4. DeeblegDeep/CovidPredict.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text Text size Lines Find and replace Editing

Connect to Google Drive to access Dataset

```
from google.colab import drive
drive.mount('/content/drive')
%cd '/content/drive/My Drive/Colab Notebooks'
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
/content/drive/My Drive/Colab Notebooks
```

Import all Libraries

```
%tensorflow_version 1.x
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
from keras.preprocessing import display
from sklearn.preprocessing import LabelEncoder, MinMaxScaler, StandardScaler
from sklearn.metrics import confusion_matrix, roc_curve
from keras.preprocessing import image
import glob
import os
from glob import glob
from skimage.transform import resize
```

```
%tensorflow_version 1.x if
from tensorflow import keras
from keras.models import Sequential, Model, load_model
from keras.layers import Dense, Conv2D, MaxPool2D, Flatten, AveragePool2D, Dropout, Lambda
from keras.utils.np_utils import to_categorical
from keras.callbacks import EarlyStopping
from keras.preprocessing.image import ImageDataGenerator
from keras.applications.inceptionv3 import InceptionV3, InceptionV3
```

Define Few Parameters

```
IMAGE_SIZE = [224, 224] # Feed Free to change depending on dataset

#define path
train = '/content/Train/Covid'
test = '/content/Train/Normal'
val = '/content/Validation'
train1 = '/content/Train/Covid'
test1 = '/content/Train/Normal'
val1 = '/content/Validation'

# plot glob to get images from path .jpg or .jpeg
train = glob(train + '/*')
test = glob(test + '/*')
val = glob(val + '/*')
train1 = glob(train1 + '/*')
test1 = glob(test1 + '/*')
val1 = glob(val1 + '/*')
```

```
len(train)
```

```
100
```

Fetch Images and Class Labels from Files

```
train = []
test = []
val = []
train1 = []
test1 = []
val1 = []

for i in range(len(train)):
    image = cv2.imread(train[i])
    image = cv2.resize(image,(IMAGE_SIZE[0],IMAGE_SIZE[1]))
    train.append(image)
    train1.append(image)

for i in range(len(test)):
    image = cv2.imread(test[i])
    image = cv2.resize(image,(IMAGE_SIZE[0],IMAGE_SIZE[1]))
    test.append(image)
    test1.append(image)

for i in range(len(val)):
    image = cv2.imread(val[i])
    image = cv2.resize(image,(IMAGE_SIZE[0],IMAGE_SIZE[1]))
    val.append(image)
    val1.append(image)

for i in range(len(train1)):
    image = cv2.imread(train1[i])
    image = cv2.resize(image,(IMAGE_SIZE[0],IMAGE_SIZE[1]))
    train1.append(image)

for i in range(len(test1)):
    image = cv2.imread(test1[i])
    image = cv2.resize(image,(IMAGE_SIZE[0],IMAGE_SIZE[1]))
    test1.append(image)

for i in range(len(val1)):
    image = cv2.imread(val1[i])
    image = cv2.resize(image,(IMAGE_SIZE[0],IMAGE_SIZE[1]))
    val1.append(image)
```

```
len(train1)
```

```
40
```

Visualize from Data set

```
# look at a random image for fun
def plot_imgs(imgpath, ROWS=
    nrows, ncols = 5, 4
    figsize = [16, 10]

    fig, ax = plt.subplots(nrows=nrows, ncols=ncols, figsize=figsize, facecolor=(1, 1, 1))

    for i, ax in enumerate(ax.flat):
        ax[i].imshow(imgpath[i])
        ax[i].axis('off')

    plt.subplots_adjust(wspace=0.02,
    plt.tight_layout(pad=1.5, rect=[0, 0, 1, 0.9])
    plt.show()
```

```
plot_imgs(train, 'COVID-19 Training')
plot_imgs(train1, 'Negative COVID-19 Training')
plot_imgs(val, 'COVID-19 Validation')
plot_imgs(val1, 'Negative COVID-19 Validation')
plot_imgs(test, 'COVID-19 Testing')
plot_imgs(test1, 'Negative COVID-19 Testing')
```

COVID-19 Training

Negative COVID-19 Training

Normalizing the data to help with the training

```
train = np.array(train).astype('float32') / 255
test = np.array(test).astype('float32') / 255
val = np.array(val).astype('float32') / 255
train1 = np.array(train1).astype('float32') / 255
test1 = np.array(test1).astype('float32') / 255
val1 = np.array(val1).astype('float32') / 255
```

Train Test Split

```
train = np.concatenate((train, test), axis=0)
val = np.concatenate((val, val1), axis=0)
X_val = np.concatenate((val, val1), axis=0)
X_train = np.concatenate((train, train1), axis=0)
y_test = np.concatenate((test, test1), axis=0)
y_train = np.concatenate((train1, train1), axis=0)
```

```
print("The data type of X_train, X_train, X_val, y_val, X_train, y_test
,y_train,y_test,y_val,X_train,y_test,y_train,y_test,y_val,y_test,y_test,y_test
```

```
[0.0.0.0000000 0.0.0.0000000 0.0.0.0000000 
```

- **Classification Report**

```
[47] from sklearn.metrics import classification_report
     print(classification_report(y_train_binl1, y_pred_binl1))
```

```
              precision    recall  f1-score   support

           0       0.70      1.00      0.82       300
           1       1.00      0.70      0.85       300

    accuracy                           0.83       600
   macro avg       0.90      0.87      0.87       600
weighted avg       0.90      0.83      0.87       600
```

```
[48] from sklearn.metrics import classification_report
     print(classification_report(y_test_binl1, y_pred_binl1))
```

```
              precision    recall  f1-score   support

           0       0.64      0.89      0.77        80
           1       0.77      0.40      0.52        80

    accuracy                           0.71       160
   macro avg       0.80      0.71      0.69       160
weighted avg       0.80      0.71      0.69       160
```

- **Build Xception-Model**

```
[49] xception = keras.applications.Xception(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

     outputs = xception.output
     outputs = Flatten(name="flatten")(outputs)
     outputs = Dense(128, activation="relu")(outputs)
     outputs = Dense(2, activation="softmax")(outputs)

     model1 = Model(inputs=xception.input, outputs=outputs)

     for layer in xception.layers:
         layer.trainable = False

     #image Augmentation
     train_aug = ImageDataGenerator(rotation_range=15, width_shift_range=0.1, height_shift_range=0.1, horizontal_flip=True)
```

```
[50] model1.summary()
```

```
block2_sepconv1_bn (BatchNorma (None, 16, 16, 728)  2912        block2_sepconv1[0][0]
block2_sepconv1_act (Activatio (None, 16, 16, 728)  0           block2_sepconv1_bn[0][0]
block2_sepconv2_separableconv (None, 16, 16, 728)  536536      block2_sepconv1_act[0][0]
block2_sepconv2_bn (BatchNorma (None, 16, 16, 728)  2912        block2_sepconv2[0][0]
block2_sepconv2_act (Activatio (None, 16, 16, 728)  0           block2_sepconv2_bn[0][0]
block2_sepconv3_separableconv (None, 16, 16, 728)  536536      block2_sepconv2_act[0][0]
block2_sepconv3_bn (BatchNorma (None, 16, 16, 728)  2912        block2_sepconv3[0][0]
add_10 (Add)                   (None, 16, 16, 728)  0           block2_sepconv3_bn[0][0]
                                                                add_9[0][0]
block13_sepconv1_act (Activatio (None, 16, 16, 728)  0           add_10[0][0]
block13_sepconv1_separableconv (None, 16, 16, 728)  536536      block13_sepconv1_act[0][0]
block13_sepconv1_bn (BatchNorma (None, 16, 16, 728)  2912        block13_sepconv1[0][0]
block13_sepconv2_act (Activatio (None, 16, 16, 728)  0           block13_sepconv1_bn[0][0]
block13_sepconv2_separableconv (None, 16, 16, 1024) 752024      block13_sepconv2_act[0][0]
block13_sepconv2_bn (BatchNorma (None, 16, 16, 1024) 4096        block13_sepconv2[0][0]
conv2d_57 (Conv2D)             (None, 7, 7, 1024)   745472      add_10[0][0]
block13_pool (MaxPooling2D)    (None, 7, 7, 1024)   0           block13_sepconv2_bn[0][0]
batch_normalization_37 (BatchNo (None, 7, 7, 1024)  4096        conv2d_57[0][0]
add_11 (Add)                   (None, 7, 7, 1024)   0           block13_pool[0][0]
                                                                batch_normalization_37[0][0]
block14_sepconv1 (SeparableConv (None, 7, 7, 1536)  1582080     add_11[0][0]
block14_sepconv1_bn (BatchNorma (None, 7, 7, 1536)  6144        block14_sepconv1[0][0]
block14_sepconv1_act (Activatio (None, 7, 7, 1536)  0           block14_sepconv1_bn[0][0]
block14_sepconv2 (SeparableConv (None, 7, 7, 2048)  3159552     block14_sepconv1_act[0][0]
block14_sepconv2_bn (BatchNorma (None, 7, 7, 2048)  8192        block14_sepconv2[0][0]
block14_sepconv2_act (Activatio (None, 7, 7, 2048)  0           block14_sepconv2_bn[0][0]
flatten (Flatten)              (None, 100352)       0           block14_sepconv2_act[0][0]
dropout_2 (Dropout)            (None, 100352)       0           flatten[0][0]
dense_2 (Dense)                (None, 2)            200706      dropout_2[0][0]
==================================================================================================
Total params: 21,061,186
Trainable params: 200,706
Non-trainable params: 20,861,480
__________________________________________________________________________________________________
```

- **Compiling Model**

```
[51] model1.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

- **Train the Model**

```
[52] h1 = model1.fit(train_aug.flow(x_train, y_train, batch_size=32), validation_data=(x_val, y_val),epochs=100)
```

```
Epoch 75/100
18/18 [==============================] - 7s 410ms/step - loss: 0.3652 - accuracy: 0.8370 - val_loss: 0.4193 - val_accuracy: 0.8167
Epoch 76/100
18/18 [==============================] - 7s 411ms/step - loss: 0.4098 - accuracy: 0.8194 - val_loss: 0.8962 - val_accuracy: 0.6500
Epoch 77/100
18/18 [==============================] - 7s 414ms/step - loss: 0.3932 - accuracy: 0.8305 - val_loss: 0.8533 - val_accuracy: 0.6500
Epoch 78/100
18/18 [==============================] - 7s 414ms/step - loss: 0.2708 - accuracy: 0.8911 - val_loss: 0.8299 - val_accuracy: 0.6500
Epoch 79/100
18/18 [==============================] - 7s 408ms/step - loss: 0.3079 - accuracy: 0.8541 - val_loss: 1.0089 - val_accuracy: 0.6167
Epoch 80/100
18/18 [==============================] - 7s 406ms/step - loss: 0.4424 - accuracy: 0.8038 - val_loss: 1.0678 - val_accuracy: 0.6333
Epoch 81/100
18/18 [==============================] - 7s 414ms/step - loss: 0.4341 - accuracy: 0.8127 - val_loss: 1.0905 - val_accuracy: 0.6500
Epoch 82/100
18/18 [==============================] - 7s 408ms/step - loss: 0.2647 - accuracy: 0.8544 - val_loss: 0.9648 - val_accuracy: 0.6250
Epoch 83/100
18/18 [==============================] - 7s 409ms/step - loss: 0.3142 - accuracy: 0.8461 - val_loss: 1.6796 - val_accuracy: 0.6167
Epoch 84/100
18/18 [==============================] - 7s 406ms/step - loss: 0.5108 - accuracy: 0.8274 - val_loss: 1.8377 - val_accuracy: 0.6167
Epoch 85/100
18/18 [==============================] - 7s 406ms/step - loss: 0.4882 - accuracy: 0.8012 - val_loss: 1.2885 - val_accuracy: 0.6417
Epoch 86/100
18/18 [==============================] - 7s 406ms/step - loss: 0.2176 - accuracy: 0.9047 - val_loss: 1.0464 - val_accuracy: 0.6333
Epoch 87/100
18/18 [==============================] - 7s 406ms/step - loss: 0.1672 - accuracy: 0.9344 - val_loss: 1.0396 - val_accuracy: 0.6750
Epoch 88/100
18/18 [==============================] - 7s 414ms/step - loss: 0.1964 - accuracy: 0.9048 - val_loss: 1.3606 - val_accuracy: 0.6333
Epoch 89/100
18/18 [==============================] - 7s 406ms/step - loss: 0.2180 - accuracy: 0.9598 - val_loss: 1.2791 - val_accuracy: 0.6083
Epoch 90/100
18/18 [==============================] - 7s 414ms/step - loss: 0.2408 - accuracy: 0.9127 - val_loss: 1.3837 - val_accuracy: 0.6000
Epoch 91/100
18/18 [==============================] - 7s 409ms/step - loss: 0.2317 - accuracy: 0.9519 - val_loss: 1.1423 - val_accuracy: 0.6000
Epoch 92/100
18/18 [==============================] - 7s 407ms/step - loss: 0.2958 - accuracy: 0.9519 - val_loss: 1.1725 - val_accuracy: 0.6333
Epoch 93/100
18/18 [==============================] - 7s 407ms/step - loss: 0.2132 - accuracy: 0.9466 - val_loss: 1.2591 - val_accuracy: 0.6333
Epoch 94/100
18/18 [==============================] - 7s 414ms/step - loss: 0.2238 - accuracy: 0.9579 - val_loss: 0.9828 - val_accuracy: 0.6250
Epoch 95/100
18/18 [==============================] - 7s 414ms/step - loss: 0.3154 - accuracy: 0.9411 - val_loss: 1.4313 - val_accuracy: 0.6083
Epoch 96/100
18/18 [==============================] - 7s 407ms/step - loss: 0.2586 - accuracy: 0.9519 - val_loss: 1.2862 - val_accuracy: 0.6333
Epoch 97/100
18/18 [==============================] - 7s 412ms/step - loss: 0.2560 - accuracy: 0.9582 - val_loss: 1.1875 - val_accuracy: 0.6417
Epoch 98/100
18/18 [==============================] - 7s 405ms/step - loss: 0.2624 - accuracy: 0.9474 - val_loss: 2.2012 - val_accuracy: 0.6000
Epoch 99/100
18/18 [==============================] - 7s 406ms/step - loss: 0.3663 - accuracy: 0.9300 - val_loss: 2.7613 - val_accuracy: 0.6000
Epoch 100/100
18/18 [==============================] - 7s 414ms/step - loss: 0.4349 - accuracy: 0.9279 - val_loss: 1.6615 - val_accuracy: 0.6250
Epoch 1/100
18/18 [==============================] - 7s 415ms/step - loss: 0.4179 - accuracy: 0.9008 - val_loss: 1.1868 - val_accuracy: 0.6167
Epoch 2/100
18/18 [==============================] - 7s 414ms/step - loss: 0.8965 - accuracy: 0.6718 - val_loss: 0.9133 - val_accuracy: 0.6167
Epoch 3/100
18/18 [==============================] - 7s 415ms/step - loss: 0.3932 - accuracy: 0.9381 - val_loss: 0.9903 - val_accuracy: 0.6750
Epoch 100/100
18/18 [==============================] - 7s 414ms/step - loss: 0.3751 - accuracy: 0.9450 - val_loss: 1.3931 - val_accuracy: 0.6000
```

- **Model Evaluate**

```
[53] score = model1.evaluate(x_test, y_test, verbose=0)
     print('test loss:', score[0])
     print('test accuracy:', score[1])

     test loss: 1.2120648382121728
     test accuracy: 0.6724999976158142
```

```
[54] y_pred_test1 = model1.predict(x_train, batch_size=64)

     y_pred_test1 = model1.predict(x_test, batch_size=64)
```

```
[55] print(y_pred_test1,'\n',y_pred_test1)

     [[ 9.9834176e-01 2.1604994e-03]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.0000000e+00 8.6138734e-09]
     [ 1.0000000e+00 0.0000000e+00]
     [ 9.0344054e-04 9.9765656e-01]
     [ 9.9642406e-01 3.0585556e-03]
     [ 1.0000000e+00 0.0000000e+00]
     [ 9.9999894e-01 1.0511224e-06]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.1727752e-04 9.9988264e-01]
     [ 9.9999424e-01 5.5583586e-06]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.0122924e-04 9.9989876e-01]
     [ 1.0000000e+00 2.4616758e-16]
     [ 1.0000000e+00 0.0000000e+00]
     [ 9.6559886e-01 3.4393968e-02]
     [ 9.9999754e-01 4.6271134e-06]
     [ 1.0000000e+00 0.0000000e+00]
     [ 9.9994824e-01 5.1700668e-05]
     [ 1.0000000e+00 1.5750580e-08]
     [ 1.0000000e+00 0.0000000e+00]
     [ 9.9994294e-01 4.9439174e-05]
     [ 7.1608171e-01 2.8976868e-01]
     [ 1.0000000e+00 0.0000000e+00]
     [ 9.9994824e-01 5.1700668e-05]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.0000000e+00 2.0591212e-08]
     [ 9.9993254e-01 6.8012194e-04]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.0000000e+00 1.5384559e-10]
     [ 2.6782174e-08 1.0000000e+00]
     [ 1.0000000e+00 0.0000000e+00]
     [ 9.9539534e-01 4.2460354e-03]
     [ 9.9619584e-01 3.8008146e-03]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.0000000e+00 2.7917388e-09]
     [ 9.9150794e-01 8.5206611e-04]
     [ 1.0000000e+00 0.0000000e+00]
     [ 2.6617064e-02 9.7338296e-01]
     [ 2.6117764e-03 9.9763836e-01]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.0000000e+00 5.5384559e-09]
     [ 8.0783234e-11 1.0000000e+00]
     [ 1.0000000e+00 0.0000000e+00]
     [ 9.4859724e-01 5.5051284e-02]
     [ 9.9999526e-01 4.5976762e-07]
     [ 1.0000000e+00 0.0000000e+00]
     [ 9.9663767e-01 3.3600954e-03]
     [ 9.9682124e-01 3.1475298e-03]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.0000000e+00 2.6712294e-08]
     [ 5.1293382e-01 4.8706054e-01]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.5596768e-01 8.4403174e-01]
     [ 1.0000000e+00 1.2603461e-15]
     [ 1.0000000e+00 0.0000000e+00]
     [ 1.2288839e-01 8.7831616e-01]
     [ 9.9999948e-01 1.2799666e-07]]
```

- **Convert to Binary classes**

```
[56] y_pred_binl1 = np.argmax(y_pred_train1, axis=1)
     y_true_binl1 = np.argmax(y_train, axis=1)
     y_pred_binl2 = np.argmax(y_test1, axis=1)
     y_test_binl1 = np.argmax(y_test, axis=1)
```

```
[57] print(y_pred_binl1,'\n',y_true_binl1,'\n',y_pred_binl2,'\n',y_test_binl1)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 
```