```
# configuring the path of Kaggle.json file
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
!chmod 600 ~/.kaggle/kaggle.json
```

Importing Face Mask Dataset

```
# API to fetch the dataset from Kaggle
!kaggle datasets download -d omkargurav/face-mask-dataset
```

```
Downloading face-mask-dataset.zip to /content
100% 163M/163M [00:09<00:00, 22.1MB/s]
100% 163M/163M [00:09<00:00, 18.9MB/s]
```

```
# extracting the compessed Dataset
from zipfile import ZipFile
dataset = '/content/face-mask-dataset.zip'

with ZipFile(dataset,'r') as zip:
  zip.extractall()
  print('The dataset is extracted')
```

```
The dataset is extracted
```

```
!ls
```

```
data  face-mask-dataset.zip  kaggle.json  sample_data
```

Importing the Dependencies

```
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import cv2
from google.colab.patches import cv2_imshow
from PIL import Image
from sklearn.model_selection import train_test_split
```

```
with_mask_files = os.listdir('/content/data/with_mask')
print(with_mask_files[0:5])
print(with_mask_files[-5:])
```

```
['with_mask_193.jpg', 'with_mask_754.jpg', 'with_mask_486.jpg', 'with_mask_2756.jpg', 'with_mask_1328.jpg']
['with_mask_2590.jpg', 'with_mask_1545.jpg', 'with_mask_3357.jpg', 'with_mask_1143.jpg', 'with_mask_2196.jpg']
```

```
without_mask_files = os.listdir('/content/data/without_mask')
print(without_mask_files[0:5])
print(without_mask_files[-5:])
```

```
['without_mask_1871.jpg', 'without_mask_1012.jpg', 'without_mask_2600.jpg', 'without_mask_1623.jpg', 'without_mask_1116.jpg']
['without_mask_2925.jpg', 'without_mask_3559.jpg', 'without_mask_38.jpg', 'without_mask_1333.jpg', 'without_mask_1137.jpg']
```

```
print('Number of with mask images:', len(with_mask_files))
print('Number of without mask images:', len(without_mask_files))
```

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

Creating Labels for the two class of Images

with mask --> 1

without mask --> 0

```
# create the labels

with_mask_labels = [1]*3725

without_mask_labels = [0]*3828


print(with_mask_labels[0:5])

print(without_mask_labels[0:5])
```

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

```
print(len(with_mask_labels))
print(len(without_mask_labels))
```

```
3725
3828
```

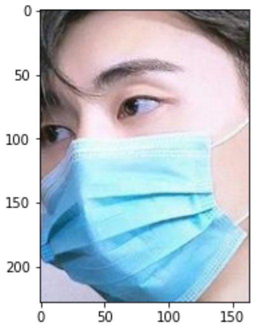
```
labels = with_mask_labels + without_mask_labels

print(len(labels))
print(labels[0:5])
print(labels[-5:])
```

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

Displaying the Images

```
# displaying with mask image
img = mpimg.imread('/content/data/with_mask/with_mask_1545.jpg')
imgplot = plt.imshow(img)
plt.show()
```



```
# displaying without mask image
img = mpimg.imread('/content/data/without_mask/without_mask_2925.jpg')
imgplot = plt.imshow(img)
plt.show()
```

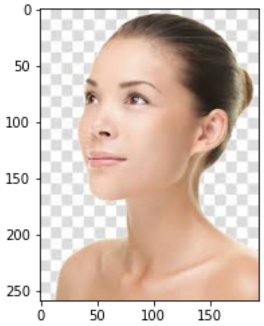


Image Processing

1. Resize the Images
2. Convert the images to numpy arrays

```
# convert images to numpy arrays+

with_mask_path = '/content/data/with_mask/'

data = []

for img_file in with_mask_files:

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



without_mask_path = '/content/data/without_mask/'


for img_file in without_mask_files:

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

```
/usr/local/lib/python3.8/dist-packages/PIL/Image.py:959: UserWarning: Palette images with Transparency expressed in bytes should be conv
  warnings.warn(
```

```
type(data)
```

```
list
```

```
len(data)
```

```
7553
```

```
data[0]
```

```
array([[[166, 115,  98],
        [159, 110,  95],
        [156, 111,  99],
        ...,
        [142, 100,  85],
        [145, 105,  90],
        [150, 113,  97]],

       [[164, 113,  97],
        [157, 109,  93],
        [153, 107,  96],
        ...,
        [138,  95,  80],
        [144, 103,  88],
        [147, 108,  93]],

       [[160, 111,  96],
        [155, 109,  95],
        [150, 108,  95],
        ...,
        [138,  95,  80],
        [147, 106,  92],
        [140, 102,  87]],

       ...,

       [[192, 145, 134],
        [193, 148, 137],
        [194, 149, 139],
        ...,
        [100,  74,  77],
        [102,  76,  79],
        [101,  75,  76]],

       [[190, 146, 131],
        [192, 149, 134],
        [195, 151, 137],
        ...,
        [ 95,  70,  73],
        [101,  75,  77],
        [102,  76,  77]],

       [[188, 145, 128],
        [191, 148, 132],
        [193, 148, 134],
        ...,
        [ 95,  70,  73],
        [ 94,  67,  70],
        [ 96,  70,  71]]], dtype=uint8)
```

```
type(data[0])
```

```
numpy.ndarray
```

```
data[0].shape
```

```
(128, 128, 3)
```

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

X = np.array(data)
Y = np.array(labels)
```

```
type(X)
```

```
numpy.ndarray
```

```
type(Y)
```

```
numpy.ndarray
```

```
print(X.shape)
print(Y.shape)
```

```
(7553, 128, 128, 3)
(7553,)
```

```
print(Y)
```

```
[1 1 1 ... 0 0 0]
```

Train Test Split

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
```

```
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]
```

```
array([[[109, 107, 118],
        [114, 113, 121],
        [109, 107, 116],
        ...,
        [ 90,  97, 107],
        [ 90,  94, 105],
        [ 93,  97, 108]],

       [[110, 108, 119],
        [111, 108, 117],
        [110, 105, 114],
        ...,
        [ 86,  93, 103],
        [ 88,  92, 103],
        [ 89,  93, 104]],

       [[112, 107, 118],
        [113, 109, 118],
        [123, 117, 125],
        ...,
        [ 89,  95, 105],
        [ 91,  95, 106],
        [ 87,  91, 102]],

       ...,

       [[ 46,  66,  91],
        [ 45,  65,  90],
        [ 47,  67,  92],
        ...,
        [177, 143, 123],
        [176, 144, 123],
        [177, 145, 124]],

       [[ 49,  69,  93],
        [ 47,  67,  91],
        [ 46,  66,  90],
        ...,
        [179, 146, 126],
        [178, 146, 125],
        [177, 146, 125]],

       [[ 43,  63,  87],
        [ 43,  63,  87],
        [ 44,  64,  88],
        ...,
        [179, 147, 126],
        [177, 145, 124],
        [175, 144, 123]]], dtype=uint8)
```

```
X_train_scaled[0]
```

```
array([[[0.42745098, 0.41960784, 0.4627451 ],
        [0.44705882, 0.44313725, 0.4745098 ],
        [0.42745098, 0.41960784, 0.45490196],
        ...,
        [0.35294118, 0.38039216, 0.41960784],
        [0.35294118, 0.36862745, 0.41176471],
        [0.36470588, 0.38039216, 0.42352941]],

       [[0.43137255, 0.42352941, 0.46666667],
        [0.43529412, 0.42352941, 0.45882353],
        [0.43137255, 0.41176471, 0.44705882],
        ...,
        [0.3372549 , 0.36470588, 0.40392157],
        [0.34509804, 0.36078431, 0.40392157],
        [0.34901961, 0.36470588, 0.40784314]],

       [[0.43921569, 0.41960784, 0.4627451 ],
        [0.44313725, 0.42745098, 0.4627451 ],
        [0.48235294, 0.45882353, 0.49019608],
        ...,
        [0.34901961, 0.37254902, 0.41176471],
        [0.35686275, 0.37254902, 0.41568627],
        [0.34117647, 0.35686275, 0.4       ]],

       ...,

       [[0.18039216, 0.25882353, 0.35686275],
        [0.17647059, 0.25490196, 0.35294118],
        [0.18431373, 0.2627451 , 0.36078431],
        ...,
        [0.69411765, 0.56078431, 0.48235294],
        [0.69019608, 0.56470588, 0.48235294],
        [0.69411765, 0.56862745, 0.48627451]],

       [[0.19215686, 0.27058824, 0.36470588],
        [0.18431373, 0.2627451 , 0.35686275],
        [0.18039216, 0.25882353, 0.35294118],
        ...,
        [0.70196078, 0.57254902, 0.49411765],
        [0.69803922, 0.57254902, 0.49019608],
        [0.69411765, 0.57254902, 0.49019608]],

       [[0.16862745, 0.24705882, 0.34117647],
        [0.16862745, 0.24705882, 0.34117647],
        [0.17254902, 0.25098039, 0.34509804],
        ...,
        [0.70196078, 0.57647059, 0.49411765],
        [0.69411765, 0.56862745, 0.48627451],
        [0.68627451, 0.56470588, 0.48235294]]]])
```

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 [==============================] - 15s 24ms/step - loss: 0.4886 - acc: 0.7848 - val_loss: 0.3200 - val_acc: 0.8711
Epoch 2/5
170/170 [==============================] - 3s 17ms/step - loss: 0.2937 - acc: 0.8847 - val_loss: 0.2501 - val_acc: 0.9008
Epoch 3/5
170/170 [==============================] - 3s 17ms/step - loss: 0.2523 - acc: 0.9016 - val_loss: 0.2516 - val_acc: 0.8992
Epoch 4/5
170/170 [==============================] - 3s 19ms/step - loss: 0.1970 - acc: 0.9270 - val_loss: 0.2292 - val_acc: 0.9256
Epoch 5/5
170/170 [==============================] - 3s 17ms/step - loss: 0.1810 - acc: 0.9308 - val_loss: 0.2427 - val_acc: 0.9074
```

Model Evaluation

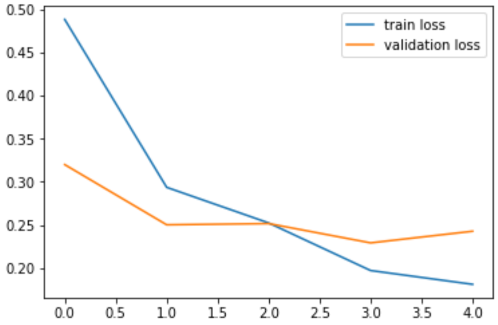
```
loss, accuracy = model.evaluate(X_test_scaled, Y_test)
print('Test Accuracy =', accuracy)
```

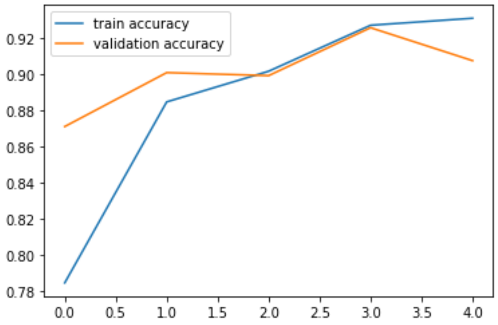
```
48/48 [==============================] - 1s 11ms/step - loss: 0.2065 - acc: 0.9219
Test Accuracy = 0.9219059944152832
```

```
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 = cv2.imread(input_image_path)

cv2_imshow(input_image)

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)
```

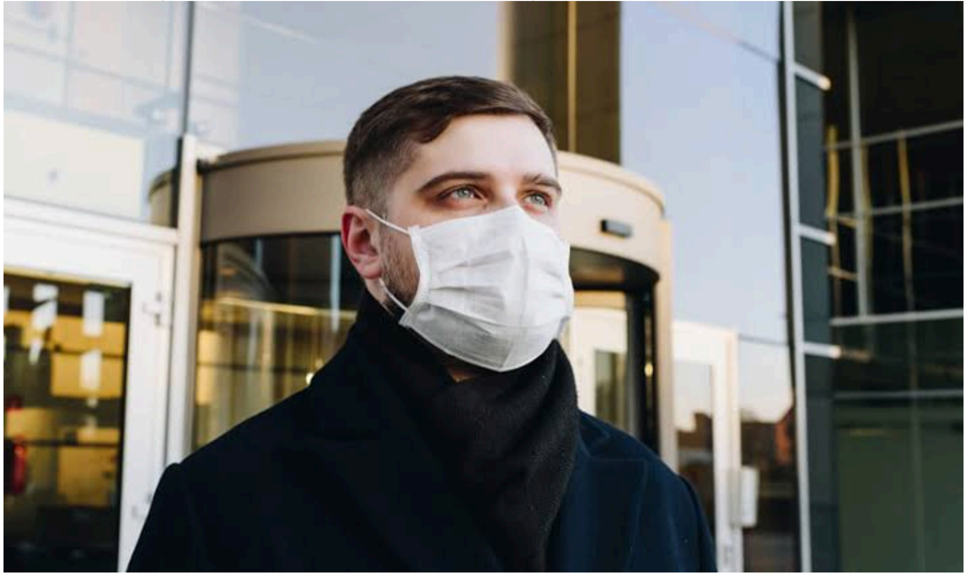
```
if input_pred_label == 1:

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

else:

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

```
Path of the image to be predicted: /content/test.png
```



```
1/1 [==============================] - 0s 176ms/step
[[0.23994292 0.70647454]]
1
The person in the image is wearing a mask
```

```
input_image_path = input('Path of the image to be predicted: ')

input_image = cv2.imread(input_image_path)

cv2_imshow(input_image)

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)


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')
```

```
Path of the image to be predicted: /content/test.jpg
```

```
1/1 [==============================] - 0s 21ms/step
[[0.49811754 0.47740024]]
0
The person in the image is not wearing a mask
```

Start coding or generate with AI.