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

```
     Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
#IMPORT REQUIRED LIBRARIES:

import numpy as np
import pandas as pd
import os
from re import search
import shutil
from PIL import Image
import matplotlib.pyplot as plt
from tqdm import tqdm
import cv2
import seaborn as sns


from sklearn.preprocessing import MultiLabelBinarizer

import tensorflow as tf
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.applications import ResNet50, ResNet50V2
from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Input, Dense, ZeroPadding2D, Dropout, Activation, Flatten, Conv2D, MaxPooling2D, ReLU, BatchNormalizatior
```

```
#IMAGE PATH & DATAFRAME:

train_dir= '../content/drive/MyDrive/train'
test_dir =  '../content/drive/MyDrive/test'
train = pd.read_csv('../content/sample_data/train (1).csv')
```

```
train.head
```

```
<bound method NDFrame.head of                       image                           labels
0      800113bb65efe69e.jpg                          healthy
1      8002cb321f8bfcdf.jpg  scab frog_eye_leaf_spot complex
2      80070f7fb5e2ccaa.jpg                             scab
3      80077517781fb94f.jpg                             scab
4      800cbf0ff87721f8.jpg                          complex
...                     ...                              ...
18627  fffb900a92289a33.jpg                          healthy
18628  fffc488fa4c0e80c.jpg                             scab
18629  fffc94e092a59086.jpg                             rust
18630  fffe105cf6808292.jpg          scab frog_eye_leaf_spot
18631  fffe472a0001bd25.jpg                          healthy

[18632 rows x 2 columns]>
```

```
train = pd.DataFrame(train,columns = ['image','labels'])
train['labels'].value_counts()
```

```
scab                               4826
healthy                            4624
frog_eye_leaf_spot                 3181
rust                               1860
complex                            1602
powdery_mildew                     1184
scab frog_eye_leaf_spot             686
scab frog_eye_leaf_spot complex     200
frog_eye_leaf_spot complex          165
rust frog_eye_leaf_spot             120
rust complex                         97
powdery_mildew complex               87
Name: labels, dtype: int64
```

```
train['labels'] = train['labels'].apply(lambda s: s.split(' '))
train[:10]
```

	image	labels
0	800113bb65efe69e.jpg	[healthy]
1	8002cb321f8bfcdf.jpg	[scab, frog_eye_leaf_spot, complex]
2	80070f7fb5e2ccaa.jpg	[scab]
3	80077517781fb94f.jpg	[scab]
4	800cbf0ff87721f8.jpg	[complex]
5	800edef467d27c15.jpg	[healthy]
6	800f85dc5f407aef.jpg	[rust]
7	801d6dcd96e48ebc.jpg	[healthy]
8	801f78399a44e7af.jpg	[complex]
9	8021b94d437eb7d3.jpg	[healthy]

```
# Use the Image Data Generator to import the images from the dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rescale = 1/255.,
    rotation_range = 10,#Performing Rotation
    width_shift_range = 0.2,
    height_shift_range = 0.2,
    brightness_range = [0.2,1.0],
    shear_range = 0.2,
    zoom_range = 0.2,
    horizontal_flip=True,
    vertical_flip=True,
    validation_split= 0.2)
```

```
HEIGHT = 224
WIDTH=224
SEED = 444
BATCH_SIZE=32
train_ds = datagen.flow_from_dataframe(
    train,
    directory = '../input/resized-plant2021/img_sz_512',# We are using the resized images otherwise it will take a lot of time to train
    x_col = 'image',
    y_col = 'labels',
    subset="training",
    color_mode="rgb",
    target_size = (HEIGHT,WIDTH),
    class_mode="categorical",
    batch_size=BATCH_SIZE,
    shuffle=True,
    seed=SEED,
)


val_ds = datagen.flow_from_dataframe(
    train,
    directory = '../input/resized-plant2021/img_sz_512',# We are using the resized images otherwise it will take a lot of time to train
    x_col = 'image',
    y_col = 'labels',
    subset="validation",
    color_mode="rgb",
    target_size = (HEIGHT,WIDTH),
    class_mode="categorical",
    batch_size=BATCH_SIZE,
    shuffle=True,
    seed=SEED,
)
```

```
Found 0 validated image filenames belonging to 0 classes.
Found 0 validated image filenames belonging to 0 classes.
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1137: UserWarning: Found 18632 invalid image filename(s) in x_
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/keras/src/preprocessing/image.py:1137: UserWarning: Found 18632 invalid image filename(s) in x_
  warnings.warn(
```

```
example = next(train_ds)
print(example[0].shape)
#plt.imshow(example[0][0,:,:,:])
plt.show()
```

```
(0, 224, 224, 3)
```

```
#Convolution Block
def Convolution_Block(Input, filters, k, s, stage, block):

    filter1, filter2, filter3 = filters
    base_name = str(stage) + block + "_branch"
    conv_name_base = "res" + base_name
    bn_name_base = "bn" + base_name

    #Block 1
    x = Conv2D(filters=filter1, kernel_size=(1, 1),
               strides=(s, s),
               name=conv_name_base + "2a",
               kernel_initializer='he_normal')(Input)

    x = BatchNormalization(axis=1, name=bn_name_base + "2a")(x)
    x = Activation('relu')(x)

    #Block 2
    x = Conv2D(filters=filter2, kernel_size=(k, k),
               padding='same',
               name=conv_name_base + "2b",
              kernel_initializer='he_normal')(x)

    x = BatchNormalization(axis=1, name=bn_name_base + "2b")(x)
    x = Activation('relu')(x)

    #Block 3
    x = Conv2D(filters=filter3, kernel_size=(1, 1),
              name=conv_name_base + "2c",
              kernel_initializer='he_normal')(x)

    x = BatchNormalization(axis=1,name=bn_name_base + "2c")(x)

    #Residual Connection
    skip_connection = Conv2D(filters=filter3, kernel_size=(1, 1),
                             strides=(s, s),
                             name=conv_name_base + "1",
                             kernel_initializer='he_normal')(Input)

    skip_connection = BatchNormalization(axis=1, name=bn_name_base + "1")(skip_connection)

    x = add([x, skip_connection])
    x = Activation('relu')(x)

    return x


#Identity Block
def Identity_Block(Input, filters, k, stage, block):

    filter1, filter2, filter3 = filters
    base_name = str(stage) + block + "_branch"
    conv_name_base = "res" + base_name
    bn_name_base = "bn" + base_name

    #Block 1
    x = Conv2D(filters=filter1, kernel_size=(1, 1),
                  name=conv_name_base + "2a",
                  kernel_initializer='he_normal')(Input)

        x = BatchNormalization(axis=1, name=bn_name_base + "2a")(x)
        x = Activation('relu')(x)

        #Block 2
        x = Conv2D(filters=filter2, kernel_size=(k, k),
                  padding='same',
                  name=conv_name_base + "2b",
                  kernel_initializer='he_normal')(x)

        x = BatchNormalization(axis=1, name=bn_name_base + "2b")(x)
        x = Activation('relu')(x)

        #Block 3
        x = Conv2D(filters=filter3, kernel_size=(1, 1),
                  name=conv_name_base + "2c",
                  kernel_initializer='he_normal')(x)

        x = BatchNormalization(axis=1, name=bn_name_base + "2c")(x)

        #Residual Connection
        x = add([x, Input])
        x = Activation('relu')(x)

        return x


    input = Input(shape=(HEIGHT,WIDTH,3));

    # #Initial Block
    # x = ZeroPadding2D(padding=(3, 3), name='conv1_pad')(input)
    # x = Conv2D(filters=64, kernel_size=(7, 7),
    #           strides=(2, 2), padding='valid',
    #           name='conv1',
    #           kernel_initializer='he_normal')(x)

    # x = BatchNormalization(axis=1, name='bn_conv1')(x)
    # x = Activation('relu')(x)
    # x = ZeroPadding2D(padding=(1, 1), name='pool1_pad')(x)
    # x = MaxPooling2D((3, 3), strides=(2, 2))(x)

    # #Block 1
    # x = Convolution_Block(x, [64, 64, 256], 3, s=1, stage=2, block="a")
    # x = Identity_Block(x, [64, 64, 256], 3, stage=2, block="b")
    # x = Identity_Block(x, [64, 64, 256], 3, stage=2, block="c")

    # #Block 2
    # x = Convolution_Block(x, [128, 128, 512], 3, s=2, stage=3, block="a")
    # x = Identity_Block(x, [128, 128, 512], 3, stage=3, block="b")
    # x = Identity_Block(x, [128, 128, 512], 3, stage=3, block="c")
    # x = Identity_Block(x, [128, 128, 512], 3, stage=3, block="d")

    # #Block 3
    # x = Convolution_Block(x, [256, 256, 1024], 3, s=2, stage=4, block="a")
    # x = Identity_Block(x, [256, 256, 1024], 3, stage=4, block="b")
    # x = Identity_Block(x, [256, 256, 1024], 3, stage=4, block="c")
    # x = Identity_Block(x, [256, 256, 1024], 3, stage=4, block="d")
    # x = Identity_Block(x, [256, 256, 1024], 3, stage=4, block="e")
    # x = Identity_Block(x, [256, 256, 1024], 3, stage=4, block="f")

    # #Block 4
    # x = Convolution_Block(x, [512, 512, 2048], 3, s=2, stage=5, block="a")
    # x = Identity_Block(x, [512, 512, 2048], 3, stage=5, block="b")
    # x = Identity_Block(x, [512, 512, 2048], 3, stage=5, block="c")

    # #Block 5
    # x = GlobalAveragePooling2D()(x)
    # # block5_flatten = Flatten()(block5_avg_pooling)

    # x = Dense(64, activation='relu')(x)
```

```
# x = Dense(64, activation='relu')(x)
# # block5_dropout1 = Dropout(0.2)(block5_dense1)

# x = Dense(16, activation='relu')(x)
# # block5_dropout2 = Dropout(0.2)(block5_dense2)

# output = Dense(6, name='model_output',
#                 activation='softmax')(x)

# model = Model(input, output)


pretrained_model = ResNet50(input_shape=(HEIGHT,WIDTH,3), include_top=False, weights='..')

for layer in pretrained_model.layers[:]:
    layer.trainable = False

for layer in pretrained_model.layers:
    print(layer, layer.trainable)

model = Sequential()
model.add(pretrained_model)
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(16, activation='relu'))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(6, activation='softmax'))

# Compile the Model
model.compile(optimizer=tf.keras.optimizers.RMSprop(learning_rate=2e-5),
    loss='binary_crossentropy',
    metrics=['accuracy'])
```

```
model.summary()
```

```
---------------------------------------------------------------------------
IsADirectoryError                         Traceback (most recent call last)
<ipython-input-74-18f5e491b2d1> in <cell line: 55>()
     53 
     54 
---> 55 pretrained_model = ResNet50(input_shape=(HEIGHT,WIDTH,3), include_top=False, weights='..')
     56 
     57 for layer in pretrained_model.layers[:]:
```

4 frames

```
/usr/local/lib/python3.10/dist-packages/h5py/_hl/files.py in make_fid(name, mode, userblock_size, fapl, fcpl, swmr)
    229         if swmr and swmr_support:
    230             flags |= h5f.ACC_SWMR_READ
--> 231         fid = h5f.open(name, flags, fapl=fapl)
    232     elif mode == 'r+':
    233         fid = h5f.open(name, h5f.ACC_RDWR, fapl=fapl)

h5py/_objects.pyx in h5py._objects.with_phil.wrapper()

h5py/_objects.pyx in h5py._objects.with_phil.wrapper()

h5py/h5f.pyx in h5py.h5f.open()

IsADirectoryError: [Errno 21] Unable to open file (file read failed: time = Fri Oct 13 10:56:41 2023
, filename = '..', file descriptor = 46, errno = 21, error message = 'Is a directory', buf = 0x7fffa96ffb18, total read size = 8,
bytes this sub-read = 8, bytes actually read = 18446744073709551615, offset = 0)
```

SEARCH STACK OVERFLOW

```
checkpoint=ModelCheckpoint(r'ResNet50.h5',
                           monitor='val_loss',
                           mode='min',
                           save_best_only=True,
                           verbose=1)
earlystop=EarlyStopping(monitor='val_loss',
                        min_delta=0,
                        patience=10,
                        verbose=1,
                        restore_best_weights=True)

callbacks=[checkpoint,earlystop]


resNet_model=model.fit(train_ds,
                         validation_data=val_ds,
                         epochs=15,
                         shuffle=True,
                         batch_size=BATCH_SIZE,
                         callbacks=callbacks)


model_history = resNet_model.history

plt.figure()
plt.plot(model_history['accuracy'])
plt.plot(model_history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.savefig('accuracy')
plt.show()


plt.figure()
plt.plot(model_history['loss'])
plt.plot(model_history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'])
plt.savefig('loss')
plt.show()
```

```
submission = pd.read_csv('/kaggle/input/plant-pathology-2021-fgvc8/sample_submission.csv')
submission.head()
```

```
test_datagen = ImageDataGenerator(
    rescale = 1/255.0
)

test_generator = test_datagen.flow_from_dataframe(
    submission,
    directory="../input/plant-pathology-2021-fgvc8/test_images",
    x_col='image',
    y_col=None,
    class_mode=None,
    color_mode="rgb",
    target_size=(HEIGHT,WIDTH),
)
```

```
example = next(test_generator)
for i in range(len(example)):
    print(example[i].shape)
    plt.imshow(example[i][:,:,:])
    plt.show()
```

```
preds = model.predict(test_generator)
print(preds)
```

```
preds = preds.tolist()

indices = []
for pred in preds:
    temp = []
    for category in pred:
        if category>=0.23:
            temp.append(pred.index(category))
    if temp!=[]:
        indices.append(temp)
    else:
        temp.append(np.argmax(pred))
        indices.append(temp)

print(indices)


labels = (train_ds.class_indices)
labels = dict((v,k) for k,v in labels.items())
print(labels)

testlabels = []

for image in indices:
    temp = []
    for i in image:
        temp.append(str(labels[i]))
    testlabels.append(' '.join(temp))

print(testlabels)


submission['labels'] = testlabels
submission.head()
submission.to_csv('submission.csv', index=False)
```