X-ray image Covid prediction model

December 15, 2021

1 Project Name- Covid virus predictions using X-ray images dataset.

1.1 Author- Shubham Kumar

1.2 Dated- May07,2021

[1]:
```
from imutils import paths
import matplotlib.pyplot as plt
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import AveragePooling2D
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Input
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
```

```
␣
↪---------------------------------------------------------------------------

ModuleNotFoundError                       Traceback (most recent call␣
↪last)

<ipython-input-1-a0a3b13fbb4c> in <module>
----> 1 from imutils import paths
      2 import matplotlib.pyplot as plt
      3 from tensorflow.keras.applications import VGG16
      4 from tensorflow.keras.preprocessing.image import ImageDataGenerator
      5 from tensorflow.keras.layers import AveragePooling2D


ModuleNotFoundError: No module named 'imutils'
```

The first cell consists of importing of various libraries in Python. –> First of all we imported matplotlib.pyplot which is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. –> Now imported various tensorflow layers for fast numerical computing. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries. –> At last I imported sklearn library as it contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

[ ]:
```
dataset= r'/home/aarush100616/Downloads/Data'
```

In above cell I have given location for dataset on which predictions are to be made.It differs from system to system.

[ ]:
```
INIT_LR=1e-3
EPOCHS=10
BS=8
```

The above cell is used to crete model in this first of all INIT_LR is the initial learning rate at which model will learn,EPOCHS is the reading the quantity of images,BS is the number of times you want to throw the images for analysis.

[ ]:
```
args={}
args['dataset']=dataset
```

[ ]:
```
args
```

[ ]:
```
import numpy as np
import cv2
import os



iPaths=list(paths.list_images(args["dataset"]))

data=[]
labels=[]

for iPath in iPaths:
    label=iPath.split(os.path.sep)[-2]
    image=cv2.imread(iPath)
    image=cv2.cvtColor(image,cv2.COLOR_BGR2RGB)
    image=cv2.resize(image,(224,224))
    data.append(image)
    labels.append(label)

data=np.array(data)/255.0
```

```
labels=np.array(labels)
```

[ ]:
```
image
```

[ ]:
```
data
```

[ ]:
```
labels
```

[ ]:
```
import matplotlib.pyplot as plt
import cv2
import skimage
from skimage import filters
Cimages = os.listdir(dataset+"/Covid")
Nimages = os.listdir(dataset+"/Normal")
import numpy as np
def plotter(i):
    normal = cv2.imread(dataset+"/Normal//"+Nimages[i])
    normal = skimage.transform.resize(normal, (150, 150, 3))
    coronavirus = cv2.imread(dataset+"/Covid//"+Cimages[i])
    coronavirus = skimage.transform.resize(coronavirus, (150, 150, 3) , mode =␣
 ↪'reflect')
    pair = np.concatenate((normal, coronavirus), axis=1)
    print("Normal Chest X-ray Vs Covid-19 Chest X-ray")
    plt.figure(figsize=(10,5))
    plt.imshow(pair)
    plt.show()
for i in range(0,5):
    plotter(i)
```

[ ]:
```
LB=LabelBinarizer()
labels=LB.fit_transform(labels)
labels=to_categorical(labels)
print(labels)
```

[ ]:
```
from sklearn.model_selection import train_test_split as sklearn_train_test_split
(X_train,X_test,Y_train,Y_test)=train_test_split(data,labels,test_size=0.
 ↪20,random_state=42,stratify=labels)
```

[ ]:
```
X_train.shape
```

[ ]:
```
trainAug=ImageDataGenerator(rotation_range=15,fill_mode='nearest')
```

[ ]:
```
trainAug
```

[ ]:
```
bmodel=VGG16(weights='imagenet',include_top=False,input_tensor=Input(shape=(224,224,3)))
```

[ ]:
```
bmodel.summary()
```

[ ]:
```
hmodel=bmodel.output
hmodel=AveragePooling2D(pool_size=(4,4))(hmodel)
hmodel=Flatten(name='flatten')(hmodel)
hmodel=Dense(64,activation='relu')(hmodel)
hmodel=Dropout(0.5)(hmodel)
hmodel=Dense(2,activation='softmax')(hmodel)

model=Model(bmodel.input,hmodel)
for layer in bmodel.layers:
    layer.trainable=False
```

[ ]:
```
model.summary()
```

[ ]:
```
W_grid = 4
L_grid = 4
fig, axes= plt.subplots(L_grid,W_grid,figsize=(25,25))
axes=axes.ravel()
n_training=len(X_train)
for i in np.arange(0, L_grid * W_grid):
    index=np.random.randint(0,n_training)
    axes[i].imshow(X_train[index])
    axes[i].set_title(Y_train[index])
    axes[i].axis('off')

plt.subplots_adjust(hspace=0.4)
```

[ ]:
```
opt= Adam(lr=INIT_LR,decay=INIT_LR/EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,metrics=["accuracy"])
print("Compiling Starts")
```

[ ]:
```
R = model.fit_generator(
    trainAug.flow(X_train, Y_train, batch_size=BS),
    steps_per_epoch=len(X_train) // BS,
    validation_data=(X_test, Y_test),
    validation_steps=len(X_test) // BS,
    epochs=EPOCHS)
```

[ ]:
```
L = 6
W = 5
fig, axes = plt.subplots(L, W, figsize = (12, 12))
axes = axes.ravel()
y_pred = model.predict(X_test, batch_size=BS)
for i in np.arange(0,L*W):
    axes[i].imshow(X_test[i])
    axes[i].set_title('Prediction = {}\n True = {}'.format(y_pred.
 ↪argmax(axis=1)[i], Y_test.argmax(axis=1)[i]))
    axes[i].axis('off')
```

```
plt.subplots_adjust(wspace = 1, hspace=1)
```

[ ]:
```
from sklearn.metrics import classification_report
y_pred = model.predict(X_test, batch_size=BS)
y_pred = np.argmax(y_pred, axis=1)
print(classification_report(Y_test.argmax(axis=1), y_pred,target_names=LB.
 ↪classes_))
```

[ ]:
```
#graph for loss
plt.plot(R.history['loss'], label='train loss')
plt.plot(R.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')
```

[ ]:
```
# graph for accuracy
plt.plot(R.history['accuracy'], label='train acc')
plt.plot(R.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
```

[ ]:
```
model.save(r'/home/aarush100616/Downloads/Data/Model.h5')
```

[ ]:
```
import tensorflow as tf
from keras.preprocessing import image
model=tf.keras.models.load_model(r'/home/aarush100616/Downloads/Data/Model.h5')
from keras.applications.vgg16 import preprocess_input
```

[ ]:
```
img = image.load_img('/home/aarush100616/Downloads/Data/Normal/IM-0135-0001.
 ↪jpeg', target_size=(224, 224)) #insert a random covid-19 x-ray image
imgplot = plt.imshow(img)
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)
classes = model.predict(img_data)
New_pred = np.argmax(classes, axis=1)
if New_pred==[1]:
  print('Prediction: Normal')
else:
  print('Prediction: Corona')
```

[ ]:
```
img = image.load_img('/home/aarush100616/Downloads/Data/Covid/1-s2.
 ↪0-S1684118220300682-main.pdf-002-a2.png', target_size=(224, 224)) #insert a␣
 ↪random covid-19 x-ray image
imgplot = plt.imshow(img)
x = image.img_to_array(img)
```

```
x = np.expand_dims(x, axis=0)
img_data = preprocess_input(x)
classes = model.predict(img_data)
New_pred = np.argmax(classes, axis=1)
if New_pred==[1]:
  print('Prediction: Normal')
else:
  print('Prediction: Corona')
```