FaceMaskDetection

January 4, 2022

[1]:
```
# Import Library
import numpy as np
import os
import pandas as pd
import matplotlib.pyplot as plt
```

[77]:
```
# Dataset path
test_path = '/home/artpark/personal project/Face Mask Detection/data/test/'
train_path = '/home/artpark/personal project/Face Mask Detection/data/train/'
```

[78]:
```
os.listdir(test_path)
```

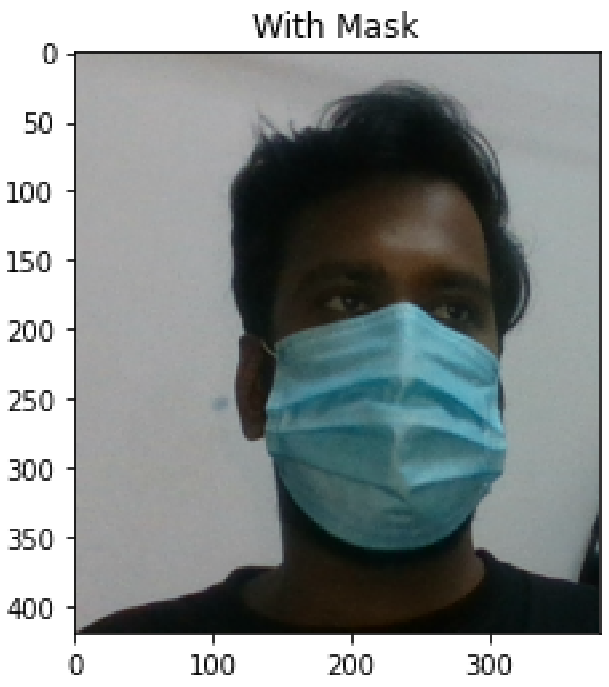
[78]:
```
['NoMask', 'Mask']
```

[79]:
```
os.listdir(train_path)
```

[79]:
```
['NoMask', 'Mask']
```

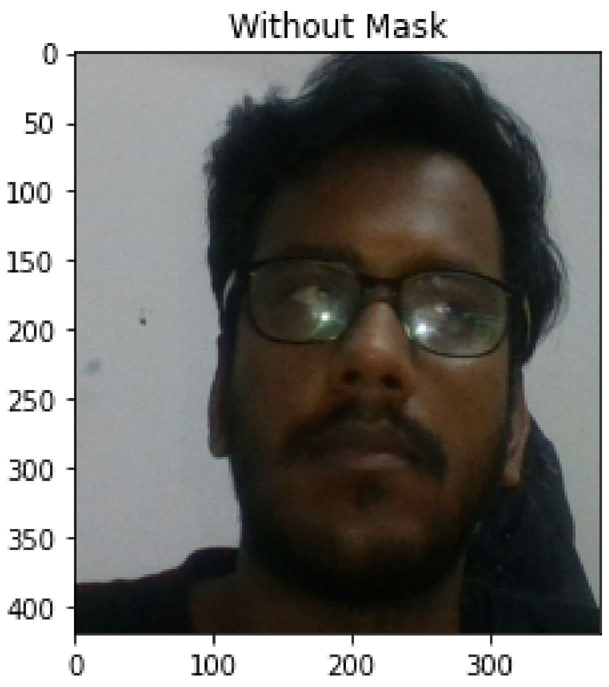
[83]:
```
v = test_path + 'Mask'
#os.listdir(v)
img_path = v + '/image1_134.png'
img_r = plt.imread(img_path)
plt.imshow(img_r)
plt.title('With Mask')
```

[83]:
```
Text(0.5, 1.0, 'With Mask')
```



```
[82]: v = test_path + 'NoMask'
      #os.listdir(v)
      img_path = v + '/image3_27.png'
      img_r = plt.imread(img_path)
      plt.imshow(img_r)
      plt.title('Without Mask')
```

```
[82]: Text(0.5, 1.0, 'Without Mask')
```



[69]:
```
# Check for average dim of image in dataset
dim1 = []
dim2 = []
for image in os.listdir(test_path + 'Mask'):
    pic = plt.imread(test_path + 'Mask/' + image)
#     print(pic.shape)
    d1, d2, c = pic.shape
    dim1.append(d1)
    dim2.append(d2)
```

[7]:
```
np.mean(dim1)
```

[7]:
```
390.4142857142857
```

[8]:
```
np.mean(dim2)
```

[8]:
```
532.6023809523809
```

[9]:
```
# Set image shape
image_shape = (370, 430, 3)
batch_size = 16
```

[10]:
```
# pre-processing of dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
image_gen = ImageDataGenerator(rotation_range = 10,
                              width_shift_range = 0.10,
                              height_shift_range = 0.10,
                              rescale = 1/255,
                              shear_range = 0.1,
                              zoom_range = 0.05,
                              horizontal_flip = True,
                              fill_mode = 'nearest')
```

[11]:
```
# Create Train - Test generator, which will used to train the model
train_image_gen = image_gen.flow_from_directory(train_path,
                                                 target_size=image_shape[:2],
                                                 color_mode='rgb',
                                                 batch_size=batch_size,
                                                 class_mode='binary')

test_image_gen = image_gen.flow_from_directory(test_path,
                                                target_size=image_shape[:2],
                                                color_mode='rgb',
                                                batch_size=batch_size,
                                                class_mode='binary',
                                                shuffle=False)
```

```
Found 4286 images belonging to 2 classes.
Found 835 images belonging to 2 classes.
```

[12]:
```
#check
train_image_gen.class_indices
```

[12]:
```
{'Mask': 0, 'NoMask': 1}
```

[13]:
```
# Start Creating the model
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, BatchNormalization, MaxPool2D,␣
 ↪Flatten, Dense, Dropout
```

[70]:
```
model = Sequential()
model.add(Conv2D(filters=64, kernel_size=(3, 3), input_shape = image_shape,␣
 ↪activation= 'relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation= 'relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation= 'relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation= 'relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation= 'relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(filters=32, kernel_size=(3, 3), activation= 'relu'))
model.add(MaxPool2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.3))
model.add(Dense(1, activation='sigmoid'))

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

[15]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 368, 428, 64)      1792
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 184, 214, 64)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 182, 212, 64)      36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 91, 106, 64)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 89, 104, 64)       36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 44, 52, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 42, 50, 32)        18464
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 21, 25, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 19, 23, 32)        9248
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 9, 11, 32)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 7, 9, 32)          9248
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 3, 4, 32)          0
_________________________________________________________________
flatten (Flatten)            (None, 384)               0
_________________________________________________________________
dense (Dense)                (None, 256)               98560
```

```
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                16448
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 227,681
Trainable params: 227,681
Non-trainable params: 0
_________________________________________________________________
```

[16]:
```
# Apply early stopping criteria
from tensorflow.keras.callbacks import EarlyStopping
early_stop = EarlyStopping(monitor='val_loss', patience=10)
```

[17]:
```
model.fit(train_image_gen, epochs=50, validation_data=test_image_gen,␣
 ↪callbacks=[early_stop])
```

```
Epoch 1/50

2022-01-04 18:08:05.449010: I tensorflow/stream_executor/cuda/cuda_dnn.cc:366]
Loaded cuDNN version 8201

 86/268 [========>…] - ETA: 57s - loss: 0.5056 - accuracy:
0.7275

/home/artpark/anaconda3/envs/tf/lib/python3.9/site-packages/PIL/Image.py:975:
UserWarning: Palette images with Transparency expressed in bytes should be
converted to RGBA images
  warnings.warn(

268/268 [==============================] - 108s 391ms/step - loss: 0.3019 -
accuracy: 0.8712 - val_loss: 0.4625 - val_accuracy: 0.8431
Epoch 2/50
268/268 [==============================] - 105s 390ms/step - loss: 0.1763 -
accuracy: 0.9475 - val_loss: 0.4587 - val_accuracy: 0.8323
Epoch 3/50
268/268 [==============================] - 106s 394ms/step - loss: 0.1384 -
accuracy: 0.9573 - val_loss: 0.5106 - val_accuracy: 0.8635
Epoch 4/50
268/268 [==============================] - 103s 386ms/step - loss: 0.1320 -
accuracy: 0.9631 - val_loss: 0.4088 - val_accuracy: 0.8539
Epoch 5/50
268/268 [==============================] - 104s 389ms/step - loss: 0.1233 -
accuracy: 0.9617 - val_loss: 0.7898 - val_accuracy: 0.7497
Epoch 6/50
268/268 [==============================] - 104s 389ms/step - loss: 0.1173 -
```

```
accuracy: 0.9648 - val_loss: 1.1426 - val_accuracy: 0.7617
Epoch 7/50
268/268 [==============================] - 109s 407ms/step - loss: 0.1132 -
accuracy: 0.9671 - val_loss: 0.5130 - val_accuracy: 0.8599
Epoch 8/50
268/268 [==============================] - 111s 414ms/step - loss: 0.0967 -
accuracy: 0.9697 - val_loss: 0.4116 - val_accuracy: 0.8623
Epoch 9/50
268/268 [==============================] - 108s 403ms/step - loss: 0.1068 -
accuracy: 0.9680 - val_loss: 0.4434 - val_accuracy: 0.8527
Epoch 10/50
268/268 [==============================] - 109s 408ms/step - loss: 0.1009 -
accuracy: 0.9671 - val_loss: 0.6143 - val_accuracy: 0.8431
Epoch 11/50
268/268 [==============================] - 109s 408ms/step - loss: 0.0971 -
accuracy: 0.9715 - val_loss: 0.3091 - val_accuracy: 0.8743
Epoch 12/50
268/268 [==============================] - 108s 403ms/step - loss: 0.0961 -
accuracy: 0.9760 - val_loss: 0.3952 - val_accuracy: 0.8731
Epoch 13/50
268/268 [==============================] - 108s 403ms/step - loss: 0.1006 -
accuracy: 0.9659 - val_loss: 0.3072 - val_accuracy: 0.8910
Epoch 14/50
268/268 [==============================] - 109s 407ms/step - loss: 0.0938 -
accuracy: 0.9727 - val_loss: 0.6177 - val_accuracy: 0.7832
Epoch 15/50
268/268 [==============================] - 109s 408ms/step - loss: 0.0860 -
accuracy: 0.9713 - val_loss: 0.3566 - val_accuracy: 0.8659
Epoch 16/50
268/268 [==============================] - 108s 402ms/step - loss: 0.0789 -
accuracy: 0.9755 - val_loss: 0.2829 - val_accuracy: 0.9030
Epoch 17/50
268/268 [==============================] - 108s 403ms/step - loss: 0.0867 -
accuracy: 0.9713 - val_loss: 0.3827 - val_accuracy: 0.8778
Epoch 18/50
268/268 [==============================] - 109s 407ms/step - loss: 0.0797 -
accuracy: 0.9762 - val_loss: 0.3801 - val_accuracy: 0.8515
Epoch 19/50
268/268 [==============================] - 108s 404ms/step - loss: 0.0696 -
accuracy: 0.9804 - val_loss: 0.2838 - val_accuracy: 0.8970
Epoch 20/50
268/268 [==============================] - 108s 404ms/step - loss: 0.0842 -
accuracy: 0.9713 - val_loss: 0.3650 - val_accuracy: 0.8443
Epoch 21/50
268/268 [==============================] - 108s 403ms/step - loss: 0.0720 -
accuracy: 0.9750 - val_loss: 0.2768 - val_accuracy: 0.8922
Epoch 22/50
268/268 [==============================] - 108s 403ms/step - loss: 0.0612 -
```

```
accuracy: 0.9811 - val_loss: 0.3802 - val_accuracy: 0.8707
Epoch 23/50
268/268 [==============================] - 104s 388ms/step - loss: 0.0810 -
accuracy: 0.9729 - val_loss: 0.4348 - val_accuracy: 0.8754
Epoch 24/50
268/268 [==============================] - 107s 400ms/step - loss: 0.0626 -
accuracy: 0.9818 - val_loss: 0.3484 - val_accuracy: 0.9018
Epoch 25/50
268/268 [==============================] - 107s 400ms/step - loss: 0.0731 -
accuracy: 0.9767 - val_loss: 0.2345 - val_accuracy: 0.9174
Epoch 26/50
268/268 [==============================] - 108s 404ms/step - loss: 0.0529 -
accuracy: 0.9813 - val_loss: 0.3329 - val_accuracy: 0.8970
Epoch 27/50
268/268 [==============================] - 107s 400ms/step - loss: 0.0552 -
accuracy: 0.9806 - val_loss: 0.2565 - val_accuracy: 0.9066
Epoch 28/50
268/268 [==============================] - 108s 402ms/step - loss: 0.0708 -
accuracy: 0.9769 - val_loss: 0.3132 - val_accuracy: 0.8790
Epoch 29/50
268/268 [==============================] - 109s 409ms/step - loss: 0.0557 -
accuracy: 0.9830 - val_loss: 0.2797 - val_accuracy: 0.8982
Epoch 30/50
268/268 [==============================] - 108s 401ms/step - loss: 0.0562 -
accuracy: 0.9818 - val_loss: 0.2901 - val_accuracy: 0.9150
Epoch 31/50
268/268 [==============================] - 110s 410ms/step - loss: 0.0780 -
accuracy: 0.9743 - val_loss: 0.3213 - val_accuracy: 0.8802
Epoch 32/50
268/268 [==============================] - 110s 410ms/step - loss: 0.0660 -
accuracy: 0.9811 - val_loss: 0.2835 - val_accuracy: 0.9150
Epoch 33/50
268/268 [==============================] - 110s 411ms/step - loss: 0.0536 -
accuracy: 0.9837 - val_loss: 0.3570 - val_accuracy: 0.9198
Epoch 34/50
268/268 [==============================] - 111s 414ms/step - loss: 0.0517 -
accuracy: 0.9839 - val_loss: 0.3244 - val_accuracy: 0.9042
Epoch 35/50
268/268 [==============================] - 110s 411ms/step - loss: 0.0513 -
accuracy: 0.9811 - val_loss: 0.2546 - val_accuracy: 0.9006
```

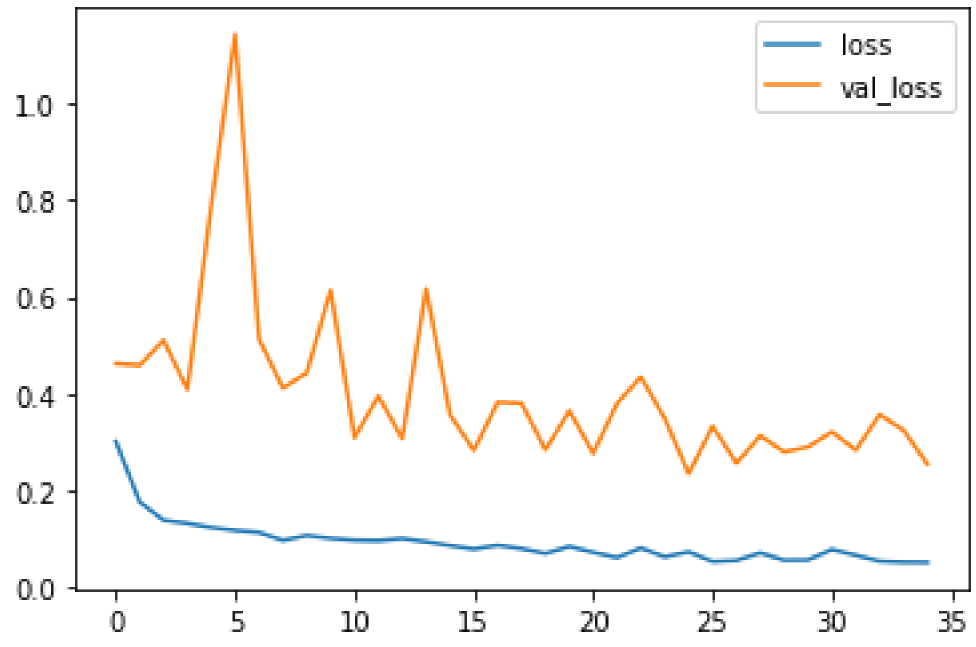
```
[17]: <tensorflow.python.keras.callbacks.History at 0x7f6974074910>
```

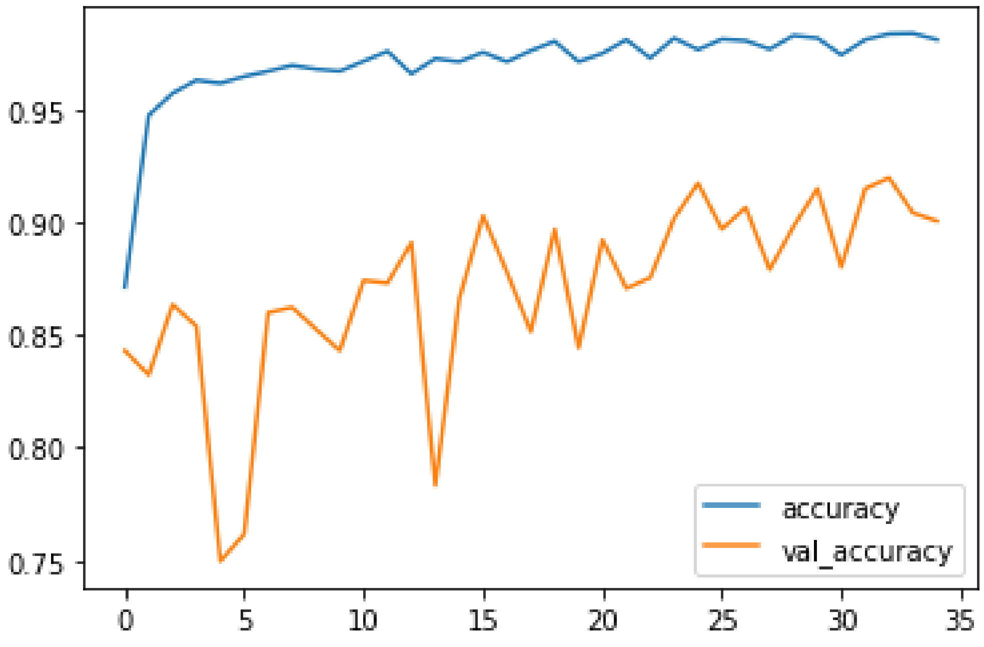
```
[18]: model.save('FaceMaskDetection4.h5')
```

```
[19]: loss = pd.DataFrame(model.history.history)
      loss[['loss', 'val_loss']].plot()
```

```
loss[['accuracy', 'val_accuracy']].plot()
```

```
[19]: <AxesSubplot:>
```





```
[74]: model.evaluate_generator(test_image_gen)
```

```
[76]: predd = model.predict_generator(test_image_gen)
```

```
[71]: #predd
```

```
[23]: prediction = predd > 0.5
```

```
[72]: #prediction
```

```
[25]: from sklearn.metrics import confusion_matrix, classification_report
      print(confusion_matrix(test_image_gen.classes, prediction))
      print('*****************************************************')
      print(classification_report(test_image_gen.classes, prediction))
```

```
[[397  23]
 [ 50 365]]
*****************************************************
              precision    recall  f1-score   support

           0       0.89      0.95      0.92       420
           1       0.94      0.88      0.91       415

    accuracy                           0.91       835
   macro avg       0.91      0.91      0.91       835
weighted avg       0.91      0.91      0.91       835
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]: #######################################################################################
     ############## Load saved Model    ␣
      ↪###################################################
     #######################################################################################
```

```
[66]: from tensorflow.keras.models import load_model
      import cv2
      from sklearn.metrics import confusion_matrix, classification_report
      import numpy as np
      from tensorflow.keras.preprocessing import image
```

```
[67]: new_model = load_model('FaceMaskDetection4.h5')
```

[ ]:
```
image_shape = (370, 430, 3)
image2 = image.load_img('with_mask271.jpg', target_size=image_shape[:2])
my_img_arr = image.img_to_array(image2)              #check my_img_arr.shape␣
 ↪should be ---> (370, 430, 3)
# print(my_img_arr.shape)
my_img_arr = np.expand_dims(my_img_arr, axis = 0)    #check my_img_arr.shape␣
 ↪should be ---> (1, 370, 430, 3)
print(my_img_arr.shape)
predd = new_model.predict(my_img_arr)
```

[ ]:
```
```