# -\*- coding: utf-8 -\*-

"""

Created on Thu Jan 13 03:33:29 2022

@author: WAEL

"""

from keras.models import Sequential

from keras.layers import Conv2D,MaxPooling2D,Dropout,Flatten,Dense,Activation,BatchNormalization,GlobalAveragePooling2D

import numpy as np

import pandas as pd

from pathlib import Path

import os.path

import matplotlib.pyplot as plt

import seaborn as sns

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

import tensorflow as tf

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

image\_dir = Path('../content/drive/MyDrive/X-ray (2)/train')

filepaths = list(image\_dir.glob(r'\*\*/\*.png'))

labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths))

filepaths = pd.Series(filepaths, name='Filepath').astype(str)

labels = pd.Series(labels, name='Label')

image\_df = pd.concat([filepaths, labels], axis=1)

train\_df, test\_df = train\_test\_split(image\_df, train\_size=0.7, shuffle=True, random\_state=1)

train\_generator = tf.keras.preprocessing.image.ImageDataGenerator(

    rescale=1./255,

    horizontal\_flip=True,

    width\_shift\_range=0.2,

    height\_shift\_range=0.2,

    validation\_split=0.2

)

test\_generator = tf.keras.preprocessing.image.ImageDataGenerator(

    rescale=1./255

)

train\_images = train\_generator.flow\_from\_dataframe(

    dataframe=train\_df,

    x\_col='Filepath',

    y\_col='Label',

    target\_size=(224, 224),

    color\_mode='rgb',

    class\_mode='binary',

    batch\_size=32,

    shuffle=True,

    seed=42,

    subset='training'

)

val\_images = train\_generator.flow\_from\_dataframe(

    dataframe=train\_df,

    x\_col='Filepath',

    y\_col='Label',

    target\_size=(224, 224),

    color\_mode='rgb',

    class\_mode='binary',

    batch\_size=32,

    shuffle=True,

    seed=42,

    subset='validation'

)

test\_images = test\_generator.flow\_from\_dataframe(

    dataframe=test\_df,

    x\_col='Filepath',

    y\_col='Label',

    target\_size=(224, 224),

    color\_mode='rgb',

    class\_mode='binary',

    batch\_size=32,

    shuffle=False

)

inputs = tf.keras.Input(shape=(224, 224, 3))

x=tf.keras.layers.Conv2D(32,(3,3),activation='relu')(inputs)

x=tf.keras.layers.BatchNormalization()(x)

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

x=tf.keras.layers.Dropout(0.25)(x)

x=tf.keras.layers.Conv2D(64,(3,3),activation='relu')(x)

x=tf.keras.layers.BatchNormalization()(x)

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

x=tf.keras.layers.Dropout(0.25)(x)

x=tf.keras.layers.Conv2D(128,(3,3),activation='relu')(x)

x=tf.keras.layers.BatchNormalization()(x)

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

x=tf.keras.layers.Dropout(0.25)(x)

x=tf.keras.layers.Flatten()(x)

x=tf.keras.layers.Dense(512,activation='relu')(x)

x=tf.keras.layers.BatchNormalization()(x)

x=tf.keras.layers.Dropout(0.5)(x)

outputs = tf.keras.layers.Dense(1, activation='sigmoid')(x)

model = tf.keras.Model(inputs=inputs, outputs=outputs)

model.compile(loss='categorical\_crossentropy',optimizer='rmsprop',metrics=['accuracy'])

history = model.fit(train\_images,

    validation\_data=val\_images,

    epochs=100,

    callbacks=[

        tf.keras.callbacks.EarlyStopping(

            monitor='val\_loss',

            patience=5,

            restore\_best\_weights=True

        ),

        tf.keras.callbacks.ReduceLROnPlateau(

            monitor='val\_loss',

            patience=3

        )

    ]

)

results = model.evaluate(test\_images, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))

print("Test Accuracy: {:.2f}%".format(results[1] \* 100))

predictions = (model.predict(test\_images) >= 0.5).astype(np.int)

cm = confusion\_matrix(test\_images.labels, predictions, labels=[0, 1])

clr = classification\_report(test\_images.labels, predictions, labels=[0, 1], target\_names=["covid", "normal"])

plt.figure(figsize=(6, 6))

sns.heatmap(cm, annot=True, fmt='g', vmin=0, cmap='Blues', cbar=False)

plt.xticks(ticks=[0.5, 1.5], labels=["covid", "normal"])

plt.yticks(ticks=[0.5, 1.5], labels=["covid", "normal"])

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix")

plt.show()

print("Classification Report:\n----------------------\n", clr)