<<<<< SOURCE CODE >>>>>

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import skimage.io

import os

import tqdm

import glob

import tensorflow

from tqdm import tqdm

from sklearn.utils import shuffle

from sklearn.model\_selection import train\_test\_split

from skimage.color import rgb2gray

from skimage.io import imread, imshow

from skimage.transform import resize

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import InputLayer, BatchNormalization, Dropout, Flatten, Dense, Activation, MaxPool2D, Conv2D

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

from tensorflow.keras.applications.densenet import DenseNet169

from tensorflow.keras.preprocessing.image import load\_img, img\_to\_array

train\_datagen = ImageDataGenerator(rescale = 1./255,

rotation\_range=30,

zoom\_range=0.2,

horizontal\_flip=True,

vertical\_flip=True,

validation\_split = 0.2)

valid\_datagen = ImageDataGenerator(rescale = 1./255,

validation\_split = 0.2)

test\_datagen = ImageDataGenerator(rescale = 1./255)

train\_dataset = train\_datagen.flow\_from\_directory(directory = 'D:/brain alzheimer detection/Dataset',

target\_size = (224,224),

class\_mode = 'categorical',

subset = 'training',

batch\_size = 128)

valid\_dataset = valid\_datagen.flow\_from\_directory(directory = 'D:/brain alzheimer detection/Dataset',

target\_size = (224,224),

class\_mode = 'categorical',

subset = 'validation',

batch\_size = 128)

fig, ax = plt.subplots(nrows = 1, ncols = 5, figsize=(20,20))

for i in tqdm(range(0,5)):

rand1 = np.random.randint(len(train\_dataset))

rand2 = np.random.randint(100)

ax[i].imshow(train\_dataset[rand1][0][rand2])

ax[i].axis('off')

a = train\_dataset[rand1][1][rand2]

if a[0] == 1:

ax[i].set\_title('Mild Dementia')

elif a[1] == 1:

ax[i].set\_title('Moderate Dementia')

elif a[2] == 1:

ax[i].set\_title('Non Demetia')

elif a[3] == 1:

ax[i].set\_title('Very Mild Dementia')

# Model Initialization

base\_model = DenseNet169(input\_shape=(224,224,3),

include\_top=False,

weights="imagenet")

# Freezing Layers

for layer in base\_model.layers:

layer.trainable=False

# Building Model

model=Sequential()

model.add(base\_model)

model.add(Dropout(0.5))

model.add(Flatten())

model.add(BatchNormalization())

model.add(Dense(2048,kernel\_initializer='he\_uniform'))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(1024,kernel\_initializer='he\_uniform'))

model.add(BatchNormalization())

model.add(Activation('relu'))

model.add(Dropout(0.5))

model.add(Dense(4,activation='softmax'))

# Summary

model.summary()

# Model Compile

OPT = tensorflow.keras.optimizers.Adam(lr=0.001)

model.compile(loss='categorical\_crossentropy',

metrics=[tensorflow.keras.metrics.AUC(name = 'auc')],

optimizer=OPT)

# Defining Callbacks

filepath = './best\_weights.hdf5'

earlystopping = EarlyStopping(monitor = 'val\_auc',

mode = 'max' ,

patience = 15,

verbose = 1)

checkpoint = ModelCheckpoint(filepath,

monitor = 'val\_auc',

mode='max',

save\_best\_only=True,

verbose = 1)

callback\_list = [earlystopping, checkpoint]

model\_history=model.fit(train\_dataset,

validation\_data=valid\_dataset,

epochs =30 ,

callbacks = callback\_list,

verbose = 1)

# Summarize history for loss

plt.plot(model\_history.history['loss'])

plt.plot(model\_history.history['val\_loss'])

plt.title('Model Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend(['Train', 'Validation'], loc='upper left', bbox\_to\_anchor=(1,1))

plt.show()

# Summarize history for loss

plt.plot(model\_history.history['auc'])

plt.plot(model\_history.history['val\_auc'])

plt.title('Model AUC')

plt.ylabel('AUC')

plt.xlabel('Epoch')

plt.legend(['Train', 'Validation'], loc='upper left', bbox\_to\_anchor=(1,1))

plt.show()

# Test Data

test\_dataset = test\_datagen.flow\_from\_directory(directory = 'D:/brain alzheimer detection/Dataset',

target\_size = (224,224),

class\_mode = 'categorical',

batch\_size = 128)

# Evaluating Loss and AUC

model.evaluate(test\_dataset)

# Test Case 1: Non-Dementia

dic = test\_dataset.class\_indices

idc = {k:v for v, k in dic.items()}

img = load\_img('D:/brain alzheimer detection/Dataset/Moderate\_Demented/moderate\_9.jpg', target\_size = (224,224,3))

img = img\_to\_array(img)

img = img/255

imshow(img)

plt.axis('off')

img = np.expand\_dims(img,axis=0)

answer = np.argmax(model.predict(img),axis=1)

probability = round(np.max(model.predict(img)\*100),2)

print(probability, '% chances are there that the image is',idc[answer[0]])