In [ ]:

*# This Python 3 environment comes with many helpful analytics libraries installed*  
*# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python*  
*# For example, here's several helpful packages to load*  
  
**import** **numpy** **as** **np** *# linear algebra*  
**import** **pandas** **as** **pd** *# data processing, CSV file I/O (e.g. pd.read\_csv)*  
  
*# Input data files are available in the read-only "../input/" directory*  
*# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory*  
  
**import** **os**  
**for** dirname, \_, filenames **in** os.walk('/kaggle/input'):  
 **for** filename **in** filenames:  
 print(os.path.join(dirname, filename))  
  
*# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"*   
*# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session*

In [ ]:

**import** **os**  
**import** **numpy** **as** **np**  
**import** **torch**  
**import** **glob**  
**import** **torch.nn** **as** **nn**  
**from** **torchvision.transforms** **import** transforms  
**from** **torch.utils.data** **import** DataLoader  
**from** **torch.optim** **import** Adam  
**from** **torch.autograd** **import** Variable  
**import** **torchvision**  
**import** **pathlib**

In [ ]:

device=torch.device('cude' **if** torch.cuda.is\_available() **else** 'cpu')  
print(device)

In [ ]:

test\_path='../input/chest-xray-covid19-pneumonia/Data/test'  
train\_path='../input/chest-xray-covid19-pneumonia/Data/train'

In [ ]:

**import** **tensorflow** **as** **tf**  
**from** **tensorflow.keras** **import** datasets, layers, models  
**import** **matplotlib.pyplot** **as** **plt**  
**import** **numpy** **as** **np**  
**from** **tensorflow.keras.preprocessing** **import** image  
**from** **tensorflow.keras.preprocessing.image** **import** ImageDataGenerator  
**from** **tensorflow.keras.optimizers** **import** RMSprop  
**from** **keras.models** **import** Sequential  
**from** **keras.layers** **import** Conv2D  
**from** **keras.layers** **import** MaxPooling2D  
**from** **keras.layers** **import** Flatten  
**from** **keras.layers** **import** Dense

In [ ]:

train =ImageDataGenerator(rescale=1/255)  
test =ImageDataGenerator(rescale=1/255)  
train\_dataset =train.flow\_from\_directory(train\_path,target\_size=(256,256),batch\_size=5,color\_mode="rgb",class\_mode="categorical")  
test\_dataset =test.flow\_from\_directory(test\_path,target\_size=(256,256),batch\_size=1,color\_mode="rgb",class\_mode="categorical")

In [ ]:

In [ ]:

train\_dataset.class\_indices  
test\_dataset.class\_indices

In [ ]:

CNN\_Classifier =Sequential()  
CNN\_Classifier.add(Conv2D(32,3,3,input\_shape=(150,150,3),activation='relu'))  
CNN\_Classifier.add(MaxPooling2D(pool\_size=(2,2)))  
  
CNN\_Classifier.add(Conv2D(16,3,3,activation='relu'))  
CNN\_Classifier.add(MaxPooling2D(pool\_size=(2,2)))  
  
CNN\_Classifier.add(Flatten())  
CNN\_Classifier.add(Dense(units=128,activation='relu'))  
CNN\_Classifier.add(Dense(units=3,activation='sigmoid'))

In [ ]:

CNN\_Classifier.compile(optimizer='adam',loss='sparse\_categorical\_crossentropy',metrics=['accuracy'])

In [ ]:

train\_datagen = ImageDataGenerator(  
 rescale=1./255,  
 shear\_range=0.2,  
 zoom\_range=0.2,  
 horizontal\_flip=**True**)  
test\_datagen = ImageDataGenerator(rescale=1./255)  
train\_generator = train\_datagen.flow\_from\_directory(  
 '../input/chest-xray-covid19-pneumonia/Data/train',  
 target\_size=(150, 150),  
 batch\_size=32,  
 class\_mode='binary')  
validation\_generator = test\_datagen.flow\_from\_directory(  
 '../input/chest-xray-covid19-pneumonia/Data/test',  
 target\_size=(150, 150),  
 batch\_size=32,  
 class\_mode='binary')  
CNN\_Classifier.fit(  
 train\_generator,  
 steps\_per\_epoch=20,  
 epochs=50,  
 validation\_data=validation\_generator,  
 validation\_steps=800)

In [ ]:

model =tf.keras.models.Sequential([tf.keras.layers.Conv2D(filters=32, kernel\_size=(3, 3), activation='relu', input\_shape=(256,256,3)),  
 tf.keras.layers.MaxPool2D(2,2),  
   
 tf.keras.layers.Conv2D(14,(3,3),activation ='relu'),  
 tf.keras.layers.MaxPool2D(2,2),  
   
   
   
 tf.keras.layers.Flatten(),  
 tf.keras.layers.Dense(64,activation ='relu'),  
 tf.keras.layers.Dense(3,activation='sigmoid')  
 ])

In [ ]:

model.compile(loss='sparse\_categorical\_crossentropy',  
 *#optimizer='SGD',*  
 optimizer = RMSprop(lr=0.001),  
 metrics=['accuracy'])

In [ ]:

In [ ]:

model\_fit=model.fit(train\_dataset,steps\_per\_epoch=3,epochs=10,validation\_data=test\_dataset)