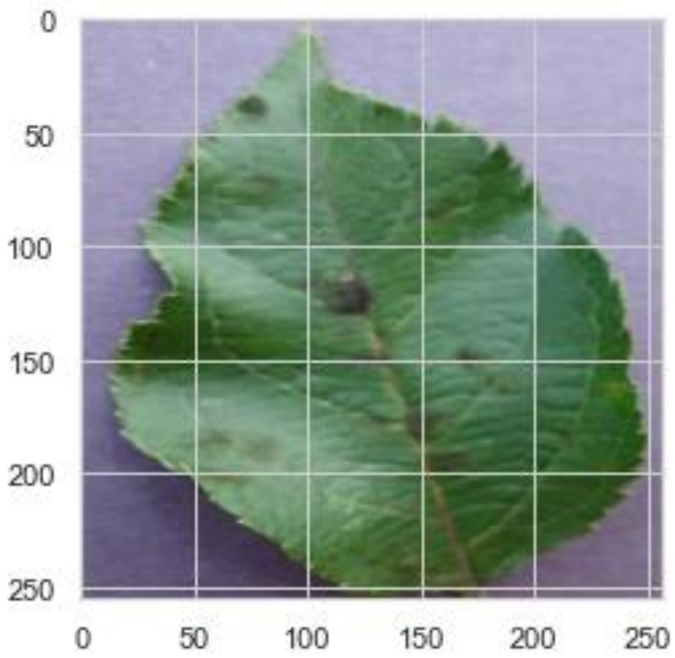
Load Dataset

```
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

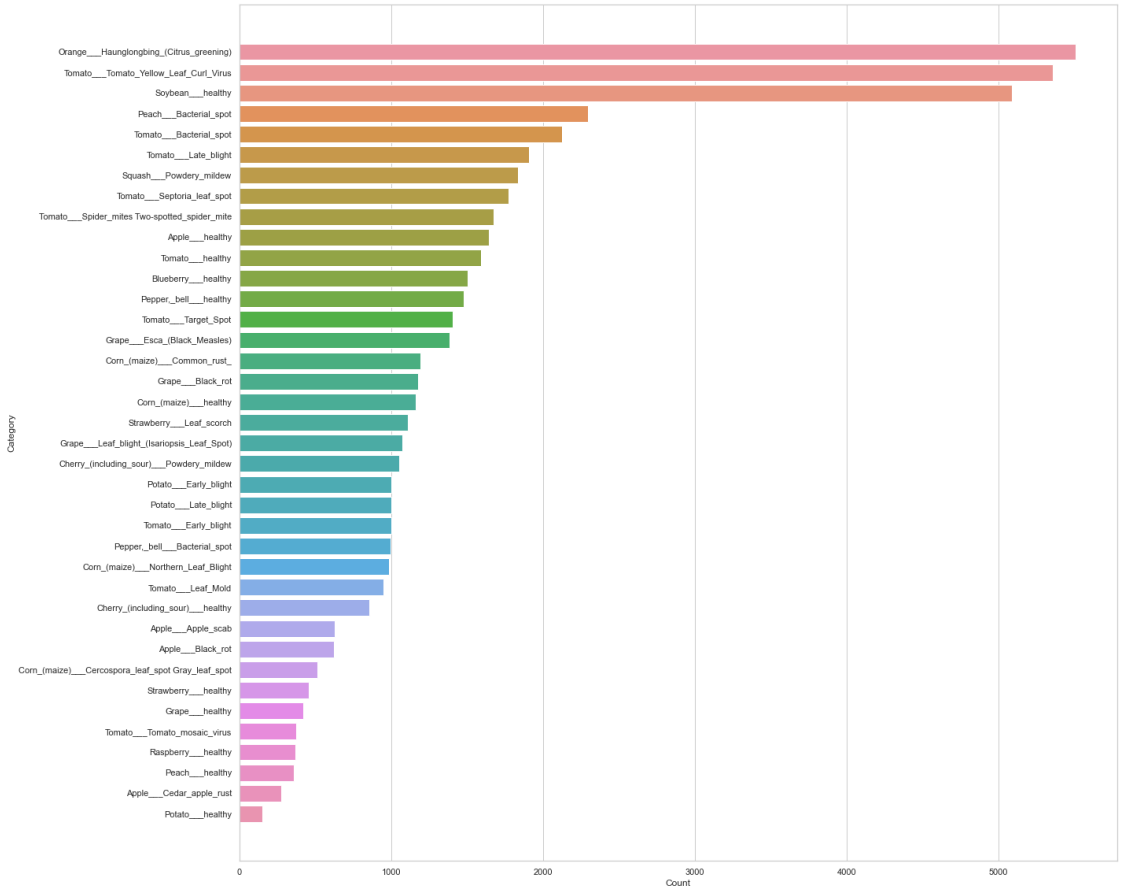
```
total_images = count_df['Count'].sum()
total_images
```

```
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

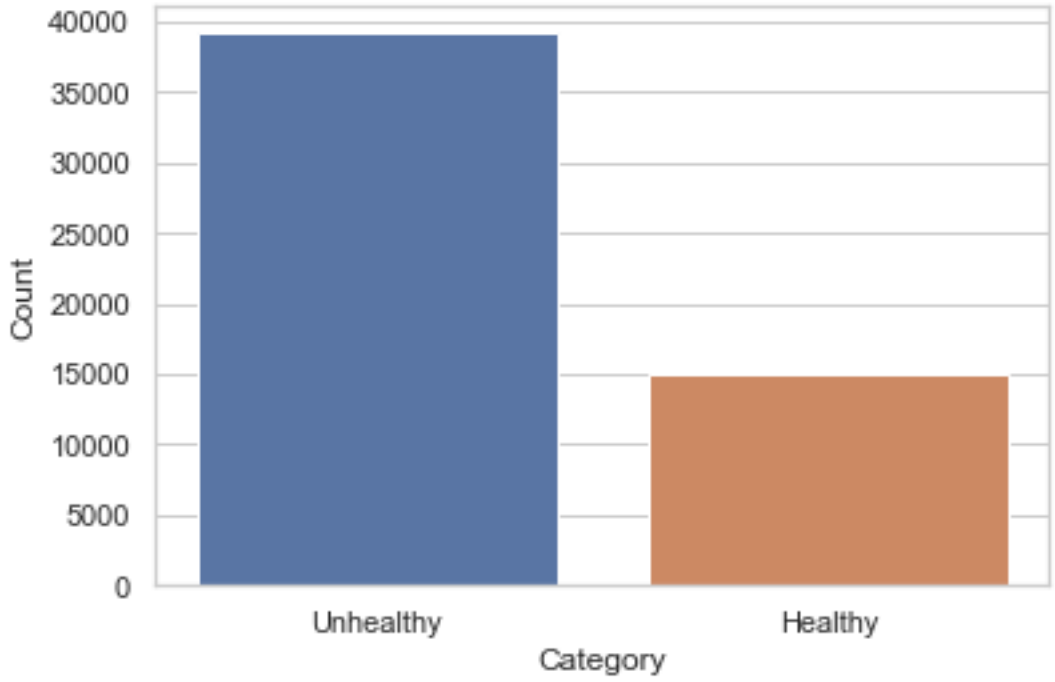
```
disease_images_count = total_images - healthy_images_count
disease_images_count
```

```
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)
```

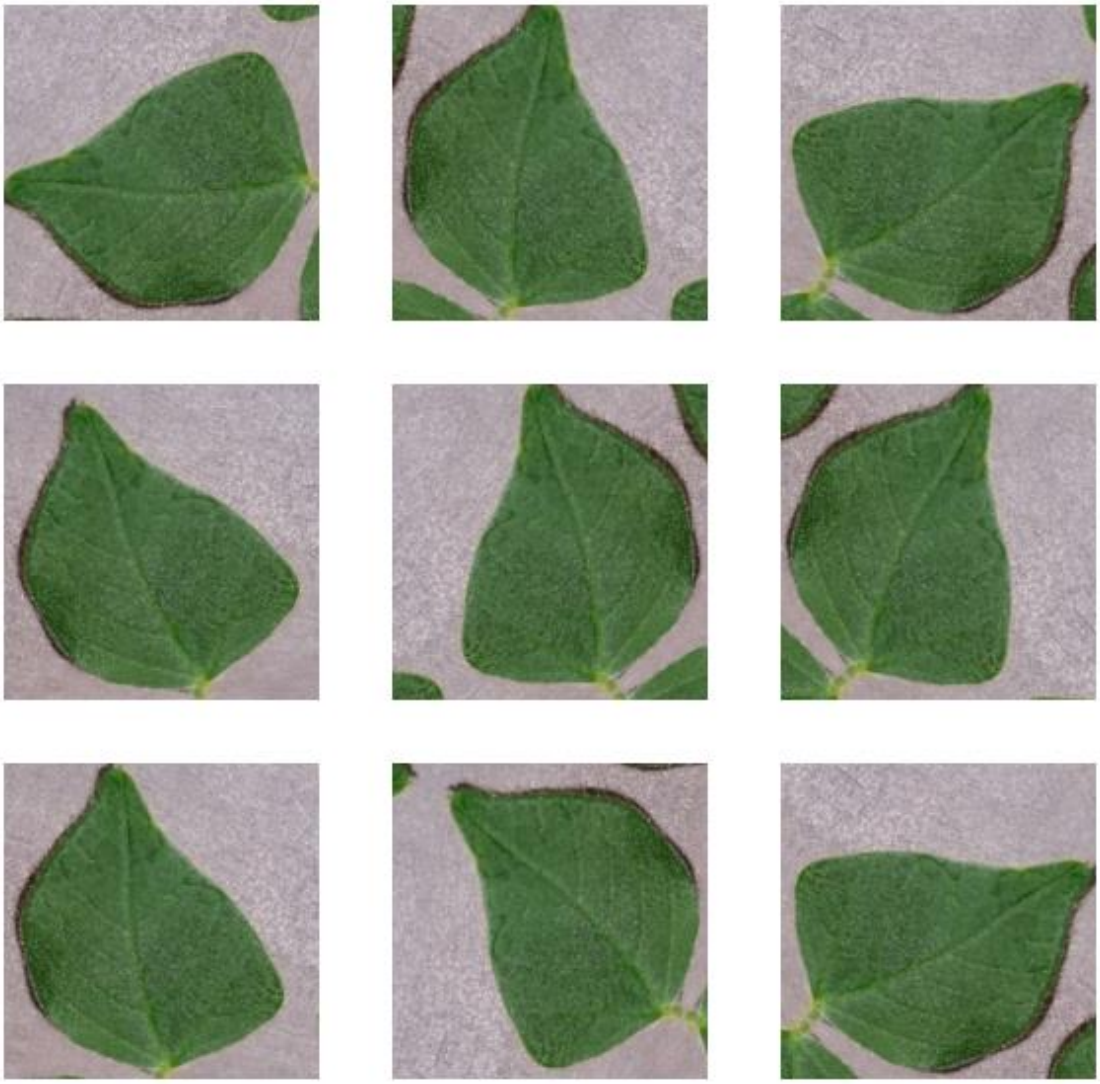
```
print('Number of validation batches: %d' %
tf.data.experimental.cardinality(validation_dataset))
print('Number of test batches: %d' %
tf.data.experimental.cardinality(test_dataset))
```

```
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)
```

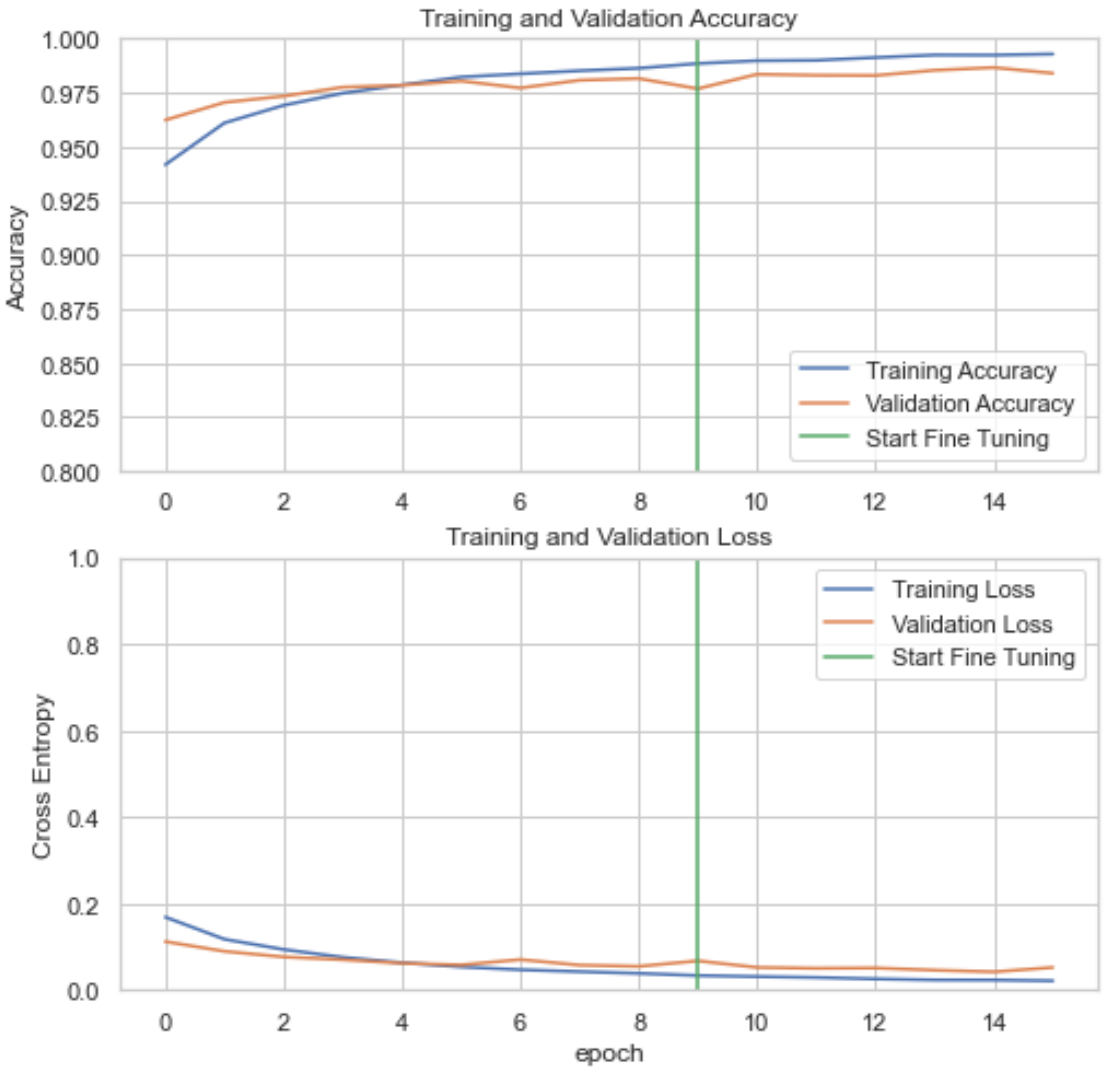
```
Epoch 10/25
1358/1358 [==============================] - 4612s 3s/step - loss: 0.1686 -
```

```
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:
```

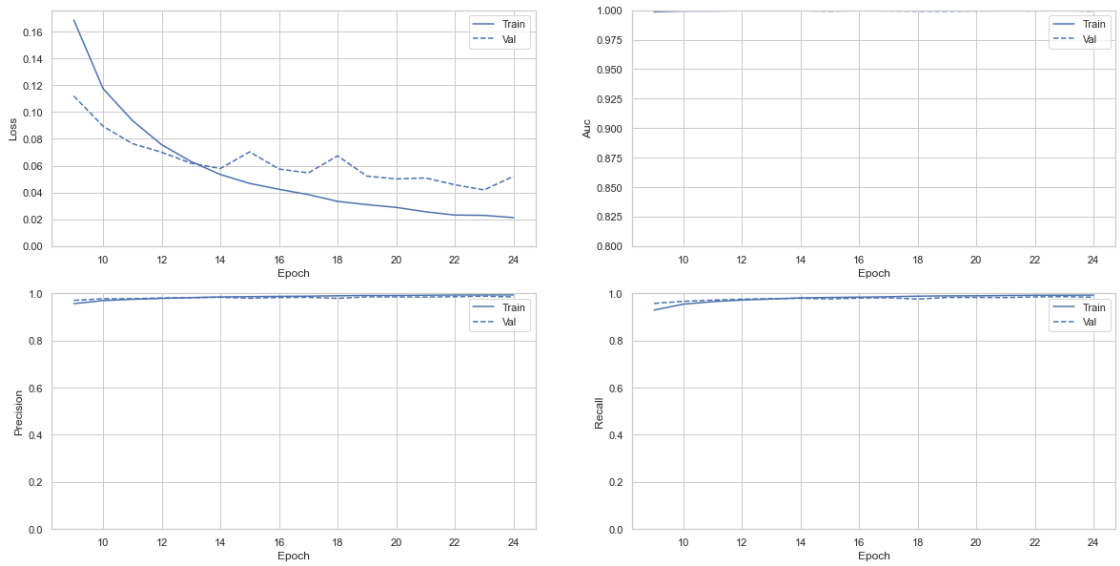
```
0.9852 - precision: 0.9868 - recall: 0.9836 - auc: 0.9998 - val_loss: 0.0575
- val_tp: 8508.0000 - val_fp: 154.0000 - val_tn: 321191.0000 - val_fn:
177.0000 - val_accuracy: 0.9808 - val_precision: 0.9822 - val_recall: 0.9796
- val_auc: 0.9997
Epoch 18/25
1358/1358 [==============================] - 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
```

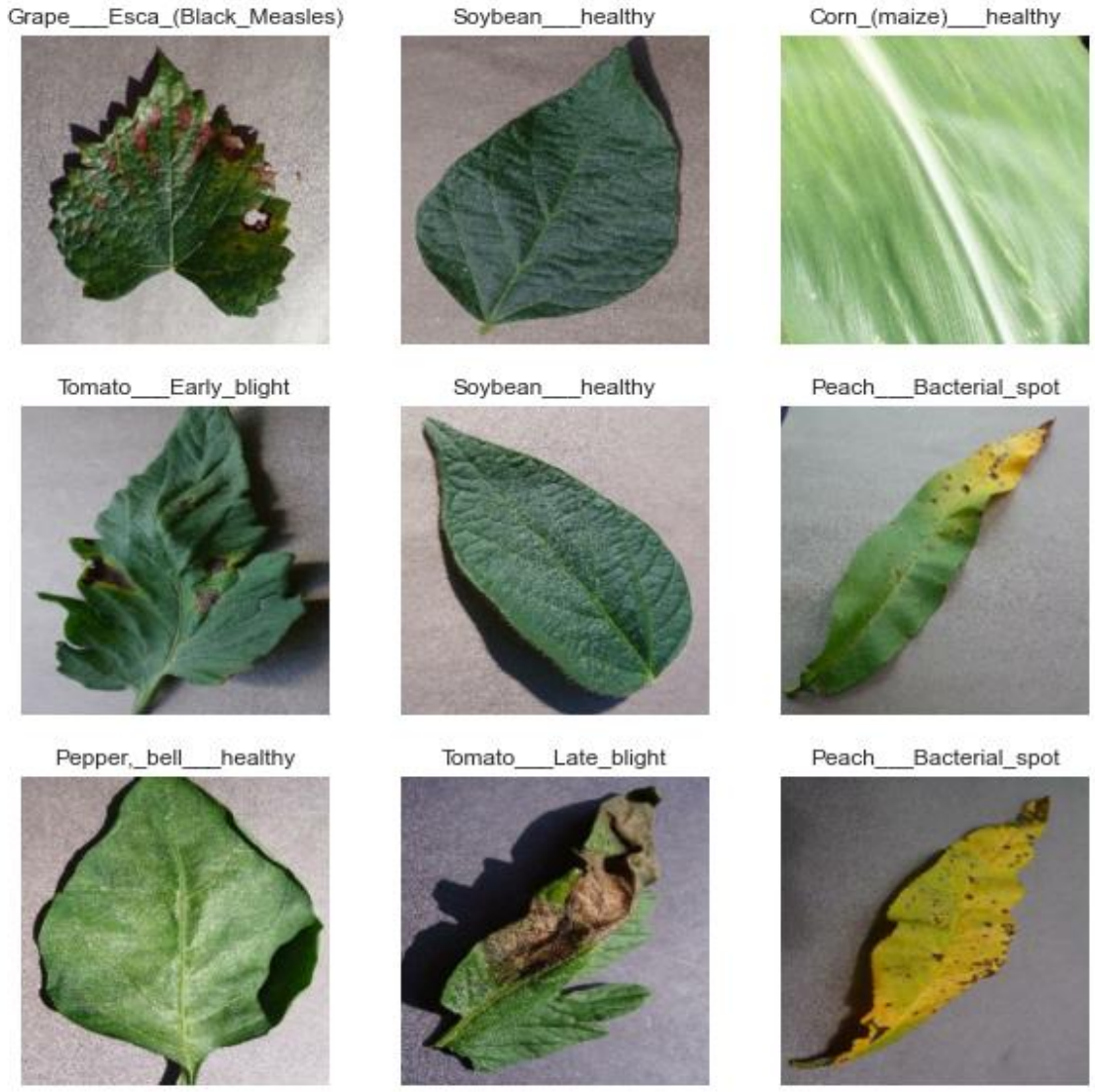
```
metrics = ["loss", "tp", "fp", "tn", "fn", "accuracy", "precision", "recall",
"auc"]
for i in range(len(result)):
    print("{} : {}".format(metrics[i],round(result[i], 3)))
```

```
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")
```