```
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.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.resnet_v2 import ResNet152V2
from tensorflow.keras.applications.efficientnet import EfficientNetB7
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2
```

```
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:
```

```
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
```

```
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 = .8
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 = .8
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 = .8
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 = .8
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 = .8
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 = .8
split_data(SOURCE_DIR, TRAINING_DIR, TESTING_DIR, split_size)


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

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

```
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,
                                                       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 8283 images belonging to 6 classes.
Found 4693 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 keras.applications import ResNet50
# from keras.optimizers import Adam,SGD
# from keras.models import Sequential
# model = Sequential()
# # Sequential needed for add method
# model.add(ResNet50(include_top = False, pooling = 'avg' ,weights = 'imagenet'))
# model.add(Dense(6, activation='softmax'))
# model.layers[0].trainable = False
```

```
# model.compile(tf.keras.optimizers.Adam(0.001), loss='categorical_crossentropy', m
```

```
def add_new_last_layer(base_model, nb_classes):
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(512, activation='relu')(x)
    predictions = Dense(nb_classes, activation='softmax')(x)
    model = Model(base_model.input, predictions)
    return model
```

```
base_model = ResNet50(weights='imagenet', include_top=False) #include_top=False exc
model = add_new_last_layer(base_model, 6)
```

```
model.compile(tf.keras.optimizers.SGD(0.001), loss='categorical_crossentropy', metr


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

```
Epoch 1/47
64/64 [==============================] - 173s 3s/step - loss: 1.7272 - accurac
Epoch 2/47
64/64 [==============================] - 167s 3s/step - loss: 1.0637 - accurac
Epoch 3/47
64/64 [==============================] - 167s 3s/step - loss: 0.7593 - accurac
Epoch 4/47
64/64 [==============================] - 167s 3s/step - loss: 0.5779 - accurac
Epoch 5/47
64/64 [==============================] - 167s 3s/step - loss: 0.4467 - accurac
Epoch 6/47
64/64 [==============================] - 168s 3s/step - loss: 0.3702 - accurac
Epoch 7/47
64/64 [==============================] - 168s 3s/step - loss: 0.3152 - accurac
Epoch 8/47
64/64 [==============================] - 168s 3s/step - loss: 0.2660 - accurac
Epoch 9/47
64/64 [==============================] - 168s 3s/step - loss: 0.2328 - accurac
Epoch 10/47
64/64 [==============================] - 168s 3s/step - loss: 0.2077 - accurac
Epoch 11/47
64/64 [==============================] - 167s 3s/step - loss: 0.1879 - accurac
Epoch 12/47
64/64 [==============================] - 167s 3s/step - loss: 0.1635 - accurac
Epoch 13/47
64/64 [==============================] - 168s 3s/step - loss: 0.1536 - accurac
Epoch 14/47
64/64 [==============================] - 167s 3s/step - loss: 0.1358 - accurac
Epoch 15/47
64/64 [==============================] - 168s 3s/step - loss: 0.1278 - accurac
Epoch 16/47
64/64 [==============================] - 167s 3s/step - loss: 0.1142 - accurac
Epoch 17/47
64/64 [==============================] - 168s 3s/step - loss: 0.1096 - accurac
Epoch 18/47
64/64 [==============================] - 167s 3s/step - loss: 0.1021 - accurac
Epoch 19/47
64/64 [==============================] - 167s 3s/step - loss: 0.0957 - accurac
Epoch 20/47
64/64 [==============================] - 167s 3s/step - loss: 0.0884 - accurac
Epoch 21/47
64/64 [==============================] - 167s 3s/step - loss: 0.0970 - accurac
Epoch 22/47
64/64 [==============================] - 168s 3s/step - loss: 0.0801 - accurac
Epoch 23/47
64/64 [==============================] - 167s 3s/step - loss: 0.0747 - accurac
Epoch 24/47
64/64 [==============================] - 167s 3s/step - loss: 0.0721 - accurac
```

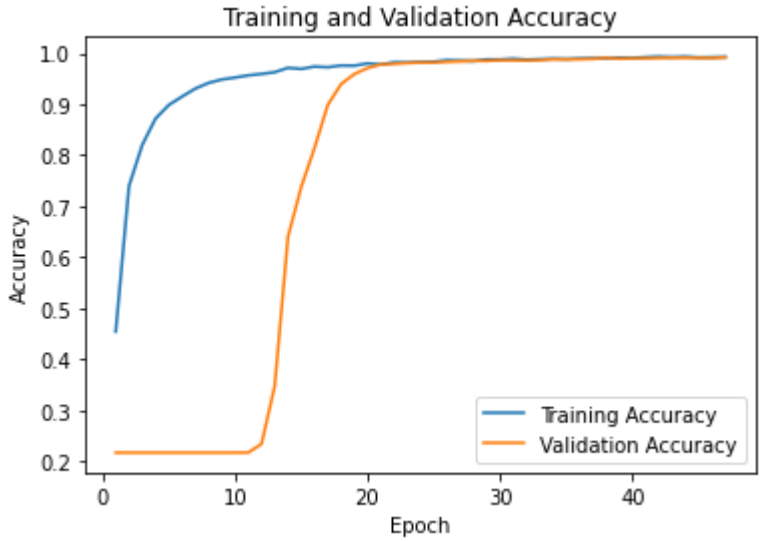
```
Epoch 25/47
64/64 [==============================] - 167s 3s/step - loss: 0.0685 - accurac
Epoch 26/47
64/64 [==============================] - 167s 3s/step - loss: 0.0639 - accurac
Epoch 27/47
64/64 [==============================] - 167s 3s/step - loss: 0.0646 - accurac
Epoch 28/47
64/64 [==============================] - 168s 3s/step - loss: 0.0625 - accurac
Epoch 29/47
64/64 [==============================] - 168s 3s/step - loss: 0.0570 - accurac
Epoch 30/47
```

```
import pickle
filename = 'ResNet50_model.pkl'
pickle.dump(history.history, open(filename, 'wb'))
```

```
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
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_ResNet50')
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_Resnet50')
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
[[1114    2    1    0    0    0]
 [   3  492    9    0    0    7]
 [   0    4  995    0    0    2]
 [   2    2    3  491    0    1]
 [   3    0    0    0  735    0]
 [   0    0    0    0    0  827]]
Classification Report
                                precision    recall  f1-score   support

          Tomato_Bacterial_spot       0.99      1.00      1.00      1117
            Tomato_Early_blight       0.98      0.96      0.97       511
             Tomato_Late_blight       0.99      0.99      0.99      1001
               Tomato_Leaf_Mold       1.00      0.98      0.99       499
Tomato_Yellow_Leaf_Curl_Virus       1.00      1.00      1.00       738
                Tomato_healthy       0.99      1.00      0.99       827

                      accuracy                           0.99      4693
                     macro avg       0.99      0.99      0.99      4693
                  weighted avg       0.99      0.99      0.99      4693
```

```
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_InceptionV3')
    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)
```

```
37/37 [==============================] - 20s 539ms/step
No handles with labels found to put in legend.
0.9936486042745774
```