```
!pip install opencv-python
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packages (4.7.0.72)
Requirement already satisfied: numpy>=1.19.3 in /usr/local/lib/python3.10/dist-packages (from opencv-python) (1.2
```

```
import cv2
```

```
import zipfile
with zipfile.ZipFile('/content/finalproject.zip', 'r') as zip_ref:
    zip_ref.extractall('/content/')
```

```
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/Final Project/Training Data/with_mask')
print(with_mask_files[0:5])
print(with_mask_files[-5:])
```

```
['with_mask_279.jpg', 'with_mask_162.jpg', 'with_mask_259.jpg', 'with_mask_267.jpg', 'with_mask_28.jpg']
['with_mask_390.jpg', 'with_mask_29.jpg', 'with_mask_90.jpg', 'with_mask_361.jpg', 'with_mask_289.jpg']
```

```
without_mask_files = os.listdir('/content/Final Project/Training Data/without_mask')
print(without_mask_files[0:5])
print(without_mask_files[-5:])
```

```
['without_mask_17.jpg', 'without_mask_491.jpg', 'without_mask_96.jpg', 'without_mask_103.jpg', 'without_mask_311.
['without_mask_319.jpg', 'without_mask_142.jpg', 'without_mask_481.jpg', 'without_mask_43.jpg', 'without_mask_230
```

```
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: 500
Number of without mask images: 500
```

```
# create the labels

with_mask_labels = [1]*500

without_mask_labels = [0]*500
```

```
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))
```

```
500
500
```

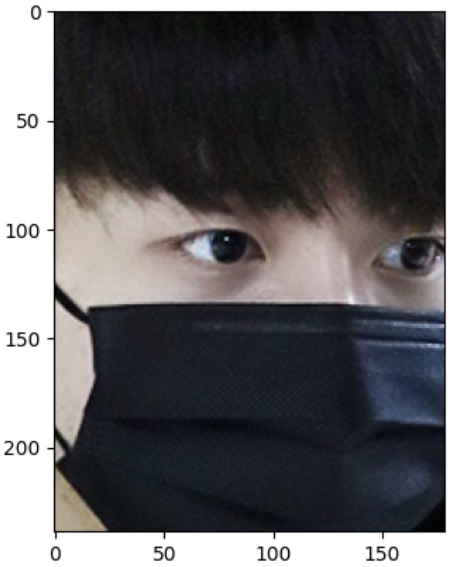
```
labels = with_mask_labels + without_mask_labels

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

```
1000
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
```

```
# display with mask image
```

```
# displaying with mask image
img = mpimg.imread('/content/Final Project/Training Data/with_mask/with_mask_390.jpg')
imgplot = plt.imshow(img)
plt.show()
```




```
# displaying without mask image
img = mpimg.imread('/content/Final Project/Training Data/without_mask/without_mask_319.jpg')
imgplot = plt.imshow(img)
plt.show()
```


```
# convert images to numpy arrays+

with_mask_path = '/content/Final Project/Training 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/Final Project/Training 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)
```

```
type(data)
```

```
    list
```

```
len(data)
```

```
    1000
```

```
data[0]
```

```
    array([[[ 60,  46,  45],
            [ 62,  48,  47],
            [ 65,  51,  50],
            ...,
            [ 54,  40,  37],
            [ 57,  43,  40],
            [ 62,  48,  45]],

           [[ 65,  51,  50],
            [ 65,  51,  50],
            [ 63,  49,  48],
            ...,
            [ 52,  38,  35],
            [ 49,  35,  32],
            [ 50,  36,  33]],

           [[ 68,  54,  53],
            [ 64,  50,  49],
            [ 56,  42,  41],
            ...,
            [ 60,  46,  44],
            [ 54,  40,  38],
            [ 49,  35,  33]],

           ...,

           [[185, 192, 198],
            [185, 194, 199],
            [208, 218, 225],
            ...,
            [192, 164, 152],
            [191, 161, 150],
            [192, 160, 149]],

           [[218, 225, 231],
            [173, 180, 186],
            [163, 172, 178],
            ...,
            [192, 164, 152],
            [192, 162, 151],
            [193, 161, 150]],

           [[232, 239, 245],
            [221, 228, 234],
            [192, 201, 207],
            ...,
            [193, 165, 153],
            [193, 163, 152],
            [193, 162, 151]]], 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)
```

```
(1000, 128, 128, 3)
(1000,)
```

```
print(Y)
```

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

```
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)
```

```
(1000, 128, 128, 3) (800, 128, 128, 3) (200, 128, 128, 3)
```

```
# scaling the data

X_train_scaled = X_train/255

X_test_scaled = X_test/255
```

```
X_train[0]
```

```
array([[[ 82,  73,  58],
        [ 82,  73,  58],
        [ 82,  73,  58],
        ...,
        [ 99,  86,  69],
        [103,  90,  71],
        [103,  90,  71]],

       [[ 81,  73,  58],
        [ 79,  70,  55],
        [ 79,  70,  55],
        ...,
        [100,  87,  70],
        [103,  90,  73],
        [102,  89,  71]],

       [[ 81,  72,  57],
        [ 79,  70,  55],
        [ 79,  70,  55],
```

```
           ...,
           [104,  89,  73],
           [103,  88,  72],
           [103,  88,  72]],

          ...,

          [[ 35,  21,  14],
           [ 34,  19,  12],
           [ 36,  21,  13],
           ...,
           [ 24,  10,   3],
           [ 41,  28,  20],
           [ 45,  34,  24]],

          [[ 34,  20,  13],
           [ 32,  16,   9],
           [ 38,  20,  13],
           ...,
           [ 30,  18,  10],
           [ 39,  28,  18],
           [ 42,  32,  22]],

          [[ 34,  19,  12],
           [ 33,  16,   9],
           [ 35,  17,  10],
           ...,
           [ 33,  23,  13],
           [ 41,  31,  21],
           [ 47,  37,  27]]], dtype=uint8)
```

```
X_train_scaled[0]
```

```
    array([[[0.32156863, 0.28627451, 0.22745098],
            [0.32156863, 0.28627451, 0.22745098],
            [0.32156863, 0.28627451, 0.22745098],
            ...,
            [0.38823529, 0.3372549 , 0.27058824],
            [0.40392157, 0.35294118, 0.27843137],
            [0.40392157, 0.35294118, 0.27843137]],

           [[0.31764706, 0.28627451, 0.22745098],
            [0.30980392, 0.2745098 , 0.21568627],
            [0.30980392, 0.2745098 , 0.21568627],
            ...,
            [0.39215686, 0.34117647, 0.2745098 ],
            [0.40392157, 0.35294118, 0.28627451],
            [0.4       , 0.34901961, 0.27843137]],

           [[0.31764706, 0.28235294, 0.22352941],
            [0.30980392, 0.2745098 , 0.21568627],
            [0.30980392, 0.2745098 , 0.21568627],
            ...,
            [0.40784314, 0.34901961, 0.28627451],
            [0.40392157, 0.34509804, 0.28235294],
            [0.40392157, 0.34509804, 0.28235294]],

           ...,

           [[0.1372549 , 0.08235294, 0.05490196],
            [0.13333333, 0.0745098 , 0.04705882],
            [0.14117647, 0.08235294, 0.05098039],
            ...,
            [0.09411765, 0.03921569, 0.01176471],
            [0.16078431, 0.10980392, 0.07843137],
            [0.17647059, 0.13333333, 0.09411765]],

           [[0.13333333, 0.07843137, 0.05098039],
            [0.1254902 , 0.0627451 , 0.03529412],
            [0.14901961, 0.07843137, 0.05098039],
            ...,
            [0.11764706, 0.07058824, 0.03921569],
            [0.15294118, 0.10980392, 0.07058824],
            [0.16470588, 0.1254902 , 0.08627451]],

           [[0.13333333, 0.0745098 , 0.04705882],
            [0.12941176, 0.0627451 , 0.03529412],
            [0.1372549 , 0.06666667, 0.03921569],
            ...,
            [0.12941176, 0.09019608, 0.05098039],
            [0.16078431, 0.12156863, 0.08235294],
            [0.18431373, 0.14509804, 0.10588235]]])
```

```
#building 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=10)
```

```
Epoch 1/10
23/23 [==============================] - 24s 911ms/step - loss: 0.7523 - acc: 0.6917 - val_loss: 0.2820 - val_acc
Epoch 2/10
23/23 [==============================] - 22s 951ms/step - loss: 0.3294 - acc: 0.8958 - val_loss: 0.1151 - val_acc
Epoch 3/10
23/23 [==============================] - 20s 889ms/step - loss: 0.2124 - acc: 0.9347 - val_loss: 0.1231 - val_acc
Epoch 4/10
23/23 [==============================] - 21s 939ms/step - loss: 0.1236 - acc: 0.9528 - val_loss: 0.1678 - val_acc
Epoch 5/10
23/23 [==============================] - 20s 878ms/step - loss: 0.1455 - acc: 0.9500 - val_loss: 0.0796 - val_acc
Epoch 6/10
23/23 [==============================] - 22s 918ms/step - loss: 0.1081 - acc: 0.9708 - val_loss: 0.1011 - val_acc
Epoch 7/10
23/23 [==============================] - 22s 943ms/step - loss: 0.0717 - acc: 0.9792 - val_loss: 0.1059 - val_acc
Epoch 8/10
23/23 [==============================] - 21s 900ms/step - loss: 0.0745 - acc: 0.9778 - val_loss: 0.0894 - val_acc
Epoch 9/10
23/23 [==============================] - 22s 952ms/step - loss: 0.0848 - acc: 0.9792 - val_loss: 0.0803 - val_acc
Epoch 10/10
23/23 [==============================] - 20s 870ms/step - loss: 0.0555 - acc: 0.9861 - val_loss: 0.1201 - val_acc
```

Calculating Model Accuracy

```
#Model evaluation


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

```
7/7 [==============================] - 1s 197ms/step - loss: 0.0271 - acc: 0.9850
Test Accuracy = 0.9850000143051147
```

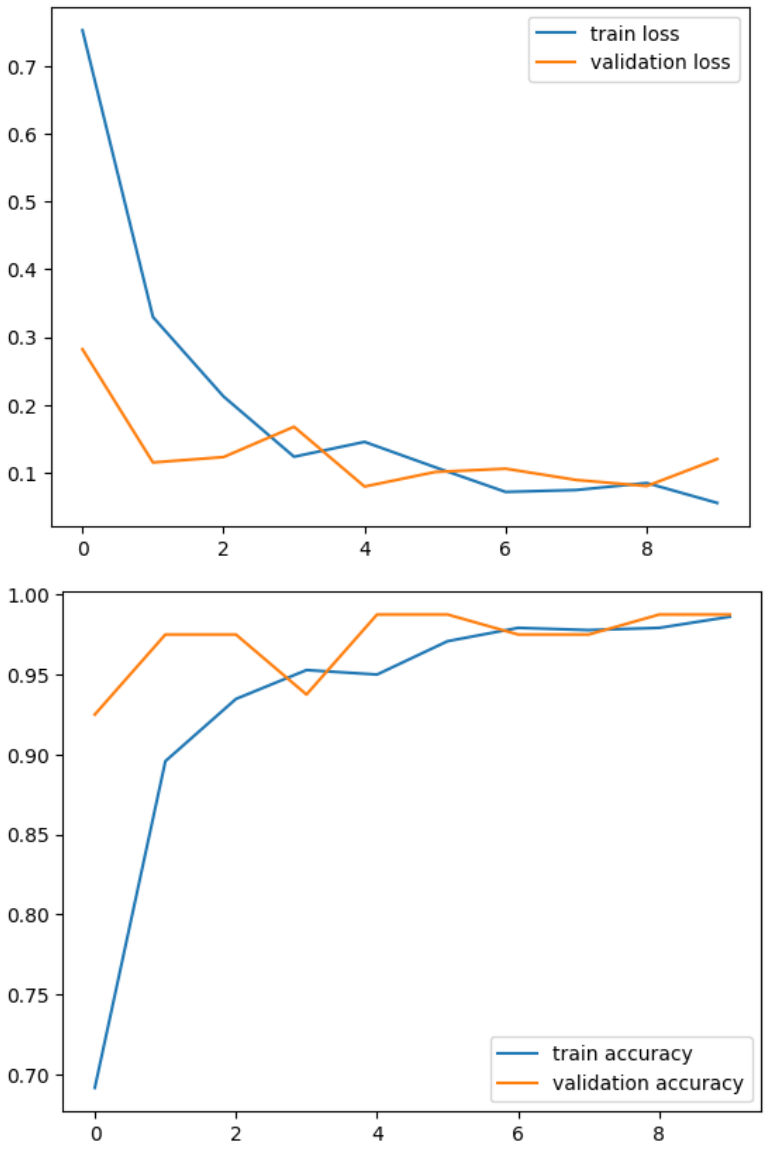
Plotting the graph

```
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()
```



Predicting with 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(type(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/anshumask.jpeg
```

```
1/1 [==============================] - 0s 48ms/step
[[0.19748174 0.9925925 ]]
<class 'numpy.int64'>
The person in the image is wearing a mask
```

Predicting No **Mask**

```
#code begins from here


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(type(input_pred_label))
if input_pred_label == 0:

  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/WhatsApp Image 2023-05-09 at 11.12.05 PM (1).jpg
```

```
1/1 [==============================] - 0s 50ms/step
[[0.3353671 0.8706998]]
<class 'numpy.int64'>
The person in the image is not wearing a mask
```