```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Drive already mounted at /content/drive; to attempt to forcibly remount, call
```

```
!unzip /content/drive/MyDrive/Tomato
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing import image
import numpy as np
import os
import random
from shutil import copyfile
import matplotlib.image  as mpimg
import matplotlib.pyplot as plt
from glob import glob
```

```
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.layers import Conv2D, Activation, MaxPooling2D, Dropout, Dens
from tensorflow.keras.layers import GlobalAveragePooling2D, Flatten, BatchNormaliza
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, EarlySto
from keras.utils.np_utils import to_categorical
```

```
to_create = [
'leaf_disease',
'leaf_disease/training',
'leaf_disease/testing',
'leaf_disease/training/Tomato_Bacterial_spot',
'leaf_disease/training/Tomato_Late_blight',
'leaf_disease/training/Tomato_Early_blight',
'leaf_disease/training/Tomato_healthy',
'leaf_disease/training/Tomato_Yellow_Leaf_Curl_Virus',
'leaf_disease/training/Tomato_Leaf_Mold',
'leaf_disease/testing/Tomato_Bacterial_spot',
'leaf_disease/testing/Tomato_Late_blight',
'leaf_disease/testing/Tomato_Early_blight',
'leaf_disease/testing/Tomato_healthy',
'leaf_disease/testing/Tomato_Yellow_Leaf_Curl_Virus',
'leaf_disease/testing/Tomato_Leaf_Mold'
]
for directory in to_create:
  try:
    os.mkdir(directory)
    print(directory, 'created')
  except:
    print(directory, 'failed')
```

```
leaf_disease created
leaf_disease/training created
leaf_disease/testing created
leaf_disease/training/Tomato_Bacterial_spot created
leaf_disease/training/Tomato_Late_blight created
leaf_disease/training/Tomato_Early_blight created
leaf_disease/training/Tomato_healthy created
leaf_disease/training/Tomato_Yellow_Leaf_Curl_Virus created
leaf_disease/training/Tomato_Leaf_Mold created
leaf_disease/testing/Tomato_Bacterial_spot created
leaf_disease/testing/Tomato_Late_blight created
leaf_disease/testing/Tomato_Early_blight created
leaf_disease/testing/Tomato_healthy created
leaf_disease/testing/Tomato_Yellow_Leaf_Curl_Virus created
leaf_disease/testing/Tomato_Leaf_Mold created
```

```python
def split_data(SOURCE, TRAINING, TESTING, SPLIT_SIZE):
    all_files = []

    for file_name in os.listdir(SOURCE):
        file_path = SOURCE + file_name

        if os.path.getsize(file_path):
            all_files.append(file_name)
        else:
            print('{} is zero length, so ignoring'.format(file_name))

    n_files = len(all_files)
    split_point = int(n_files * SPLIT_SIZE)

    shuffled = random.sample(all_files, n_files)

    train_set = shuffled[:split_point]
    test_set = shuffled[split_point:]

    for file_name in train_set:
        copyfile(SOURCE + file_name, TRAINING + file_name)

    for file_name in test_set:
        copyfile(SOURCE + file_name, TESTING + file_name)


SOURCE_DIR = "Tomato/Tomato_Late_blight/"
TRAINING_DIR = "leaf_disease/training/Tomato_Late_blight/"
TESTING_DIR ="leaf_disease/testing/Tomato_Late_blight/"
split_size = .5
split_data(SOURCE_DIR, TRAINING_DIR, TESTING_DIR, split_size)


SOURCE_DIR = "Tomato/Tomato_Bacterial_spot/"
TRAINING_DIR = "leaf_disease/training/Tomato_Bacterial_spot/"
TESTING_DIR ="leaf_disease/testing/Tomato_Bacterial_spot/"
split_size = .5
split_data(SOURCE_DIR, TRAINING_DIR, TESTING_DIR, split_size)


SOURCE_DIR = "Tomato/Tomato_Early_blight/"
```

```
TRAINING_DIR = "leaf_disease/training/Tomato_Early_blight/"
TESTING_DIR ="leaf_disease/testing/Tomato_Early_blight/"
split_size = .5
split_data(SOURCE_DIR, TRAINING_DIR, TESTING_DIR, split_size)
```

```
SOURCE_DIR = "Tomato/Tomato_healthy/"
TRAINING_DIR = "leaf_disease/training/Tomato_healthy/"
TESTING_DIR ="leaf_disease/testing/Tomato_healthy/"
split_size = .5
split_data(SOURCE_DIR, TRAINING_DIR, TESTING_DIR, split_size)
```

```
SOURCE_DIR = "Tomato/Tomato_Yellow_Leaf_Curl_Virus/"
TRAINING_DIR = "leaf_disease/training/Tomato_Yellow_Leaf_Curl_Virus/"
TESTING_DIR ="leaf_disease/testing/Tomato_Yellow_Leaf_Curl_Virus/"
split_size = .5
split_data(SOURCE_DIR, TRAINING_DIR, TESTING_DIR, split_size)
```

```
SOURCE_DIR = "Tomato/Tomato_Leaf_Mold/"
TRAINING_DIR = "leaf_disease/training/Tomato_Leaf_Mold/"
TESTING_DIR ="leaf_disease/testing/Tomato_Leaf_Mold/"
split_size = .5
split_data(SOURCE_DIR, TRAINING_DIR, TESTING_DIR, split_size)
```

```
#!unzip /content/drive/MyDrive/leaf_disease
```

```
data_dir = "leaf_disease/training"
classes = os.listdir(data_dir)
classes
```

```
['Tomato_Bacterial_spot',
 'Tomato_Early_blight',
 'Tomato_Yellow_Leaf_Curl_Virus',
 'Tomato_Leaf_Mold',
 'Tomato_healthy',
 'Tomato_Late_blight']
```

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   fill_mode='nearest')

valid_datagen = ImageDataGenerator(rescale=1./255)

batch_size = 128
training_dir = 'leaf_disease/training'
validation_dir = 'leaf_disease/testing'

training_generator = train_datagen.flow_from_directory(training_dir,
                                                 target_size=(256, 256),
                                                 batch_size=batch_size,
                                                 batch_size=batch_size,
                                                 class_mode='categorical')

valid_generator = valid_datagen.flow_from_directory(validation_dir,
                                            target_size=(256, 256),
                                            batch_size=batch_size,
                                            class_mode='categorical',
                                            shuffle=False)
```

```
Found 4491 images belonging to 6 classes.
Found 4495 images belonging to 6 classes.
```

```
class_dict = training_generator.class_indices
print(class_dict)
```

```
{'Tomato_Bacterial_spot': 0, 'Tomato_Early_blight': 1, 'Tomato_Late_blight': 2
```

```
target_names = list(class_dict.keys())
print(target_names)
```

```
['Tomato_Bacterial_spot', 'Tomato_Early_blight', 'Tomato_Late_blight', 'Tomatc
```

```
train_num = training_generator.samples
valid_num = valid_generator.samples
```

```
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
```

```
pre_trained_model = VGG16(input_shape=(256, 256, 3), include_top=False, weights="im
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applicat
58892288/58889256 [==============================] - 0s 0us/step
```

```
for layer in pre_trained_model.layers:
    print(layer.name)
    layer.trainable = False

print(len(pre_trained_model.layers))
```

```
input_1
block1_conv1
block1_conv2
block1_pool
block2_conv1
block2_conv2
block2_pool
block3_conv1
block3_conv2
block3_conv3
block3_pool
block4_conv1
block4_conv2
block4_conv3
block4_pool
block5_conv1
```

```
     block5_conv2
     block5_conv3
     block5_pool
     19
```

```
last_layer = pre_trained_model.get_layer('block5_pool')
print('last layer output shape:', last_layer.output_shape)
last_output = last_layer.output
```

```
     last layer output shape: (None, 8, 8, 512)
```

```
from tensorflow.keras import layers
from tensorflow.keras import Model
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping
```

```
# Flatten the output layer to 1 dimension
x=layers.Flatten()(last_output)
#x = layers.GlobalMaxPooling2D()(last_output)
# Add a fully connected layer with 512 hidden units and ReLU activation
x = layers.Dense(512, activation='relu')(x)
# Add a dropout rate of 0.5
x = layers.Dropout(0.2)(x)
# Add a final sigmoid layer for classification
x = layers.Dense(6, activation='softmax')(x)
# Configure and compile the model

model = Model(pre_trained_model.input, x)
optimizer = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=0.0, amsgr
model.compile(loss='categorical_crossentropy',
              optimizer=optimizer,
              metrics=['accuracy'])
```

```
history = model.fit(training_generator,
                    epochs = 47, validation_data = valid_generator,
                    verbose = 1, steps_per_epoch=(train_num // batch_size),
                    validation_steps=(valid_num// batch_size))
```

```
     35/35 [==============================] - 121s 3s/step - loss: 0.3627 - accurac
     Epoch 7/47
     35/35 [==============================] - 121s 3s/step - loss: 0.3502 - accurac

     Epoch 8/47
     35/35 [==============================] - 121s 3s/step - loss: 0.3090 - accurac
     Epoch 9/47
     35/35 [==============================] - 121s 3s/step - loss: 0.2814 - accurac
     Epoch 10/47
     35/35 [==============================] - 121s 3s/step - loss: 0.2763 - accurac
     Epoch 11/47
     35/35 [==============================] - 121s 3s/step - loss: 0.2582 - accurac
     Epoch 12/47
     35/35 [==============================] - 121s 3s/step - loss: 0.2301 - accurac
     Epoch 13/47
```

```
    35/35 [==============================] - 122s 3s/step - loss: 0.2215 - accurac
    Epoch 14/47
    35/35 [==============================] - 123s 4s/step - loss: 0.1967 - accurac
    Epoch 15/47
    35/35 [==============================] - 122s 3s/step - loss: 0.2050 - accurac
    Epoch 16/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1869 - accurac
    Epoch 17/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1809 - accurac
    Epoch 18/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1924 - accurac
    Epoch 19/47
    35/35 [==============================] - 121s 3s/step - loss: 0.2226 - accurac
    Epoch 20/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1991 - accurac
    Epoch 21/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1818 - accurac
    Epoch 22/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1726 - accurac
    Epoch 23/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1720 - accurac
    Epoch 24/47
    35/35 [==============================] - 121s 4s/step - loss: 0.1549 - accurac
    Epoch 25/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1503 - accurac
    Epoch 26/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1421 - accurac
    Epoch 27/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1580 - accurac
    Epoch 28/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1734 - accurac
    Epoch 29/47
    35/35 [==============================] - 121s 3s/step - loss: 0.1356 - accurac
    Epoch 30/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1564 - accurac
    Epoch 31/47
    35/35 [==============================] - 121s 4s/step - loss: 0.1393 - accurac
    Epoch 32/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1457 - accurac
    Epoch 33/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1083 - accurac
    Epoch 34/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1085 - accurac
    Epoch 35/47
    35/35 [==============================] - 122s 3s/step - loss: 0.1132 - accurac
```

```
import pickle
filename = 'VGG16_model.pkl'
pickle.dump(history.history, open(filename, 'wb'))
```

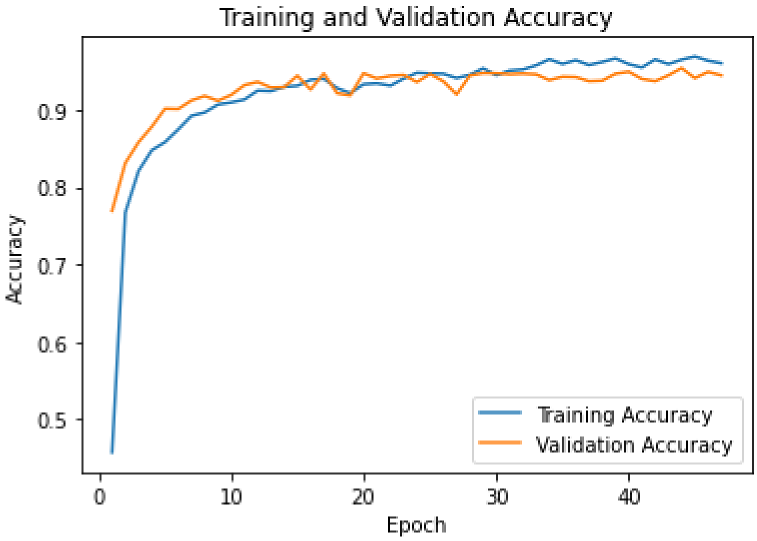
```
tf.keras.models.save_model(model,'transferlearning_VGG16.hdf5')
```

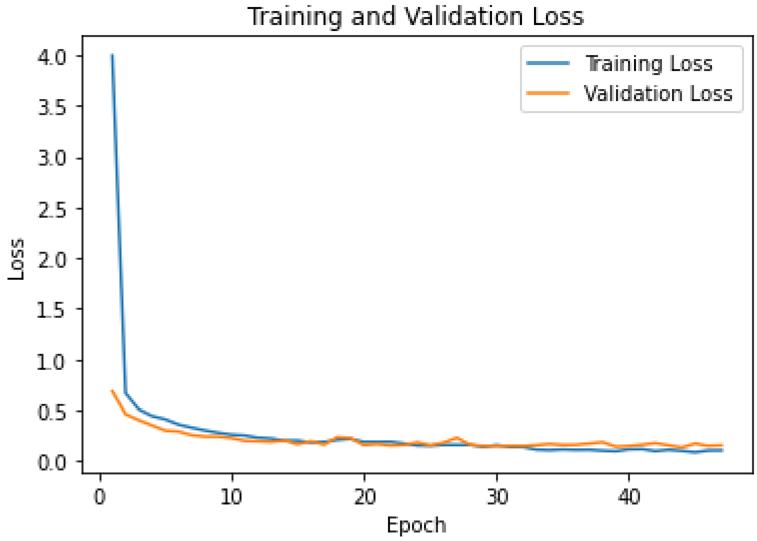
```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss) + 1)

#accuracy plot
```

```
#accuracy plot
plt.plot(epochs, acc, label='Training Accuracy')
plt.plot(epochs, val_acc, label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.savefig(fname='Training_Validation_Accuracy_VGG16')
plt.legend()

plt.figure()
#loss plot
plt.plot(epochs, loss, label='Training Loss')
plt.plot(epochs, val_loss, label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.savefig(fname='Training_Validation_Loss_VGG16')
plt.show()
```





```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
from sklearn import preprocessing


Y_pred = model.predict(valid_generator, valid_num // batch_size+1)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix')
print(confusion_matrix(valid_generator.classes, y_pred))
print('Classification Report')
target_names = ['Tomato_Bacterial_spot', 'Tomato_Early_blight',
                'Tomato_Late_blight', 'Tomato_Leaf_Mold',
                'Tomato_Yellow_Leaf_Curl_Virus', 'Tomato_healthy']
print(classification_report(valid_generator.classes, y_pred,
                            target_names=target_names))
```

```
Confusion Matrix
[[1014   18   18    0    7    7]
 [   6  427   42   12    1   12]
 [   1   41  903    6    0    4]
 [   1    9   16  440    0   10]
 [   4   16    4    6  672    2]
 [   0    1    0    1    0  794]]
Classification Report
                               precision    recall  f1-score   support

        Tomato_Bacterial_spot       0.99      0.95      0.97      1064
          Tomato_Early_blight       0.83      0.85      0.84       500
           Tomato_Late_blight       0.92      0.95      0.93       955
             Tomato_Leaf_Mold       0.95      0.92      0.94       476
Tomato_Yellow_Leaf_Curl_Virus       0.99      0.95      0.97       704
               Tomato_healthy       0.96      1.00      0.98       796

                     accuracy                           0.95      4495
                    macro avg       0.94      0.94      0.94      4495
                 weighted avg       0.95      0.95      0.95      4495
```

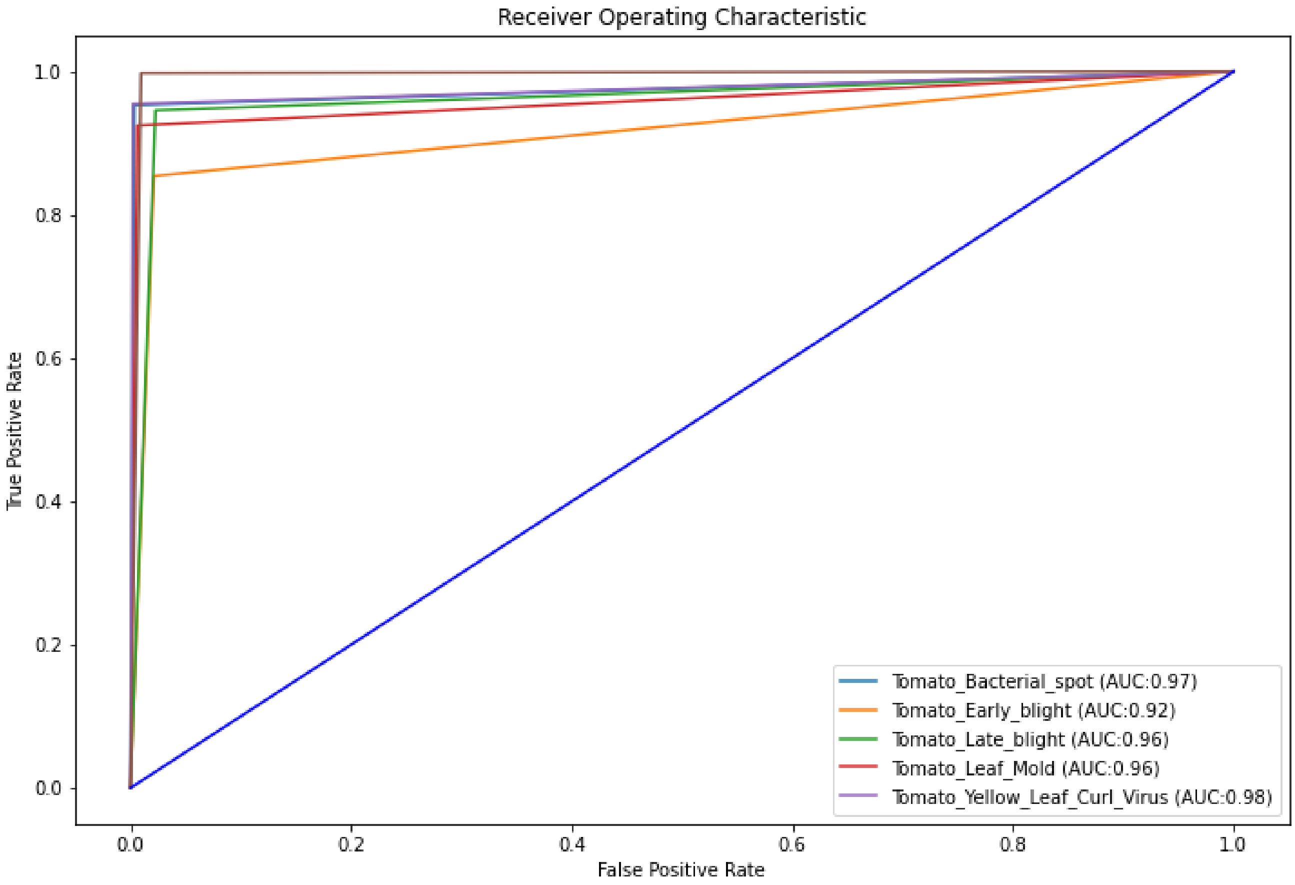
```
all_labels = ['Tomato_Bacterial_spot', 'Tomato_Early_blight', 'Tomato_Late_blight',
       'Tomato_Leaf_Mold', 'Tomato_Yellow_Leaf_Curl_Virus', 'Tomato_healthy']


fig, c_ax = plt.subplots(1,1, figsize = (12, 8))
def multiclass_roc_auc_score(y_test, y_pred, average="macro"):
    lb = preprocessing.LabelBinarizer()
    #lb = LabelBinarizer()
    lb.fit(y_test)
    y_test = lb.transform(y_test)
    y_pred = lb.transform(y_pred)

    for (idx, c_label) in enumerate(all_labels): # all_labels: no of the labels
        fpr, tpr, thresholds = roc_curve(y_test[:,idx].astype(int), y_pred[:,idx])
        plt.xlabel('False Positive Rate')
        plt.ylabel('True Positive Rate')
        plt.legend(loc = 'lower right')
        plt.title('Receiver Operating Characteristic')
        plt.plot(fpr, tpr, label = '%s (AUC:%0.2f)'  % (c_label, auc(fpr, tpr)))
    plt.plot(fpr, fpr, 'b-', label = 'Random Guessing')
    plt.savefig(fname='roc_auc_VGG16')
    return roc_auc_score(y_test, y_pred, average=average)

# calling
valid_generator.reset() # resetting generator
y_pred = model.predict(valid_generator, verbose = True)
y_pred = np.argmax(y_pred, axis=1)
multiclass_roc_auc_score(valid_generator.classes, y_pred)
```

```
    36/36 [==============================] - 39s 1s/step
    No handles with labels found to put in legend.
    0.9636495147919181
```


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
Tomato_Bacterial_spot (AUC:0.97)
Tomato_Early_blight (AUC:0.92)
Tomato_Late_blight (AUC:0.96)
Tomato_Leaf_Mold (AUC:0.96)
Tomato_Yellow_Leaf_Curl_Virus (AUC:0.98)