In [1]:

```
import tensorflow as tf
import os
```

In [2]:

```
base_dir=r'C:\Users\shive\OneDrive\Desktop\Fase_mask\face-mask-detector'
train_dir=os.path.join(base_dir,'dataset')
mask_dir=os.path.join(train_dir,'with_mask')
without_mask_dir=os.path.join(train_dir,'without_mask')
test_dir=os.path.join(base_dir,'tets')
```

In [3]:

```
mask_fnames=os.listdir(mask_dir)
without_mask_fnames=os.listdir(without_mask_dir)
print(len(mask_fnames),len(without_mask_fnames))
```

```
690 686
```

In [4]:

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt


nrows = 4
ncols = 4

pic_index = 0
```

In [5]:

```
fig = plt.gcf()
fig.set_size_inches(ncols*4, nrows*4)

pic_index+=8

next_mask_pix = [os.path.join(mask_dir, fname)
                for fname in mask_fnames[ pic_index-8:pic_index]
               ]

next_wmask_pix = [os.path.join(without_mask_dir, fname)
                for fname in without_mask_fnames[ pic_index-8:pic_index]
               ]

for i, img_path in enumerate(next_mask_pix+next_wmask_pix):

  sp = plt.subplot(nrows, ncols, i + 1)
  sp.axis('Off')

  img = mpimg.imread(img_path)
  plt.imshow(img)

plt.show()
```

In [6]:

```
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.preprocessing.image import load_img
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
import numpy as np
data=[]
labels=[]

for fnames in mask_fnames:
    path=r'C:\Users\shive\OneDrive\Desktop\Fase_mask\face-mask-detector\dataset\with_mask'
    path=path+'\\'+fnames
    image = load_img(path, target_size=(150, 150))
    image = img_to_array(image)
    image = preprocess_input(image)

    data.append(image)
    labels.append(1)

for fnames in   without_mask_fnames:
    path=r'C:\Users\shive\OneDrive\Desktop\Fase_mask\face-mask-detector\dataset\without_mas
    path=path+'\\'+fnames

    image = load_img(path, target_size=(150, 150))
    image = img_to_array(image)
    image = preprocess_input(image)

    data.append(image)
    labels.append(0)
data = np.array(data, dtype="float32")
labels = np.array(labels)
```

In [7]:

```
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.utils import to_categorical
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)
print(data.shape,labels.shape)
```

```
(1376, 150, 150, 3) (1376, 2)
```

In [9]:

```
from sklearn.model_selection import train_test_split

(train_X, test_X, train_Y, test_Y) = train_test_split(data, labels,test_size=0.2, random_st
```

In [10]:

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


train_datagen = ImageDataGenerator(
                                   rotation_range=40,
                                width_shift_range=.2,
                                height_shift_range=.2,
                                shear_range=.2,
                                zoom_range=.2,
                                fill_mode='nearest',
                                horizontal_flip=True

                                  )

test_datagen  = ImageDataGenerator(
                                  )


train_generator = train_datagen.flow(train_X,
                                     train_Y,
                                    batch_size=20,

                                                    )

validation_generator =  test_datagen.flow(test_X,
                                          test_Y,
                                        batch_size=20,

                                          )
```

In [24]:

```
model = tf.keras.models.Sequential([
        tf.keras.layers. AveragePooling2D((5,5),input_shape=(150,150,3)),

#         tf.keras.layers.Conv2D(64,(3,3),activation='relu',input_shape=(150,150,3)),
#         tf.keras.layers.MaxPooling2D(2,2),

#         tf.keras.layers.Conv2D(128,(3,3),activation='relu'),
#         tf.keras.layers.MaxPooling2D(2,2),


        tf.keras.layers.Flatten(),

        tf.keras.layers.Dense(128,activation='relu'),

        tf.keras.layers.Dropout(0.5),

        tf.keras.layers.Dense(2,activation='softmax')

])
```

In [25]:

```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
average_pooling2d (AveragePo (None, 30, 30, 3)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2700)              0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               345728
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 258
=================================================================
Total params: 345,986
Trainable params: 345,986
Non-trainable params: 0
_________________________________________________________________
```

In [26]:

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.optimizers import Adam
model.compile(loss='binary_crossentropy',optimizer=Adam(learning_rate=0.0001,epsilon=1e-05)

class myCallback(tf.keras.callbacks.Callback):
        def on_epoch_end(self,epoch,logs={}):
            if logs.get('accuracy')>0.95:
                print('Reached 95% accuracy so cancelling training!')
                self.model.stop_training=True
```

In [27]:

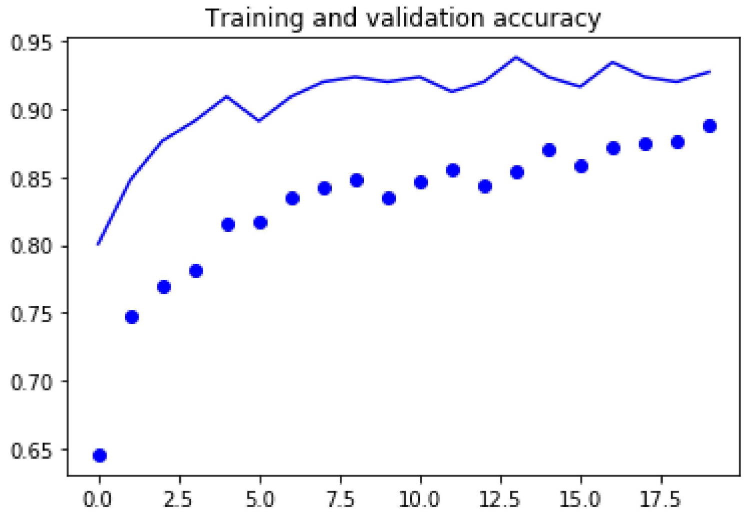
```
callbacks=myCallback()
history=model.fit(train_generator,
                  epochs=20,

                  validation_data=validation_generator,
                     callbacks=[callbacks])
```

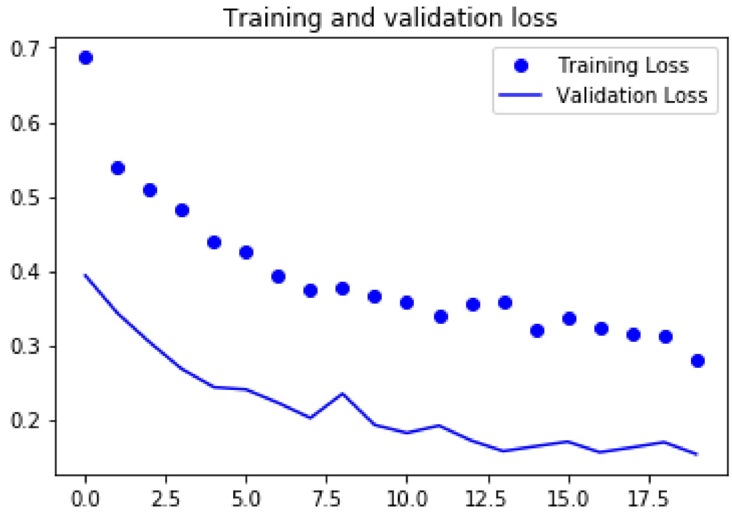
```
Epoch 1/20
55/55 [==============================] - 16s 297ms/step - loss: 0.6871 - acc
uracy: 0.6455 - val_loss: 0.3935 - val_accuracy: 0.8007
Epoch 2/20
55/55 [==============================] - 17s 309ms/step - loss: 0.5404 - acc
uracy: 0.7482 - val_loss: 0.3428 - val_accuracy: 0.8478
Epoch 3/20
55/55 [==============================] - 17s 307ms/step - loss: 0.5094 - acc
uracy: 0.7700 - val_loss: 0.3037 - val_accuracy: 0.8768
Epoch 4/20
55/55 [==============================] - 16s 297ms/step - loss: 0.4819 - acc
uracy: 0.7818 - val_loss: 0.2678 - val_accuracy: 0.8913
Epoch 5/20
55/55 [==============================] - 16s 300ms/step - loss: 0.4386 - acc
uracy: 0.8155 - val_loss: 0.2433 - val_accuracy: 0.9094
Epoch 6/20
55/55 [==============================] - 16s 291ms/step - loss: 0.4263 - acc
uracy: 0.8173 - val_loss: 0.2403 - val_accuracy: 0.8913
Epoch 7/20
55/55 [==============================] - 16s 288ms/step - loss: 0.3936 - acc
uracy: 0.8345 - val_loss: 0.2222 - val_accuracy: 0.9094
Epoch 8/20
55/55 [==============================] - 16s 283ms/step - loss: 0.3733 - acc
uracy: 0.8427 - val_loss: 0.2019 - val_accuracy: 0.9203
Epoch 9/20
55/55 [==============================] - 16s 286ms/step - loss: 0.3778 - acc
uracy: 0.8482 - val_loss: 0.2346 - val_accuracy: 0.9239
Epoch 10/20
55/55 [==============================] - 16s 286ms/step - loss: 0.3666 - acc
uracy: 0.8345 - val_loss: 0.1926 - val_accuracy: 0.9203
Epoch 11/20
55/55 [==============================] - 16s 285ms/step - loss: 0.3578 - acc
uracy: 0.8473 - val_loss: 0.1818 - val_accuracy: 0.9239
Epoch 12/20
55/55 [==============================] - 16s 295ms/step - loss: 0.3392 - acc
uracy: 0.8564 - val_loss: 0.1917 - val_accuracy: 0.9130
Epoch 13/20
55/55 [==============================] - 16s 294ms/step - loss: 0.3566 - acc
uracy: 0.8445 - val_loss: 0.1716 - val_accuracy: 0.9203
Epoch 14/20
55/55 [==============================] - 16s 293ms/step - loss: 0.3580 - acc
uracy: 0.8536 - val_loss: 0.1573 - val_accuracy: 0.9384
Epoch 15/20
55/55 [==============================] - 16s 290ms/step - loss: 0.3210 - acc
uracy: 0.8700 - val_loss: 0.1638 - val_accuracy: 0.9239
Epoch 16/20
55/55 [==============================] - 16s 290ms/step - loss: 0.3377 - acc
uracy: 0.8582 - val_loss: 0.1701 - val_accuracy: 0.9167
Epoch 17/20
55/55 [==============================] - 16s 293ms/step - loss: 0.3230 - acc
```

```
uracy: 0.8727 - val_loss: 0.1558 - val_accuracy: 0.9348
Epoch 18/20
55/55 [==============================] - 16s 299ms/step - loss: 0.3141 - acc
uracy: 0.8755 - val_loss: 0.1622 - val_accuracy: 0.9239
Epoch 19/20
55/55 [==============================] - 16s 286ms/step - loss: 0.3132 - acc
uracy: 0.8764 - val_loss: 0.1694 - val_accuracy: 0.9203
Epoch 20/20
55/55 [==============================] - 16s 298ms/step - loss: 0.2811 - acc
uracy: 0.8882 - val_loss: 0.1533 - val_accuracy: 0.9275
```

In [28]:

```
import matplotlib.pyplot as plt
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training Loss')

plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





In [29]:

```
from tensorflow.keras.preprocessing import image
import numpy as np

path=r'C:\Users\shive\OneDrive\Desktop\Fase_mask\face-mask-detector\examples\example_02.png






image = load_img(path, target_size=(150, 150))
image = img_to_array(image)
image = preprocess_input(image)

image = np.expand_dims(image, axis=0)
image = np.vstack([image])



classes = model.predict(image)
classes
```

Out[29]:

```
array([[0.959543  , 0.04045695]], dtype=float32)
```

In [30]:

```
import cv2
labels_dict={1:'without_mask',0:'with_mask'}
color_dict={1:(0,0,255),0:(0,255,0)}

size = 4
webcam = cv2.VideoCapture(0) #Use camera 0


classifier = cv2.CascadeClassifier(r'C:\Users\shive\OneDrive\Desktop\haarcascade_frontalfac

while True:
    (rval, im) = webcam.read()
    im=cv2.flip(im,1,1)


    mini = cv2.resize(im, (im.shape[1] // size, im.shape[0] // size))


    faces = classifier.detectMultiScale(mini)


    for f in faces:
        (x, y, w, h) = [v * size for v in f]

        face_img = im[y:y+h, x:x+w]
        resized=cv2.resize(face_img,(150,150))
        image= preprocess_input(resized)
        reshaped=np.expand_dims(image, axis=0)
        reshaped = np.vstack([reshaped])
        result=model.predict(reshaped)
        #print(result)

        label=np.argmax(result,axis=1)[0]

        cv2.rectangle(im,(x,y),(x+w,y+h),color_dict[label],2)
        cv2.rectangle(im,(x,y-40),(x+w,y),color_dict[label],-1)
        cv2.putText(im, labels_dict[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255


    cv2.imshow('LIVE',   im)
    key = cv2.waitKey(10)

    if key == 27: #esc
        break

webcam.release()


cv2.destroyAllWindows()
```

In [31]:

```
model.save(r'C:\Users\shive\OneDrive\Desktop\my_model.h5')
```

In [ ]: