Facemask Detection

Imports

In [6]:

```
import tensorflow as tf
import numpy as np
import os
import matplotlib.pyplot as plt
import cv2
```

DIRECTORIES FOR FILE LOADING

In [9]:

```
main_dir="D:\GIT\Face_Mask_detector using_Convulutional_Neural_Networks\dataset"
ma_dir="D:\GIT\Face_Mask_detector using_Convulutional_Neural_Networks\dataset\with_mask"
nma_dir="D:\GIT\Face_Mask_detector using_Convulutional_Neural_Networks\dataset\without_mask
ima_dir="D:\GIT\Face_Mask_detector using_Convulutional_Neural_Networks\dataset\incorrect_ma
```

No of Training Images Available

In [10]:

```
mask_images=os.listdir(ma_dir)
no_mask_images=os.listdir(nma_dir)
```

In [11]:

```
print("The Number of mask images are",len(mask_images))
print("The Number of no mask images are",len(no_mask_images))
```

```
The Number of mask images are 690
The Number of no mask images are 686
```

Listing of image files

In [12]:

```
print("First 10 Images in masked are \n \n",mask_images[:10])
print(" \n \n \n ")
print("First 10 Images in no mask images are \n \n \n",no_mask_images[:10])
```

```
First 10 Images in masked are

 ['0-with-mask.jpg', '1-with-mask.jpg', '10-with-mask.jpg', '100-with-mask.j
pg', '101-with-mask.jpg', '103-with-mask.jpg', '104-with-mask.jpg', '105-wit
h-mask.jpg', '106-with-mask.jpg', '107-with-mask.jpg']



First 10 Images in no mask images are


 ['0.jpg', '1.jpg', '10.jpg', '100.jpg', '101.jpg', '102.jpg', '104.jpg', '1
05.jpg', '106.jpg', '107.jpg']
```

Using MatplotLib We can show the images of the dataset lets view 8 pictures from each dataset

In [13]:

```
import matplotlib.image as mpimg
nrows = 4
ncols = 4
pic_index = 0
```

In [14]:

```
fig=plt.gcf()
fig.set_size_inches(ncols*5 ,nrows*5)
pic_index+=8
n_m_i=[os.path.join(ma_dir,fname)
      for fname in mask_images[pic_index-8:pic_index]]
n_n_i=[os.path.join(nma_dir,fname)
      for fname in no_mask_images[pic_index-8:pic_index]]

for i,img_path in enumerate(n_m_i+n_n_i):
    sp=plt.subplot(nrows,ncols,i+1)
    sp.axis('Off')
    img=mpimg.imread(img_path)
    plt.imshow(img)
plt.show
```

Out[14]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```

Lets Build our Neural Network

In [9]:

```
model=tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16,(3,3),activation='relu',input_shape=(300,300,3)),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(32,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64,(3,3),activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512,activation='relu'),
    tf.keras.layers.Dense(1,activation='sigmoid')

])
```

In [10]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 298, 298, 16)      448
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 149, 149, 16)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 147, 147, 32)      4640
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 73, 73, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 71, 71, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 35, 35, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 33, 33, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 512)               1606144
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 513
=================================================================
Total params: 1,704,097
Trainable params: 1,704,097
Non-trainable params: 0
_________________________________________________________________
```

MODEL COMPILATION

In [11]:

```
from tensorflow.keras.optimizers import RMSprop

model.compile(loss='binary_crossentropy',
              optimizer=RMSprop(learning_rate=0.001),
              metrics=['accuracy'])
```

Image Data Generator

In [12]:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen=ImageDataGenerator(rescale=1/255)

train_generator=train_datagen.flow_from_directory(
main_dir,
target_size=(300,300),
batch_size=128,
class_mode='binary')
```

```
Found 1376 images belonging to 2 classes.
```

Callback

In [13]:

```
class myCallback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        if(logs.get('accuracy') > 0.95):
            print("\nLoss is lower than 0.4 so cancelling training!")
            self.model.stop_training = True

# Instantiate class
callbacks = myCallback()
```

Fitting The Model

In [14]:

```
fitting=model.fit(train_generator,
                  epochs=15,
                  verbose=1,
                  callbacks=[callbacks])
```

```
Epoch 1/15
11/11 [==============================] - 79s 7s/step - loss: 0.8601 - accura
cy: 0.5741
Epoch 2/15
11/11 [==============================] - 80s 7s/step - loss: 0.8314 - accura
cy: 0.6672
Epoch 3/15
11/11 [==============================] - 78s 7s/step - loss: 0.4663 - accura
cy: 0.8677
Epoch 4/15
11/11 [==============================] - 78s 7s/step - loss: 0.3970 - accura
cy: 0.8794
Epoch 5/15
11/11 [==============================] - 83s 8s/step - loss: 0.4146 - accura
cy: 0.9092
Epoch 6/15
11/11 [==============================] - ETA: 0s - loss: 0.1126 - accuracy:
0.9666
Loss is lower than 0.4 so cancelling training!
11/11 [==============================] - 82s 7s/step - loss: 0.1126 - accura
cy: 0.9666
```

Prediction

We will test on the images

In [2]:

```
from PIL import Image
im=Image.open("C:/Users/hp/Desktop/111111111-min.JPG")
display(im)
```

In [38]:

```
from keras.preprocessing import image

path="C:/Users/hp/Desktop/111111111.JPG"
img=image.load_img(path,target_size=(300,300))
x=image.img_to_array(img)
x/=255
x=np.expand_dims(x,axis=0)
images=np.vstack([x])
classes=model.predict(images,batch_size=10)
print(classes)
if(classes[0]>0.5):
    print("The person in the image is masked")
else:
    print("The person in the image is Unmasked")
```

```
[[0.598744]]
The person in the image is masked
```

Second one is without mask

In [4]:

```
from PIL import Image
im=Image.open("C:/Users/hp/Desktop/1.jpg")
im = im.rotate(90)
display(im)
```

In [32]:

```
from keras.preprocessing import image

path="C:/Users/hp/Desktop/aaaaa.JPG"
img=image.load_img(path,target_size=(300,300))
x=image.img_to_array(img)
x/=255
x=np.expand_dims(x,axis=0)
images=np.vstack([x])
classes=model.predict(images,batch_size=10)
print(classes)
if(classes[0]>0.5):
    print("The person in the image is masked")
else:
    print("The person in the image is Unmasked")
```

```
[[0.00243011]]
The person in the image is Unmasked
```

In [ ]:

```

```

In [ ]:

```

```