import sklearn

import keras

from keras.preprocessing.image import ImageDataGenerator

#from keras.optimizers import Adam

from keras.callbacks import ModelCheckpoint

#from keras.layers.normalization import BatchNormalization

from keras.regularizers import l2

from keras.utils.np\_utils import normalize

from keras.models import Sequential,Model

from keras.layers import Dense,Dropout,Flatten

from keras.layers import Conv2D,Activation,MaxPooling2D

from keras.layers import Concatenate

from keras import Input

from keras.callbacks import ModelCheckpoint

from tensorflow.keras.optimizers import Adam

from tensorflow.keras.layers import BatchNormalization

train\_data\_path='Downloads\\TRAIN MAMMO'

test\_data\_path='Downloads\\TEST MAMMO'

training\_datagen=ImageDataGenerator(rescale=1./255,rotation\_range=40,width\_shift\_range=0.2,height\_shift\_range=0.2,shear\_range=0.2,

zoom\_range=0.2,horizontal\_flip=True,fill\_mode='nearest')

training\_data=training\_datagen.flow\_from\_directory(train\_data\_path,target\_size=(150,150),batch\_size=32,class\_mode='binary')

training\_data.class\_indices

validation\_data\_path='Downloads\\VALIDATION MAMMO'

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

valid\_data = valid\_datagen.flow\_from\_directory(validation\_data\_path,

target\_size=(150, 150),

batch\_size=32,

class\_mode='binary')

import matplotlib.pyplot as plt

def plotImages(images\_arr):

fig, axes = plt.subplots(1, 5, figsize=(20, 20))

axes = axes.flatten()

for img, ax in zip(images\_arr, axes):

ax.imshow(img)

plt.tight\_layout()

plt.show()

images = [training\_data[0][0][0] for i in range(5)]

plotImages(images)

cnn\_model = keras.models.Sequential([

keras.layers.Conv2D(filters=32, kernel\_size=7, input\_shape=[150, 150, 3],kernel\_regularizer=l2(l=0.01)),

BatchNormalization(),

keras.layers.MaxPooling2D(pool\_size=(2,2)),

keras.layers.Conv2D(filters=64, kernel\_size=5),

BatchNormalization(),

keras.layers.MaxPooling2D(pool\_size=(2,2)),

keras.layers.Conv2D(filters=128, kernel\_size=3),

BatchNormalization(),

keras.layers.MaxPooling2D(pool\_size=(2,2)),

keras.layers.Conv2D(filters=256, kernel\_size=3),

BatchNormalization(),

keras.layers.MaxPooling2D(pool\_size=(2,2)),

keras.layers.Flatten(), # neural network beulding

keras.layers.Dense(units=128, activation='relu'), # input layers

BatchNormalization(),

keras.layers.Dropout(0.5),

keras.layers.Dense(units=256, activation='relu'),

BatchNormalization(),

keras.layers.Dropout(0.5),

keras.layers.Dense(units=2, activation='softmax') # output layer

])

cnn\_model.compile(optimizer = Adam(lr=0.0001), loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

mammo\_model\_path = 'Downloads//MAMMO MODEL PATH//ABNORMAL OR NORMAL.h5'

checkpoint = ModelCheckpoint(mammo\_model\_path, monitor='val\_accuracy', verbose=1, save\_best\_only=True, mode='max')

callbacks\_list = [checkpoint]

# train cnn model

history = cnn\_model.fit(training\_data,

epochs=20,

verbose=1,

validation\_data= valid\_data,

callbacks=callbacks\_list)

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('model accuracy')

plt.ylabel('accuracy')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()

# summarize history for loss

plt.plot(history.history['loss'])

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

plt.title('model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['train', 'test'], loc='upper left')

plt.show()

mammo\_model\_path1 = 'Downloads//MAMMO MODEL PATH//ABNORMAL OR NORMAL.h5'

mammo\_model1 = keras.models.load\_model(mammo\_model\_path1)

import numpy as np

from keras.preprocessing import image

def pred\_Normal\_Abnormal(model, Normal\_or\_Tuberculosis):

test\_image = image.load\_img(Normal\_or\_Tuberculosis, target\_size = (150, 150))

test\_image = image.img\_to\_array(test\_image)/255

test\_image = np.expand\_dims(test\_image, axis = 0)

result = model.predict(test\_image).round(3)

pred = np.argmax(result)

print(result, "--->>>", pred)

if pred == 0:

print('Predicted>>> Normal')

else:

print('Predicted>>> Abnormal')

import os

mammo\_datapath=f'Downloads/VALIDATION MAMMO//Normal//mdb020.jpg'

pred\_Normal\_Abnormal(mammo\_model1,mammo\_datapath)