```
!pip install kaggle
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: kaggle in /usr/local/lib/python3.9/dist-packages (1.5.13)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.9/dist-packages (from kaggle) (8.0.1)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.9/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.9/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-packages (from kaggle) (2.27.1)
Requirement already satisfied: certifi in /usr/local/lib/python3.9/dist-packages (from kaggle) (2022.12.7)
Requirement already satisfied: tqdm in /usr/local/lib/python3.9/dist-packages (from kaggle) (4.65.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.9/dist-packages (from kaggle) (1.26.15)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.9/dist-packages (from python-slugify->kaggle) (1.3)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests->kaggle) (3.4)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.9/dist-packages (from requests->kaggle) (2.0.12)
```

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

```
!kaggle datasets download -d omkargurav/face-mask-dataset
```

```
face-mask-dataset.zip: Skipping, found more recently modified local copy (use --force to force download)
```

```
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  without_mask.jpeg
```

importing the necessary 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
```

importing with mask and without mask files

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

```
['with_mask_206.jpg', 'with_mask_331.jpg', 'with_mask_1429.jpg', 'with_mask_3343.jpg', 'with_mask_3419.jpg']
['with_mask_2842.jpg', 'with_mask_705.jpg', 'with_mask_3352.jpg', 'with_mask_3102.jpg', 'with_mask_1473.jpg']
```

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

```
['without_mask_1171.jpg', 'without_mask_2819.jpg', 'without_mask_1583.jpg', 'without_mask_68.jpg', 'without_mask_3320.jpg']
['without_mask_1651.jpg', 'without_mask_2076.jpg', 'without_mask_3655.jpg', 'without_mask_2987.jpg', 'without_mask_3772.jpg']
```

To check the balance of datasets we have

```
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 label for the two classes of the images

with mask ---> 1 without mask --> 0

```
with_mask_labels = [1]*3725

without_mask_labels = [0]*3828

#checking
print(with_mask_labels[0:5])

print(without_mask_labels[0:5])

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

```
[1, 1, 1, 1, 1]
[0, 0, 0, 0, 0]
3725
3828
```

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

```
3725
3828
```

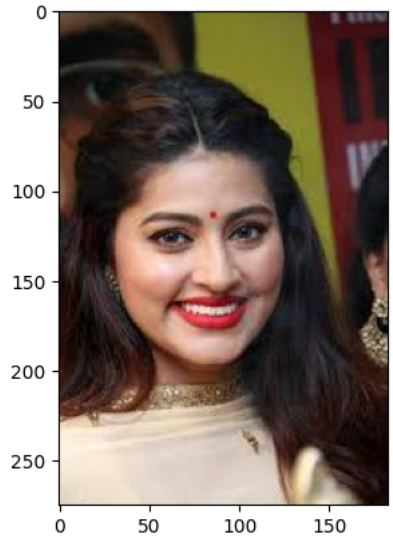
```
labels = with_mask_labels + without_mask_labels

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

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

Displaying with mask and without mask images

```
# displaying without mask image
img = mpimg.imread('/content/data/without_mask/without_mask_1915.jpg')
imgplot = plt.imshow(img)
plt.show()
```




```
# displaying with mask image
img = mpimg.imread('/content/data/with_mask/with_mask_1345.jpg')
imgplot = plt.imshow(img)
plt.show()
```

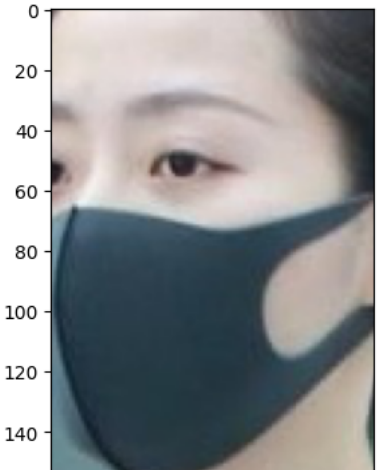


Image Processing and converting images into numpy arrays

```
# 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)


type(data)

     list


len(data)

     7553


data[0]

     array([[[ 23, 101, 139],
             [ 18,  96, 134],
             [ 13,  91, 129],
             ...,
             [ 10,  73, 104],
             [  6,  71, 101],
             [  4,  69,  99]],

            [[ 18,  97, 134],
             [ 13,  91, 129],
             [  7,  85, 123],
             ...,
             [  8,  71, 102],
             [  7,  72, 102],
             [  7,  72, 102]],

            [[ 18,  98, 135],
             [ 16,  95, 133],
             [ 12,  89, 129],
             ...,
             [ 11,  74, 105],
             [ 10,  75, 105],
             [ 10,  75, 105]],
```

```
           ...,

           [[125, 132, 153],
            [113, 119, 140],
            [ 99, 106, 126],
            ...,
            [148, 138, 155],
            [120, 109, 125],
            [119, 106, 123]],

           [[123, 130, 151],
            [115, 121, 143],
            [102, 110, 131],
            ...,
            [138, 129, 145],
            [115, 103, 120],
            [122, 109, 126]],

           [[123, 131, 153],
            [114, 122, 143],
            [104, 112, 133],
            ...,
            [130, 120, 137],
            [116, 104, 120],
            [129, 116, 133]]], dtype=uint8)
```

Double-click (or enter) to edit

```
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 train data

```
X_train_scaled = X_train/255

X_test_scaled = X_test/255
```

```
X_train[0]
```

```
array([[[224, 223, 219],
        [224, 223, 219],
        [226, 225, 221],
        ...,
        [249, 248, 246],
        [249, 248, 246],
        [249, 248, 246]],

       [[224, 223, 219],
        [225, 224, 220],
        [227, 226, 222],
        ...,
        [249, 248, 246],
        [249, 248, 246],
        [249, 248, 246]],

       [[225, 224, 220],
        [226, 225, 221],
        [228, 227, 223],
        ...,
        [249, 248, 246],
        [249, 248, 246],
        [249, 248, 246]],

       ...,

       [[ 11,  23,  65],
        [ 11,  24,  66],
        [  5,  19,  59],
        ...,
        [ 34,  53, 127],
        [ 37,  61, 134],
        [ 41,  66, 140]],

       [[  9,  21,  60],
        [ 11,  25,  62],
        [  5,  21,  56],
        ...,
        [ 30,  43, 116],
        [ 35,  53, 125],
        [ 38,  58, 133]],

       [[ 11,  20,  61],
        [ 14,  24,  65],
        [ 12,  23,  62],
        ...,
        [ 33,  46, 120],
        [ 32,  48, 122],
        [ 39,  57, 133]]], dtype=uint8)
```

```
X_train_scaled[0]
```

```
array([[[0.87843137, 0.8745098 , 0.85882353],
        [0.87843137, 0.8745098 , 0.85882353],
        [0.88627451, 0.88235294, 0.86666667],
        ...,
        [0.97647059, 0.97254902, 0.96470588],
        [0.97647059, 0.97254902, 0.96470588],
        [0.97647059, 0.97254902, 0.96470588]],

       [[0.87843137, 0.8745098 , 0.85882353],
        [0.88235294, 0.87843137, 0.8627451 ],
        [0.89019608, 0.88627451, 0.87058824],
        ...,
        [0.97647059, 0.97254902, 0.96470588],
        [0.97647059, 0.97254902, 0.96470588],
        [0.97647059, 0.97254902, 0.96470588]],

       [[0.88235294, 0.87843137, 0.8627451 ],
        [0.88627451, 0.88235294, 0.86666667],
        [0.89411765, 0.89019608, 0.8745098 ],
        ...,
        [0.97647059, 0.97254902, 0.96470588],
        [0.97647059, 0.97254902, 0.96470588],
        [0.97647059, 0.97254902, 0.96470588]],

       ...,

       [[0.04313725, 0.09019608, 0.25490196],
        [0.04313725, 0.09411765, 0.25882353],
        [0.01960784, 0.0745098 , 0.23137255],
        ...,
        [0.13333333, 0.20784314, 0.49803922],
        [0.14509804, 0.23921569, 0.5254902 ],
        [0.16078431, 0.25882353, 0.54901961]],
```

```
           [[0.03529412, 0.08235294, 0.23529412],
            [0.04313725, 0.09803922, 0.24313725],
            [0.01960784, 0.08235294, 0.21960784],
            ...,
            [0.11764706, 0.16862745, 0.45490196],
            [0.1372549 , 0.20784314, 0.49019608],
            [0.14901961, 0.22745098, 0.52156863]],

           [[0.04313725, 0.07843137, 0.23921569],
            [0.05490196, 0.09411765, 0.25490196],
            [0.04705882, 0.09019608, 0.24313725],
            ...,
            [0.12941176, 0.18039216, 0.47058824],
            [0.1254902 , 0.18823529, 0.47843137],
            [0.15294118, 0.22352941, 0.52156863]]])
```

Building CNN

```
#importing tensorflow and keras
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'))


model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['acc'])
```

Training the CNN

```
history = model.fit(X_train_scaled, Y_train, validation_split=0.1, epochs=5)
```

```
     Epoch 1/5
     170/170 [==============================] - 10s 22ms/step - loss: 0.5046 - acc: 0.8043 - val_loss: 0.2708 - val_acc: 0.8975
     Epoch 2/5
     170/170 [==============================] - 3s 17ms/step - loss: 0.2928 - acc: 0.8856 - val_loss: 0.2345 - val_acc: 0.8992
     Epoch 3/5
     170/170 [==============================] - 3s 18ms/step - loss: 0.2481 - acc: 0.8999 - val_loss: 0.1825 - val_acc: 0.9339
     Epoch 4/5
     170/170 [==============================] - 3s 17ms/step - loss: 0.2159 - acc: 0.9134 - val_loss: 0.1734 - val_acc: 0.9339
     Epoch 5/5
     170/170 [==============================] - 3s 17ms/step - loss: 0.1795 - acc: 0.9297 - val_loss: 0.1830 - val_acc: 0.9273
```

Modal Evaluation

```
loss, accuracy = model.evaluate(X_test_scaled, Y_test)
print('Accuracy of the test done =', accuracy)
```

```
     48/48 [==============================] - 0s 8ms/step - loss: 0.2249 - acc: 0.9120
     Accuracy of the test done = 0.9119788408279419
```

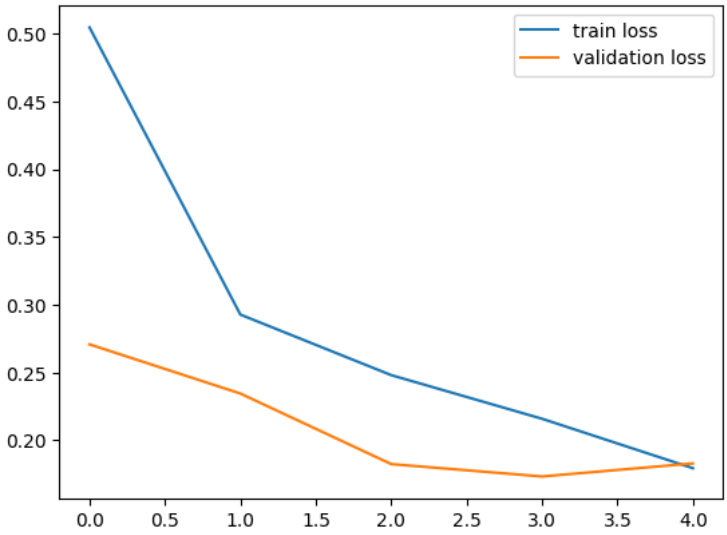
Plotting loss and accuracy value

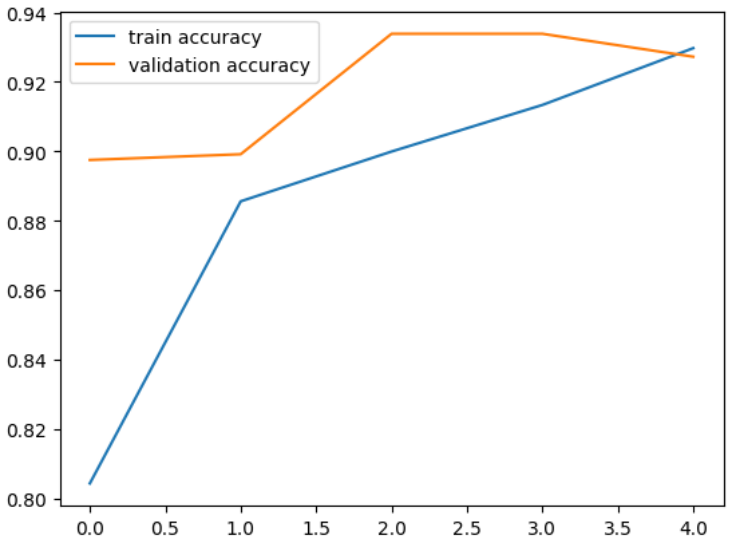
```
h = history


plt.plot(h.history['loss'], label='train loss')
plt.plot(h.history['val_loss'], label='validation loss')
```

```
plt.legend()
plt.show()


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





✓ 0s completed at 12:08