Real-Time Face Mask Detection Using Deep Learning and OpenCV

Import Required Libraries

```
import cv2,os
import numpy as np
#from tensorflow.keras.models import Sequential
#from tensorflow.keras.layers import Dense
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import  Dense, Activation, Dropout, Conv2D, Flatten, MaxPooling2D
from keras.callbacks import ModelCheckpoint
from sklearn.model_selection import train_test_split
from matplotlib import pyplot as plt
from keras.models import load_model

import warnings
warnings.filterwarnings("ignore")
```

```
from google.colab import drive
drive.mount('/content/drive')
!ls "/content/drive/My Drive/Face_recognition"
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
with_mask  without_mask
```

```
data= os.listdir('../content/drive/My Drive/Face_recognition')
data
```

```
['without_mask', 'with_mask']
```

```
#use the file path where the dataset is stored
data_path = r'../content/drive/My Drive/Face_recognition'
categories = os.listdir(data_path)
labels = [i for i in range(len(categories))]
label_dict = dict(zip(categories,labels))
print(label_dict)
print(categories)
print(labels)
```

```
{'without_mask': 0, 'with_mask': 1}
['without_mask', 'with_mask']
[0, 1]
```

Make lists for data and target:

```
img_size = 150
```

```
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)
            resized = cv2.resize(gray,(img_size,img_size))
            data.append(resized)
            target.append(label_dict[category])

        except Exception as e:
            print("Exception: ",e)
```

Design a Convolutional Neural Network (CNN) Model

```
#data values are normalized
data = np.array(data)/255.0

#reshaping of data
data = np.reshape(data,(data.shape[0],img_size,img_size,1))
target = np.array(target)
new_target = np_utils.to_categorical(target)

#saving the files
np.save('data',data)
np.save('target',new_target)


data = np.load('../content/drive/My Drive/data_harr/data.npy')
target = np.load('../content/drive/My Drive/data_harr/target.npy')

model = Sequential()

model.add(Conv2D(200,(3,3),input_shape=data.shape[1:]))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Conv2D(100,(3,3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dropout(0.5))

model.add(Dense(50,activation='relu'))
model.add(Dense(2,activation='softmax'))

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

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_4 (Conv2D)            (None, 98, 98, 200)       2000      
_________________________________________________________________
activation_4 (Activation)    (None, 98, 98, 200)       0         
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 49, 49, 200)       0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 47, 47, 100)       180100    
_________________________________________________________________
activation_5 (Activation)    (None, 47, 47, 100)       0         
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 23, 23, 100)       0         
_________________________________________________________________
flatten_2 (Flatten)          (None, 52900)             0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 52900)             0         
_________________________________________________________________
dense_4 (Dense)              (None, 50)                2645050   
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 102       
=================================================================
Total params: 2,827,252
Trainable params: 2,827,252
Non-trainable params: 0
_________________________________________________________________
```

Train the Model

```
#10% of data as testing and 90% as training data
train_data, test_data, train_target, test_target = train_test_split(data, target, test_size=0.1)
```

```
checkpoint=ModelCheckpoint('model-{epoch:03d}.model', monitor='val_loss', verbose = 0, save_best_only = True,
history = model.fit(train_data,train_target,epochs = 20, callbacks = [checkpoint], validation_split = 0.2)
```

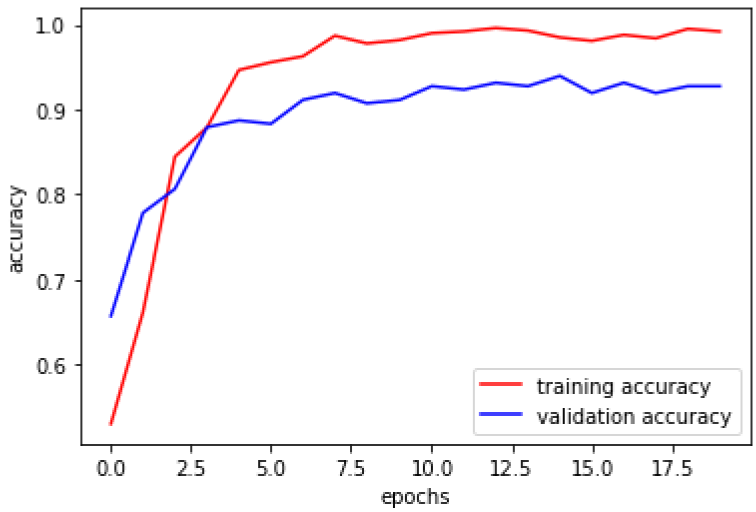
```
Epoch 1/20
31/31 [==============================] - 78s 2s/step - loss: 0.8860 - acc: 0.4987 - val_
INFO:tensorflow:Assets written to: model-001.model/assets
Epoch 2/20
31/31 [==============================] - 91s 3s/step - loss: 0.6306 - acc: 0.6589 - val_
INFO:tensorflow:Assets written to: model-002.model/assets
Epoch 3/20
31/31 [==============================] - 76s 2s/step - loss: 0.4127 - acc: 0.8218 - val_
INFO:tensorflow:Assets written to: model-003.model/assets
Epoch 4/20
31/31 [==============================] - 76s 2s/step - loss: 0.2689 - acc: 0.8739 - val_
INFO:tensorflow:Assets written to: model-004.model/assets
Epoch 5/20
31/31 [==============================] - 75s 2s/step - loss: 0.1688 - acc: 0.9575 - val_
INFO:tensorflow:Assets written to: model-005.model/assets
```

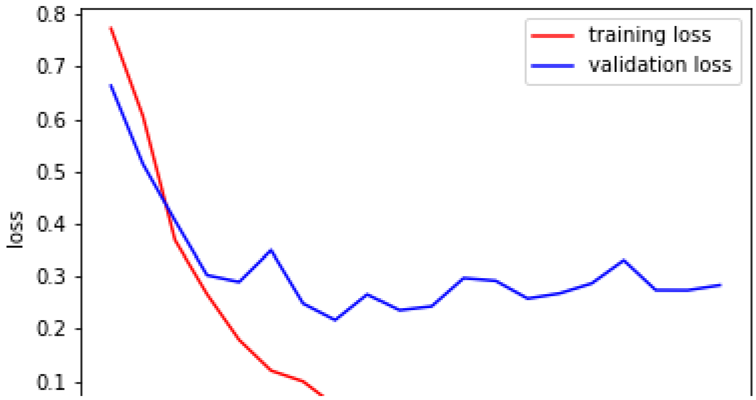
```
Epoch 6/20
31/31 [==============================] - 76s 2s/step - loss: 0.1473 - acc: 0.9447 - val_
Epoch 7/20
31/31 [==============================] - 76s 2s/step - loss: 0.1105 - acc: 0.9567 - val_
INFO:tensorflow:Assets written to: model-007.model/assets
Epoch 8/20
31/31 [==============================] - 76s 2s/step - loss: 0.0431 - acc: 0.9942 - val_
INFO:tensorflow:Assets written to: model-008.model/assets
Epoch 9/20
31/31 [==============================] - 76s 2s/step - loss: 0.0624 - acc: 0.9762 - val_
Epoch 10/20
31/31 [==============================] - 76s 2s/step - loss: 0.0384 - acc: 0.9891 - val_
Epoch 11/20
31/31 [==============================] - 76s 2s/step - loss: 0.0394 - acc: 0.9893 - val_
Epoch 12/20
31/31 [==============================] - 76s 2s/step - loss: 0.0270 - acc: 0.9894 - val_
Epoch 13/20
31/31 [==============================] - 76s 2s/step - loss: 0.0230 - acc: 0.9958 - val_
Epoch 14/20
31/31 [==============================] - 76s 2s/step - loss: 0.0268 - acc: 0.9925 - val_
Epoch 15/20
31/31 [==============================] - 76s 2s/step - loss: 0.0225 - acc: 0.9944 - val_
Epoch 16/20
31/31 [==============================] - 77s 2s/step - loss: 0.0335 - acc: 0.9875 - val_
Epoch 17/20
31/31 [==============================] - 77s 2s/step - loss: 0.0457 - acc: 0.9821 - val_
Epoch 18/20
31/31 [==============================] - 76s 2s/step - loss: 0.0345 - acc: 0.9855 - val_
Epoch 19/20
31/31 [==============================] - 76s 2s/step - loss: 0.0236 - acc: 0.9942 - val_
Epoch 20/20
31/31 [==============================] - 76s 2s/step - loss: 0.0149 - acc: 0.9954 - val_
```

Evaluate the Model

```
plt.plot(history.history['acc'],'r',label='training accuracy')
plt.plot(history.history['val_acc'],'b',label='validation accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()
plt.show()

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






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

```
5/5 [==============================] - 3s 525ms/step - loss: 0.3202 - acc: 0.9348
[0.32017114758491516, 0.9347826242446899]
```