# -\*- coding: utf-8 -\*-

"""

Created on Tue Nov 12 11:46:25 2019

Kyra Lee, Silver Harris, Victoria Puck-Karam

"""

#convolutional neural network

#importing Keras libraries and packages

from keras.models import Sequential

from keras.layers import Convolution2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

from keras.models import load\_model

#initialize the CNN

classifier = Sequential()

#step 1 - convlution

classifier.add(Convolution2D(32, 3, 3, input\_shape = (64, 64, 3), activation = 'relu'))

#step 2 - pooling

classifier.add(MaxPooling2D(pool\_size = (2, 2)))

#step 3 - flattening

classifier.add(Flatten())

#step 4 - full connection

classifier.add(Dense(output\_dim = 128, activation = 'relu'))

classifier.add(Dense(output\_dim = 1, activation = 'sigmoid'))

#compiling the CNN

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

#part 2 - fitting CNN to the images

from keras.preprocessing.image import ImageDataGenerator

train\_datagen = ImageDataGenerator(

rescale=1./255,

shear\_range= 0.2,

zoom\_range= 0.2,

horizontal\_flip=True)

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

training\_set =train\_datagen.flow\_from\_directory(

'C:/Users/1439208/CNNTest/dataset/training\_set',

target\_size=(64, 64),

batch\_size=32,

class\_mode='binary')

test\_set = test\_datagen.flow\_from\_directory(

'C:/Users/1439208/CNNTest/dataset/test\_set',

target\_size=(64, 64),

batch\_size=32,

class\_mode='binary')

from IPython.display import display

from PIL import Image

classifier.fit\_generator(

training\_set,

steps\_per\_epoch=8000,

epochs=10,

validation\_data=test\_set,

validation\_steps=800)

classifier.save('XREADTest.h5')

new\_classifier = load\_model('XREADTest.h5')

import numpy as np

from keras.preprocessing import image

test\_image = image.load\_img('C:/Users/1439208/CNNTest/TestMassPhoto.png', target\_size = (64, 64))

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

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

result = classifier.predict(test\_image)

new\_result = new\_classifier.predict(test\_image)

np.testing.assert\_allclose(result, new\_result, rtol=1e-6, atol=1e-6)

training\_set.class\_indices

if result[0][0] >= 0.5:

prediction = 'No Finding'

else:

prediction = 'Mass Detected'

print(prediction)

if new\_result[0][0] >= 0.5:

new\_prediction = 'No Finding'

else:

new\_prediction = 'Mass Detected'

print(prediction)