**VGG19 CODE**

import numpy as np # linear algebra

import pandas as pd # data processing, CSV file I/O (e.g. pd.read\_csv)

import cv2

import random

from tqdm import tqdm

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Input data files are available in the read-only "../input/" directory

# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os

for dirname, \_, filenames in os.walk('/kaggle/input'):

    for filename in filenames:

        print(os.path.join(dirname, filename))

from google.colab import drive # Mounting drive

drive.mount('/content/drive')

Path = 'drive/My Drive/Dataset' # Path of dataset in drive

# File containing detailed information of the dataset

df = pd.read\_csv("drive/My Drive/full/full\_df.csv")

# Separating cataract and normal eye information from csv file containing detailed information of the dataset

def has\_cataract(text):

    if "cataract" in text:

        return 1

    else:

        return 0

# Finding Left and right cataract from the file

df["left\_cataract"] = df["Left-Diagnostic Keywords"].apply(lambda x: has\_cataract(x))

df["right\_cataract"] = df["Right-Diagnostic Keywords"].apply(lambda x: has\_cataract(x))

# Separating Left and right cataract from the file

left\_cataract = df.loc[(df.C ==1) & (df.left\_cataract == 1)]["Left-Fundus"].values

left\_cataract[:15]

right\_cataract = df.loc[(df.C ==1) & (df.right\_cataract == 1)]["Right-Fundus"].values

right\_cataract[:15]

# Display number of left cataract and right cataract

print("Number of images in left cataract: {}".format(len(left\_cataract)))

print("Number of images in right cataract: {}".format(len(right\_cataract)))

# Separating Left and right normal from the file

left\_normal = df.loc[(df.C ==0) & (df["Left-Diagnostic Keywords"] == "normal fundus")]["Left-Fundus"].sample(250,random\_state=42).values

right\_normal = df.loc[(df.C ==0) & (df["Right-Diagnostic Keywords"] == "normal fundus")]["Right-Fundus"].sample(250,random\_state=42).values

right\_normal[:15]

# Combine left and right images of cataract and also normal eye

cataract = np.concatenate((left\_cataract,right\_cataract),axis=0)

normal = np.concatenate((left\_normal,right\_normal),axis=0)

# Preprocessing images

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

dataset\_dir = "drive/My Drive/Dataset"

image\_size=224

labels = []

dataset = []

def create\_dataset(image\_category,label):

    for img in tqdm(image\_category):

        image\_path = os.path.join(dataset\_dir,img)

        try:

            image = cv2.imread(image\_path,cv2.IMREAD\_COLOR)

            image = cv2.resize(image,(image\_size,image\_size))

        except:

            continue

        dataset.append([np.array(image),np.array(label)])

    random.shuffle(dataset)

    return dataset

# Create dataset

dataset = create\_dataset(cataract,1)

dataset = create\_dataset(normal,0)

# Plot images

plt.figure(figsize=(12,7))

for i in range(10):

    sample = random.choice(range(len(dataset)))

    image = dataset[sample][0]

    category = dataset[sample][1]

    if category== 0:

        label = "Normal"

    else:

        label = "Cataract"

    plt.subplot(2,5,i+1)

    plt.imshow(image)

    plt.xlabel(label)

plt.tight\_layout()

# Create numpy array for images

x = np.array([i[0] for i in dataset]).reshape(1,image\_size,image\_size,3)

y = np.array([i[1] for i in dataset])

# Split dataset into train and test

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

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.2)

# Import Model

from tensorflow.keras.applications.vgg19 import VGG19

vgg = VGG19(weights="imagenet",include\_top = False,input\_shape=(image\_size,image\_size,3))

for layer in vgg.layers:

    layer.trainable = False

from tensorflow.keras import Sequential

from tensorflow.keras.layers import Flatten,Dense

model = Sequential()

model.add(vgg)

model.add(Flatten())

model.add(Dense(1,activation="sigmoid"))

# Display details of model

model.summary()

# Compile the model

model.compile(optimizer="adam",loss="binary\_crossentropy",metrics=["accuracy"])

#  returns a History callback, which has a history attribute containing the lists of successive losses and other metrics

from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping

checkpoint = ModelCheckpoint("vgg19.h5",monitor="val\_acc",verbose=1,save\_best\_only=True,

                             save\_weights\_only=False,period=1)

earlystop = EarlyStopping(monitor="val\_acc",patience=5,verbose=1)

history = model.fit(x\_train,y\_train,batch\_size=32,epochs=25,validation\_data=(x\_test,y\_test),

                    verbose=1,callbacks=[checkpoint,earlystop])

# Evaluating model and accuracy

loss,accuracy = model.evaluate(x\_test,y\_test)

print("loss:",loss)

print("Accuracy:",accuracy)

# Import libraries for confusion matrix, classification report and accuracy score

from sklearn.metrics import confusion\_matrix,classification\_report,accuracy\_score

y\_pred = model.predict\_classes(x\_test)

accuracy\_score(y\_test,y\_pred)

print(classification\_report(y\_test,y\_pred))

# Plot confusion matrix

from mlxtend.plotting import plot\_confusion\_matrix

cm = confusion\_matrix(y\_test,y\_pred)

plot\_confusion\_matrix(conf\_mat = cm,figsize=(8,7),

                      show\_normed = True);

# Plot Graph

plt.style.use("ggplot")

fig = plt.figure(figsize=(12,6))

epochs = range(1,26)

plt.subplot(1,2,1)

plt.plot(epochs,history.history["accuracy"],"go-")

plt.plot(epochs,history.history["val\_accuracy"],"ro-")

plt.title("Model Accuracy")

plt.xlabel("Epochs")

plt.ylabel("Accuracy")

plt.legend(["Train","val"],loc = "upper left")

plt.subplot(1,2,2)

plt.plot(epochs,history.history["loss"],"go-")

plt.plot(epochs,history.history["val\_loss"],"ro-")

plt.title("Model Loss")

plt.xlabel("Epochs")

plt.ylabel("Loss")

plt.legend(["Train","val"],loc = "upper left")

plt.show()