PROJECT

Face Mask Detection using CNN

Table of Contents

Importing the Dependencies

```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
from PIL import Image
from sklearn.model_selection import train_test_split
```

```
/opt/conda/lib/python3.10/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this
version of SciPy (detected version 1.23.5
  warnings.warn(f"A NumPy version >={np_minversion} and
<{np_maxversion}"
```

```
mask_files =
os.listdir('/kaggle/input/face-mask-dataset/data/with_mask')
print(mask_files[0:5])
print(mask_files[-5:])
```

```
['with_mask_3326.jpg', 'with_mask_3139.jpg', 'with_mask_696.jpg',
'with_mask_2867.jpg', 'with_mask_39.jpg']
['with_mask_502.jpg', 'with_mask_110.jpg', 'with_mask_3205.jpg',
'with_mask_1863.jpg', 'with_mask_2020.jpg']
```

```
nomask_files =
os.listdir('/kaggle/input/face-mask-dataset/data/without_mask')
print(nomask_files[0:5])
print(nomask_files[-5:])
```

```
['without_mask_3248.jpg', 'without_mask_2803.jpg',
'without_mask_650.jpg', 'without_mask_2060.jpg',
'without_mask_559.jpg']
['without_mask_3215.jpg', 'without_mask_2934.jpg',
'without_mask_2572.jpg', 'without_mask_1906.jpg',
'without_mask_2551.jpg']
```

```
print(f'Number of with mask images:{len(mask_files)}')
print(f'Number of without mask images:{len(nomask_files)}')
```

```
Number of with mask images:3725
Number of without mask images:3828
```

Creating Labels for the two class of Images

```
# create the labels

mask_files = [1]*3725
nomask_files = [0]*3828

print(mask_files[0:5]),print(nomask_files[0:5])
```

```
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]

(None, None)
```

```
print(len(mask_files)),print(len(nomask_files))
```

```
3725
3828

(None, None)
```

```
labels = mask_files + nomask_files
print(len(labels))
print(labels[0:5])
print(labels[-5:])
```

```
7553
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

```
# display with mask image
img =
mpimg.imread('/kaggle/input/face-mask-dataset/data/with_mask/with_mask
_1545.jpg')
imgplot = plt.imshow(img)
plt.show()
```


```
# displaying without mask image
img =
mpimg.imread('/kaggle/input/face-mask-dataset/data/without_mask/without_mask_3215.jpg')
imgplot = plt.imshow(img)
plt.show()
```

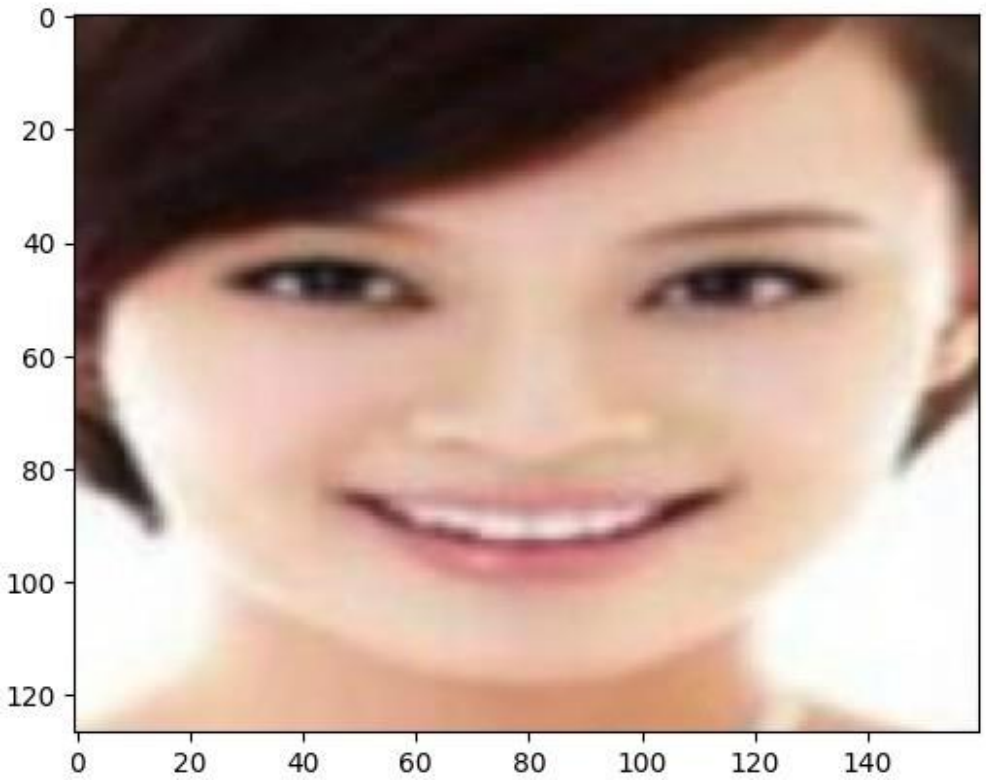


Image Processing

```
with_mask_path = '/kaggle/input/face-mask-dataset/data/with_mask/'
mask_files = os.listdir(with_mask_path)

data = []

for img_file in mask_files:

  image = Image.open(with_mask_path + str(img_file))
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)



without_mask_path =
'/kaggle/input/face-mask-dataset/data/without_mask/'
nomask_files = os.listdir(without_mask_path)

for img_file in nomask_files:

  image = Image.open(without_mask_path + str(img_file))
```

```
  image = image.resize((128,128))
  image = image.convert('RGB')
  image = np.array(image)
  data.append(image)
```

```
/opt/conda/lib/python3.10/site-packages/PIL/Image.py:992: UserWarning:
Palette images with Transparency expressed in bytes should be
converted to RGBA images
  warnings.warn(
```

```
type(data),len(data)
```

```
(list,  7553)
```

```
data[0]
```

```
array([[[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [254, 254, 254],
        [253, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [252, 253, 254],
        [246, 139, 136],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       ...,

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [212, 175, 154],
        [218, 187, 172],
        [250, 247, 245]],

       [[255, 255, 255],
```

```
         [255, 255, 255],
         [255, 255, 255],
         ...,
         [212, 175, 154],
         [220, 191, 175],
         [252, 250, 248]],

        [[255, 255, 255],
         [255, 255, 255],
         [255, 255, 255],
         ...,
         [211, 174, 155],
         [219, 194, 180],
         [252, 251, 250]]], dtype=uint8)
```

```
type(data[0]),data[0].shape
```

```
(numpy.ndarray, (128, 128, 3))
```

```
# converting image list and label list to numpy arrays

x = np.array(data)
y = np.array(labels)
```

Training

Train Test Split

```
x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.2,random_state=42)
```

```
print(x.shape, x_train.shape, x_test.shape)
```

```
(7553, 128, 128, 3) (6042, 128, 128, 3) (1511, 128, 128, 3)
```

```
# scaling the data

x_train_scaled = x_train/255
x_test_scaled = x_test/255
```

```
x_train[0],x_train_scaled[0]
```

```
(array([[[ 28,  27,  23],
         [ 28,  27,  23],
         [ 28,  27,  23],
         ...,
         [ 43,  43,  35],
```

```
         [ 43,  43,  35],
         [ 43,  43,  35]],

        [[ 30,  29,  25],
         [ 30,  29,  25],
         [ 30,  29,  25],
         ...,
         [ 43,  43,  35],
         [ 43,  43,  35],
         [ 43,  43,  35]],

        [[ 29,  28,  24],
         [ 29,  28,  24],
         [ 29,  28,  24],
         ...,
         [ 44,  44,  36],
         [ 44,  44,  36],
         [ 44,  44,  36]],

        ...,

        [[ 73,  41,  14],
         [ 70,  40,  15],
         [ 61,  35,  16],
         ...,
         [194, 189, 185],
         [193, 188, 184],
         [192, 187, 183]],

        [[ 79,  51,  27],
         [ 72,  44,  21],
         [ 61,  35,  17],
         ...,
         [193, 189, 185],
         [192, 188, 184],
         [192, 187, 183]],

        [[107,  91,  80],
         [ 87,  65,  51],
         [ 63,  39,  24],
         ...,
         [191, 188, 183],
         [190, 187, 182],
         [190, 187, 182]]], dtype=uint8),
 array([[[0.10980392, 0.10588235, 0.09019608],
         [0.10980392, 0.10588235, 0.09019608],
         [0.10980392, 0.10588235, 0.09019608],
         ...,
         [0.16862745, 0.16862745, 0.1372549 ],
         [0.16862745, 0.16862745, 0.1372549 ],
```

```
 [0.16862745, 0.16862745, 0.1372549 ]],

[[0.11764706, 0.11372549, 0.09803922],
 [0.11764706, 0.11372549, 0.09803922],
 [0.11764706, 0.11372549, 0.09803922],
 ...,
 [0.16862745, 0.16862745, 0.1372549 ],
 [0.16862745, 0.16862745, 0.1372549 ],
 [0.16862745, 0.16862745, 0.1372549 ]],

[[0.11372549, 0.10980392, 0.09411765],
 [0.11372549, 0.10980392, 0.09411765],
 [0.11372549, 0.10980392, 0.09411765],
 ...,
 [0.17254902, 0.17254902, 0.14117647],
 [0.17254902, 0.17254902, 0.14117647],
 [0.17254902, 0.17254902, 0.14117647]],

...,

[[0.28627451, 0.16078431, 0.05490196],
 [0.2745098 , 0.15686275, 0.05882353],
 [0.23921569, 0.1372549 , 0.0627451 ],
 ...,
 [0.76078431, 0.74117647, 0.7254902 ],
 [0.75686275, 0.7372549 , 0.72156863],
 [0.75294118, 0.73333333, 0.71764706]],

[[0.30980392, 0.2       , 0.10588235],
 [0.28235294, 0.17254902, 0.08235294],
 [0.23921569, 0.1372549 , 0.06666667],
 ...,
 [0.75686275, 0.74117647, 0.7254902 ],
 [0.75294118, 0.7372549 , 0.72156863],
 [0.75294118, 0.73333333, 0.71764706]],

[[0.41960784, 0.35686275, 0.31372549],
 [0.34117647, 0.25490196, 0.2       ],
 [0.24705882, 0.15294118, 0.09411765],
 ...,
 [0.74901961, 0.7372549 , 0.71764706],
 [0.74509804, 0.73333333, 0.71372549],
 [0.74509804, 0.73333333, 0.71372549]]]))
```

Building a Convolutional Neural Networks (CNN)

```
import tensorflow as tf
from tensorflow import keras
```

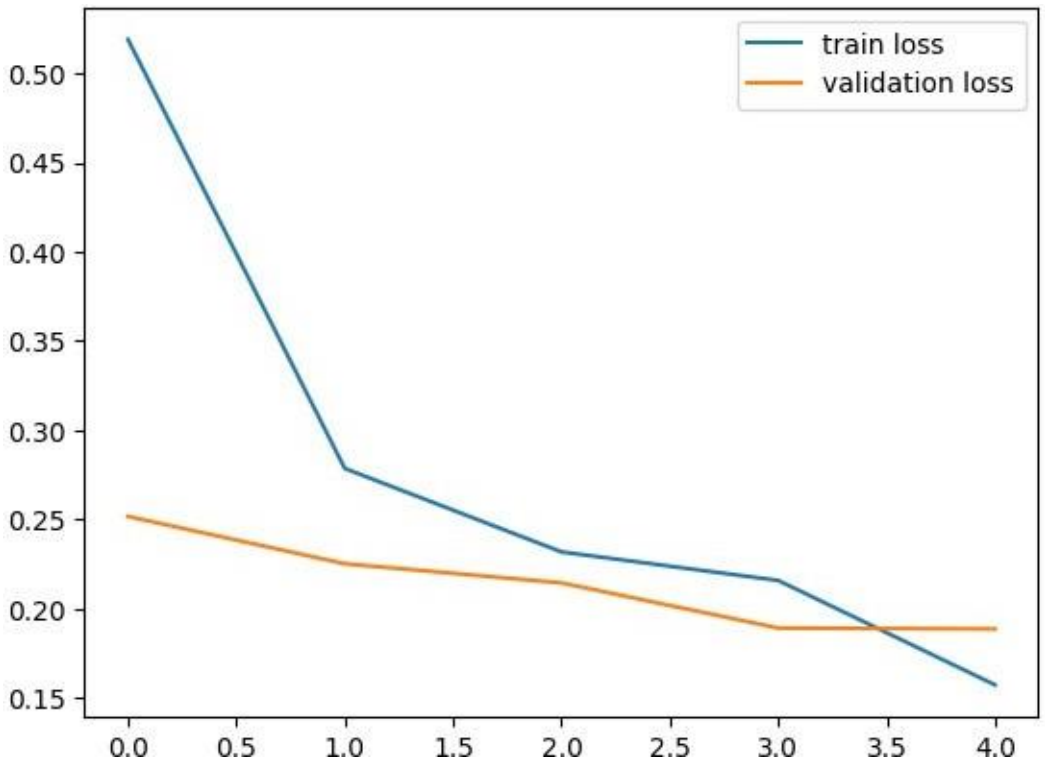
```
num_of_classes = 2

model = keras.Sequential()

model.add(keras.layers.Conv2D(32, kernel_size=(3,3),
activation='relu', input_shape=(128,128,3)))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))


model.add(keras.layers.Conv2D(64, kernel_size=(3,3),
activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))

model.add(keras.layers.Flatten())

model.add(keras.layers.Dense(128, activation='relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dropout(0.5))


model.add(keras.layers.Dense(num_of_classes, activation='sigmoid'))

# compile the neural network
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['acc'])

# training the neural network
history = model.fit(x_train_scaled, y_train, validation_split=0.1,
epochs=5)

Epoch 1/5
170/170 [==============================] - 74s 428ms/step - loss:
0.5190 - acc: 0.7835 - val_loss: 0.2517 - val_acc: 0.9025
Epoch 2/5
170/170 [==============================] - 72s 425ms/step - loss:
0.2786 - acc: 0.8948 - val_loss: 0.2252 - val_acc: 0.9058
Epoch 3/5
170/170 [==============================] - 72s 421ms/step - loss:
0.2318 - acc: 0.9088 - val_loss: 0.2145 - val_acc: 0.9306
Epoch 4/5
170/170 [==============================] - 71s 419ms/step - loss:
0.2158 - acc: 0.9183 - val_loss: 0.1892 - val_acc: 0.9240
Epoch 5/5
170/170 [==============================] - 71s 420ms/step - loss:
0.1573 - acc: 0.9404 - val_loss: 0.1886 - val_acc: 0.9273
```

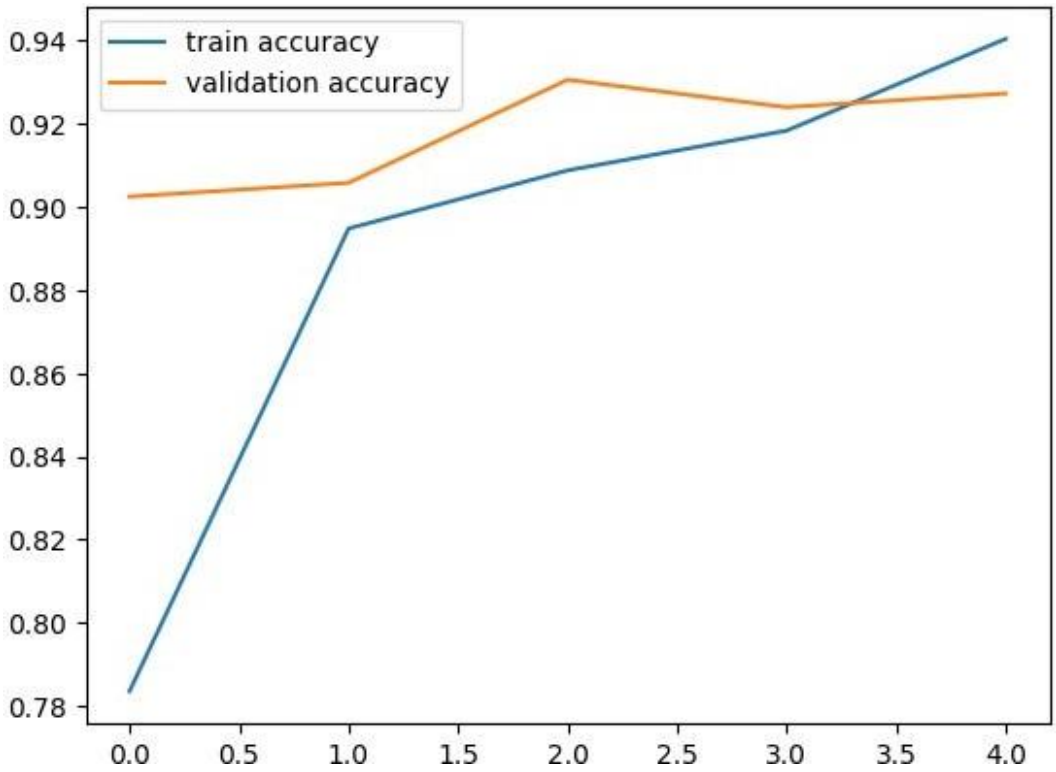
Model Evaluation

```
loss, accuracy = model.evaluate(x_test_scaled, y_test)
print('Test Accuracy =', accuracy)
```

```
48/48 [==============================] - 5s 102ms/step - loss: 0.1942
- acc: 0.9298
Test Accuracy = 0.929847776889801
```

```
h = history

# plot the loss value
plt.plot(h.history['loss'], label='train loss')
plt.plot(h.history['val_loss'], label='validation loss')
plt.legend()
plt.show()

# plot the accuracy value
plt.plot(h.history['acc'], label='train accuracy')
plt.plot(h.history['val_acc'], label='validation accuracy')
plt.legend()
plt.show()
```

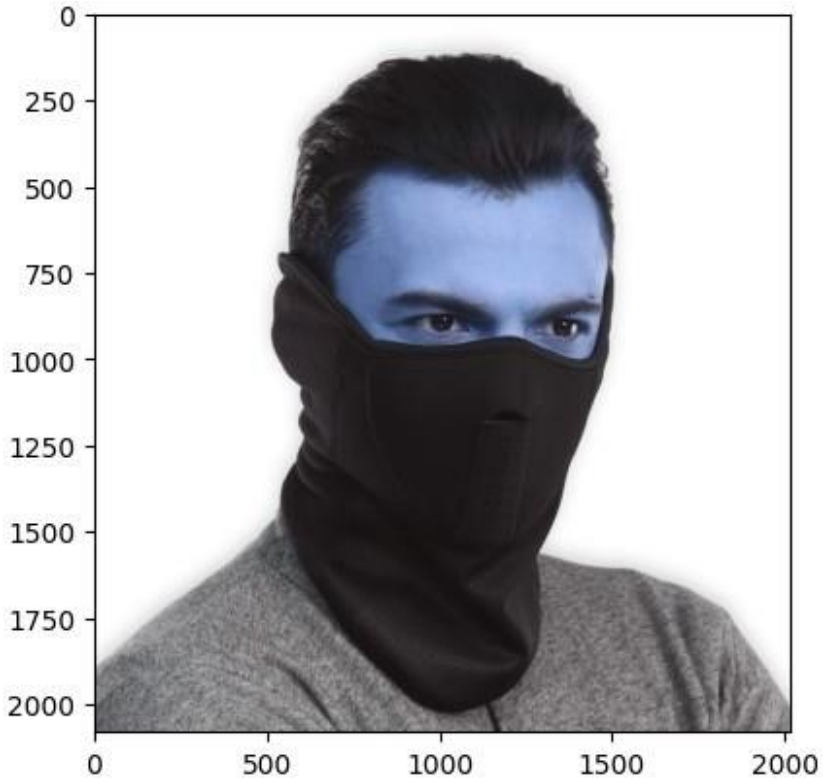




Predictive System

```
# input_image_path = input('Path of the image to be predicted: ')
input_image_path =
"/kaggle/input/face-mask-dataset/data/with_mask/with_mask_3204.jpg"
input_image = cv2.imread(input_image_path)

imgplot = plt.imshow(input_image)
plt.show()
input_image_resized = cv2.resize(input_image, (128,128))

input_image_scaled = input_image_resized/255

input_image_reshaped = np.reshape(input_image_scaled, [1,128,128,3])

input_prediction = model.predict(input_image_reshaped)

print(input_prediction)


input_pred_label = np.argmax(input_prediction)

print(input_pred_label)
```

```python
if input_pred_label == 1:

  print('The person in the image is wearing a mask')

else:

  print('The person in the image is not wearing a mask')
```




```
1/1 [==============================] - 0s 125ms/step
[[0.18832941 0.76705205]]
1
The person in the image is wearing a mask
```