In [1]:

```
import cv2,os

data_path='D:\Mask Project\dataset'
categories=os.listdir(data_path)
labels=[i for i in range(len(categories))]

label_dict=dict(zip(categories,labels)) #empty dictionary

print(label_dict)
print(categories)
print(labels)
```

```
{'with mask': 0, 'without mask': 1}
['with mask', 'without mask']
[0, 1]
```

In [2]:

```
img_size=100
data=[]
target=[]


for category in categories:
    folder_path=os.path.join(data_path,category)
    img_names=os.listdir(folder_path)

    for img_name in img_names:
        img_path=os.path.join(folder_path,img_name)
        img=cv2.imread(img_path)

        try:
            gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
            #Coverting the image into gray scale
            resized=cv2.resize(gray,(img_size,img_size))
            #resizing the gray scale into 50x50, since we need a fixed common size for
            data.append(resized)
            target.append(label_dict[category])
            #appending the image and the label(categorized) into the list (dataset)

        except Exception as e:
            print('Exception:',e)
            #if any exception rasied, the exception will be printed here. And pass to
```

In [3]:

```
import numpy as np

data=np.array(data)/255.0
data=np.reshape(data,(data.shape[0],img_size,img_size,1))
target=np.array(target)

from keras.utils import np_utils

new_target=np_utils.to_categorical(target)
```

In [4]:

```
np.save('data',data)
np.save('target',new_target)
```

In [5]:

```
import numpy as np
```

```
data=np.load('data.npy')
target=np.load('target.npy')
```

In [6]:

```
from keras.models import Sequential
from keras.layers import Dense,Activation,Flatten,Dropout
from keras.layers import Conv2D,MaxPooling2D
from keras.callbacks import ModelCheckpoint

model=Sequential()

model.add(Conv2D(200,(3,3),input_shape=data.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
#The first CNN layer followed by Relu and MaxPooling layers

model.add(Conv2D(100,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
#The second convolution layer followed by Relu and MaxPooling layers

model.add(Flatten())
model.add(Dropout(0.5))
#Flatten layer to stack the output convolutions from second convolution layer
model.add(Dense(50,activation='relu'))
#Dense layer of 64 neurons
model.add(Dense(2,activation='softmax'))
#The Final layer with two outputs for two categories

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

In [7]:

```
from sklearn.model_selection import train_test_split

train_data,test_data,train_target,test_target=train_test_split(data,target,test_size=
```

In [8]:

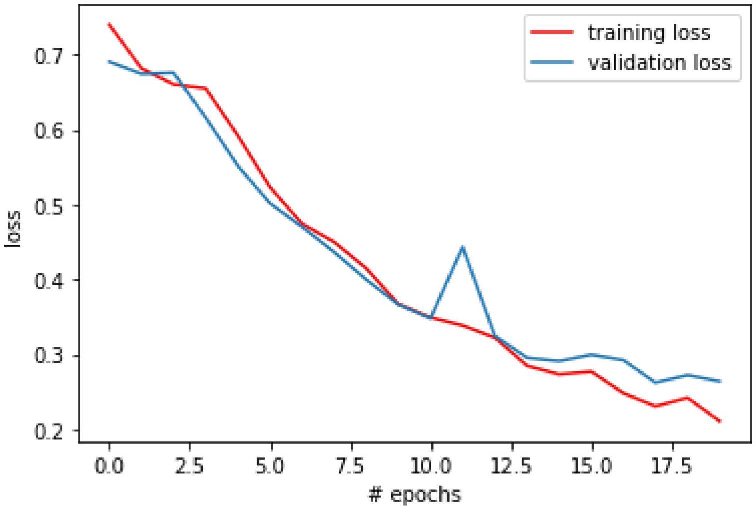
```
checkpoint = ModelCheckpoint('model-{epoch:03d}.model',monitor='val_loss',verbose=0,sa
history=model.fit(train_data,train_target,epochs=20,callbacks=[checkpoint],validation_
Epoch 1/20
31/31 [==============================] - ETA: 0s - loss: 0.7400 - accuracy: 0.4919INF
O:tensorflow:Assets written to: model-001.model\assets
31/31 [==============================] - 69s 2s/step - loss: 0.7400 - accuracy: 0.491
9 - val_loss: 0.6903 - val_accuracy: 0.5040
Epoch 2/20
31/31 [==============================] - ETA: 0s - loss: 0.6814 - accuracy: 0.5505INF
O:tensorflow:Assets written to: model-002.model\assets
31/31 [==============================] - 63s 2s/step - loss: 0.6814 - accuracy: 0.550
5 - val_loss: 0.6740 - val_accuracy: 0.7621
Epoch 3/20
31/31 [==============================] - 60s 2s/step - loss: 0.6601 - accuracy: 0.630
3 - val_loss: 0.6755 - val_accuracy: 0.6331
Epoch 4/20
31/31 [==============================] - ETA: 0s - loss: 0.6546 - accuracy: 0.6414INF
O:tensorflow:Assets written to: model-004.model\assets
31/31 [==============================] - 64s 2s/step - loss: 0.6546 - accuracy: 0.641
4 - val_loss: 0.6155 - val_accuracy: 0.7419
Epoch 5/20
31/31 [==============================] - ETA: 0s - loss: 0.5917 - accuracy: 0.7616INF
O:tensorflow:Assets written to: model-005.model\assets
31/31 [==============================] - 60s 2s/step - loss: 0.5917 - accuracy: 0.761
6 - val_loss: 0.5513 - val_accuracy: 0.7540
Epoch 6/20
31/31 [==============================] - ETA: 0s - loss: 0.5231 - accuracy: 0.8121INF
O:tensorflow:Assets written to: model-006.model\assets
31/31 [==============================] - 61s 2s/step - loss: 0.5231 - accuracy: 0.812
1 - val_loss: 0.5016 - val_accuracy: 0.8145
Epoch 7/20
31/31 [==============================] - ETA: 0s - loss: 0.4744 - accuracy: 0.8485INF
O:tensorflow:Assets written to: model-007.model\assets
31/31 [==============================] - 61s 2s/step - loss: 0.4744 - accuracy: 0.848
5 - val_loss: 0.4703 - val_accuracy: 0.8468
Epoch 8/20
31/31 [==============================] - ETA: 0s - loss: 0.4499 - accuracy: 0.8747INF
O:tensorflow:Assets written to: model-008.model\assets
31/31 [==============================] - 65s 2s/step - loss: 0.4499 - accuracy: 0.874
7 - val_loss: 0.4365 - val_accuracy: 0.8548
Epoch 9/20
31/31 [==============================] - ETA: 0s - loss: 0.4148 - accuracy: 0.8909INF
O:tensorflow:Assets written to: model-009.model\assets
31/31 [==============================] - 74s 2s/step - loss: 0.4148 - accuracy: 0.890
9 - val_loss: 0.3998 - val_accuracy: 0.8831
Epoch 10/20
31/31 [==============================] - ETA: 0s - loss: 0.3671 - accuracy: 0.9323INF
O:tensorflow:Assets written to: model-010.model\assets
31/31 [==============================] - 64s 2s/step - loss: 0.3671 - accuracy: 0.932
3 - val_loss: 0.3664 - val_accuracy: 0.9234
Epoch 11/20
31/31 [==============================] - ETA: 0s - loss: 0.3491 - accuracy: 0.9242INF
O:tensorflow:Assets written to: model-011.model\assets
31/31 [==============================] - 64s 2s/step - loss: 0.3491 - accuracy: 0.924
2 - val_loss: 0.3478 - val_accuracy: 0.9395
Epoch 12/20
31/31 [==============================] - 58s 2s/step - loss: 0.3383 - accuracy: 0.926
3 - val_loss: 0.4433 - val_accuracy: 0.8347
Epoch 13/20
31/31 [==============================] - ETA: 0s - loss: 0.3220 - accuracy: 0.9253INF
O:tensorflow:Assets written to: model-013.model\assets
31/31 [==============================] - 63s 2s/step - loss: 0.3220 - accuracy: 0.925
```

```
3 - val_loss: 0.3246 - val_accuracy: 0.9274
Epoch 14/20
31/31 [==============================] - ETA: 0s - loss: 0.2847 - accuracy: 0.9535INF
O:tensorflow:Assets written to: model-014.model\assets
31/31 [==============================] - 64s 2s/step - loss: 0.2847 - accuracy: 0.953
5 - val_loss: 0.2952 - val_accuracy: 0.9435
Epoch 15/20
31/31 [==============================] - ETA: 0s - loss: 0.2731 - accuracy: 0.9525INF
O:tensorflow:Assets written to: model-015.model\assets
31/31 [==============================] - 59s 2s/step - loss: 0.2731 - accuracy: 0.952
5 - val_loss: 0.2908 - val_accuracy: 0.9476
Epoch 16/20
31/31 [==============================] - 47s 2s/step - loss: 0.2769 - accuracy: 0.947
5 - val_loss: 0.2992 - val_accuracy: 0.9274
Epoch 17/20
31/31 [==============================] - 45s 1s/step - loss: 0.2479 - accuracy: 0.958
6 - val_loss: 0.2921 - val_accuracy: 0.9274
Epoch 18/20
31/31 [==============================] - ETA: 0s - loss: 0.2304 - accuracy: 0.9667INF
O:tensorflow:Assets written to: model-018.model\assets
31/31 [==============================] - 46s 2s/step - loss: 0.2304 - accuracy: 0.966
7 - val_loss: 0.2619 - val_accuracy: 0.9476
Epoch 19/20
31/31 [==============================] - 47s 2s/step - loss: 0.2417 - accuracy: 0.949
5 - val_loss: 0.2721 - val_accuracy: 0.9355
Epoch 20/20
31/31 [==============================] - 44s 1s/step - loss: 0.2106 - accuracy: 0.970
7 - val_loss: 0.2637 - val_accuracy: 0.9435
```

In [9]:

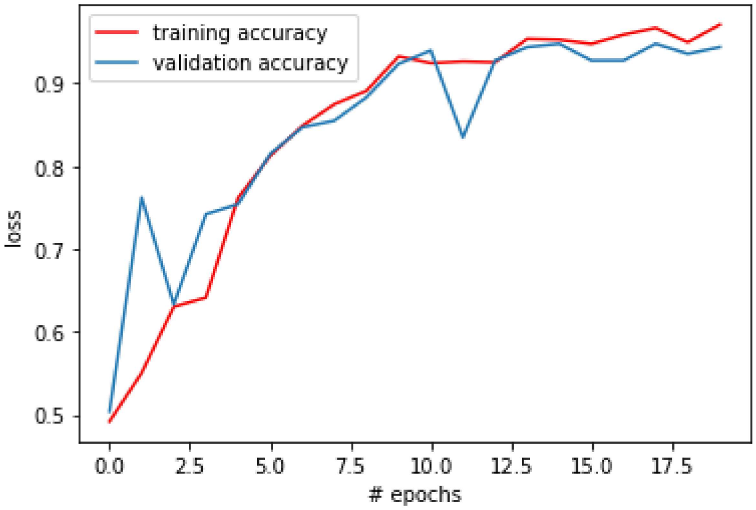
```
from matplotlib import pyplot as plt

plt.plot(history.history['loss'],'r',label='training loss')
plt.plot(history.history['val_loss'],label='validation loss')
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



In [10]:

```
plt.plot(history.history['accuracy'],'r',label='training accuracy')
plt.plot(history.history['val_accuracy'],label='validation accuracy')
plt.xlabel('# epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```



In [11]:

```
print(model.evaluate(test_data,test_target))
```

```
5/5 [==============================] - 2s 304ms/step - loss: 0.2329 - accuracy: 0.942
0
[0.23286134004592896, 0.9420289993286133]
```

In [4]:

```
from keras.models import load_model
import cv2
import sys
import numpy as np
```

In [7]:

```
model = load_model('model-017.model')

face_clsfr=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

source=cv2.VideoCapture(2)

labels_dict={0:'MASK',1:'NO MASK'}
color_dict={0:(0,255,0),1:(0,0,255)}
```

In [ ]:

```
while(True):

    #ret,img=source.read()
    img = cv2.imread('D:\\Mask Project\\dataset\\with mask\\0-with-mask.jpg')
if(img is not None):
    #cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces=face_clsfr.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:

        face_img=gray[y:y+w,x:x+w]
        resized=cv2.resize(face_img,(100,100))
        normalized=resized/255.0
        reshaped=np.reshape(normalized,(1,100,100,1))
        result=model.predict(reshaped)

        label=np.argmax(result,axis=1)[0]
        cv2.rectangle(img,(x,y),(x+w,y+h),color_dict[label],2)
        cv2.rectangle(img,(x,y-40),(x+w,y),color_dict[label],-1)
        cv2.putText(img, labels_dict[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(


    cv2.imshow('LIVE',img)
    key=cv2.waitKey(1)

    if(key==27):
        break

cv2.destroyAllWindows()
source.release()
```

In [ ]:

```
```