```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in /usr/local/lib/python3.11/dist-packages (1.7.4.2)
Requirement already satisfied: bleach in /usr/local/lib/python3.11/dist-packages (from kaggle) (6.2.0)
Requirement already satisfied: certifi>=14.05.14 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2025.4.26)
Requirement already satisfied: charset-normalizer in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.4.1)
Requirement already satisfied: idna in /usr/local/lib/python3.11/dist-packages (from kaggle) (3.10)
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-packages (from kaggle) (5.29.4)
Requirement already satisfied: python-dateutil>=2.5.3 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.9.0.post0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.11/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.32.3)
Requirement already satisfied: setuptools>=21.0.0 in /usr/local/lib/python3.11/dist-packages (from kaggle) (75.2.0)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.17.0)
Requirement already satisfied: text-unidecode in /usr/local/lib/python3.11/dist-packages (from kaggle) (1.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from kaggle) (4.67.1)
Requirement already satisfied: urllib3>=1.15.1 in /usr/local/lib/python3.11/dist-packages (from kaggle) (2.4.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.11/dist-packages (from kaggle) (0.5.1)
```

Start coding or generate with AI.

```
# 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
```

```
Dataset URL: https://www.kaggle.com/datasets/omkargurav/face-mask-dataset
License(s): unknown
```

```
# 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_3431.jpg', 'with_mask_3188.jpg', 'with_mask_970.jpg', 'with_mask_791.jpg', 'with_mask_1840.jpg']
['with_mask_281.jpg', 'with_mask_2278.jpg', 'with_mask_3301.jpg', 'with_mask_2285.jpg', 'with_mask_771.jpg']
```

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

```
['without_mask_1070.jpg', 'without_mask_3316.jpg', 'without_mask_3030.jpg', 'without_mask_3144.jpg', 'without_mask_1773.jpg']
['without_mask_1309.jpg', 'without_mask_416.jpg', 'without_mask_2195.jpg', 'without_mask_2158.jpg', 'without_mask_873.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_1546.jpg')
imgplot = plt.imshow(img)
plt.show()
```

```
# displaying without mask image
img = mpimg.imread('/content/data/without_mask/without_mask_2927.jpg')
imgplot = plt.imshow(img)
plt.show()
```

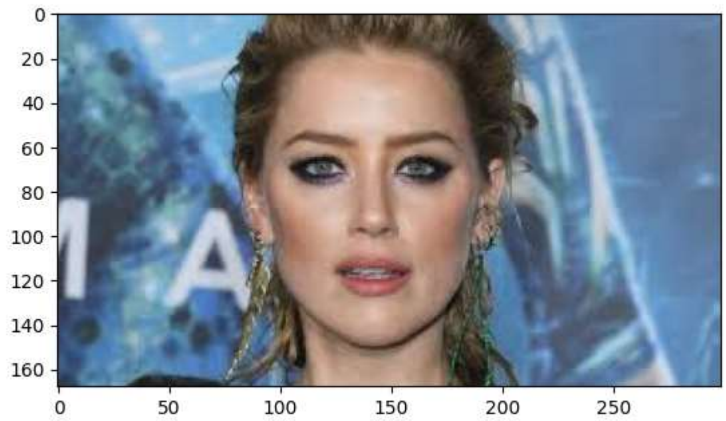


Image Processing

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/python3.11/dist-packages/PIL/Image.py:1043: UserWarning: Palette images with Transparency expressed in bytes should be cc
  warnings.warn(
```

```
type(data)
```

```
list
```

```
len(data)
```

```
7553
```

```
data[25]
```

```
ndarray (128, 128, 3) show data
```

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

```
numpy.ndarray
```

```
data[1].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[9]
```

```
ndarray (128, 128, 3) show data
```

```
X_train_scaled[0]
```

```
array([[[0.19607843, 0.28235294, 0.09019608],
        [0.16470588, 0.24705882, 0.0745098 ],
             [0.17647059, 0.24313725, 0.10196078],
             ...,
             [0.70588235, 0.64705882, 0.56862745],
             [0.83529412, 0.8       , 0.64705882],
             [0.76470588, 0.74901961, 0.54901961]],

            [[0.19215686, 0.29019608, 0.11372549],
             [0.16078431, 0.24705882, 0.07843137],
             [0.17647059, 0.25490196, 0.10196078],
             ...,
             [0.63137255, 0.58823529, 0.48627451],
             [0.96078431, 0.94901961, 0.75686275],
             [0.93333333, 0.93333333, 0.68627451]],

            [[0.15294118, 0.25882353, 0.1254902 ],
             [0.16470588, 0.2627451 , 0.10980392],
             [0.17254902, 0.25882353, 0.09803922],
             ...,
             [0.57254902, 0.54509804, 0.40784314],
             [0.8627451 , 0.8627451 , 0.62352941],
             [0.76470588, 0.78431373, 0.47058824]],

            ...,

            [[0.80784314, 0.8745098 , 0.43137255],
             [0.87058824, 0.93333333, 0.50196078],
             [0.77647059, 0.8627451 , 0.44705882],
             ...,
             [0.81568627, 0.65882353, 0.5254902 ],
             [0.78431373, 0.63529412, 0.46666667],
             [0.76470588, 0.62352941, 0.42352941]],

            [[0.8745098 , 0.93333333, 0.49019608],
             [0.81568627, 0.88627451, 0.45490196],
             [0.80784314, 0.90588235, 0.49019608],
             ...,
             [0.89803922, 0.70980392, 0.54509804],
             [0.91372549, 0.7254902 , 0.52941176],
             [0.86666667, 0.6745098 , 0.45490196]],

            [[0.94901961, 0.99215686, 0.56078431],
             [0.82745098, 0.89411765, 0.47058824],
             [0.78431373, 0.88627451, 0.47843137],
             ...,
             [0.9372549 , 0.75294118, 0.58039216],
             [0.97647059, 0.79215686, 0.58431373],
             [0.97254902, 0.80392157, 0.57254902]]]])
```

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'))
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`inpu
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
# 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)
```

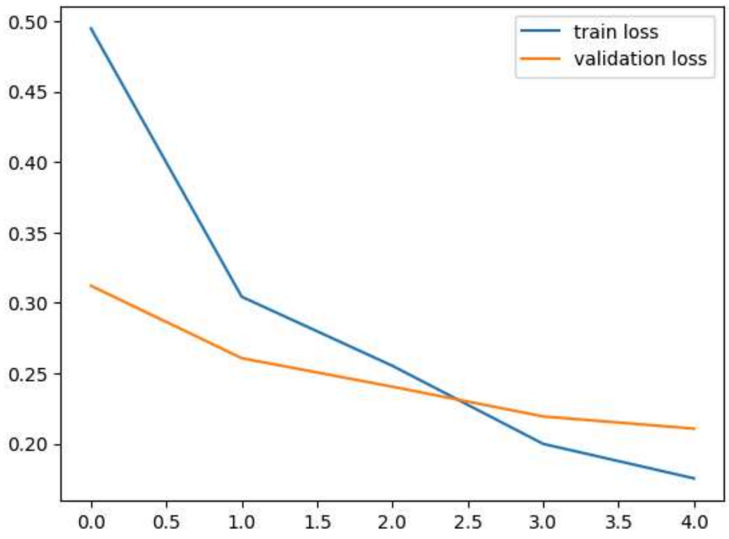
```
Epoch 1/5
170/170 ━━━━━━━━━━━━━━━━━━━━ 13s 43ms/step - acc: 0.6658 - loss: 0.6946 - val_acc: 0.8579 - val_loss: 0.3121
Epoch 2/5
170/170 ━━━━━━━━━━━━━━━━━━━━ 12s 17ms/step - acc: 0.8743 - loss: 0.3056 - val_acc: 0.8793 - val_loss: 0.2608
Epoch 3/5
170/170 ━━━━━━━━━━━━━━━━━━━━ 3s 16ms/step - acc: 0.9071 - loss: 0.2450 - val_acc: 0.8860 - val_loss: 0.2405
Epoch 4/5
170/170 ━━━━━━━━━━━━━━━━━━━━ 5s 16ms/step - acc: 0.9253 - loss: 0.1950 - val_acc: 0.9124 - val_loss: 0.2194
Epoch 5/5
170/170 ━━━━━━━━━━━━━━━━━━━━ 3s 17ms/step - acc: 0.9359 - loss: 0.1713 - val_acc: 0.9091 - val_loss: 0.2108
```

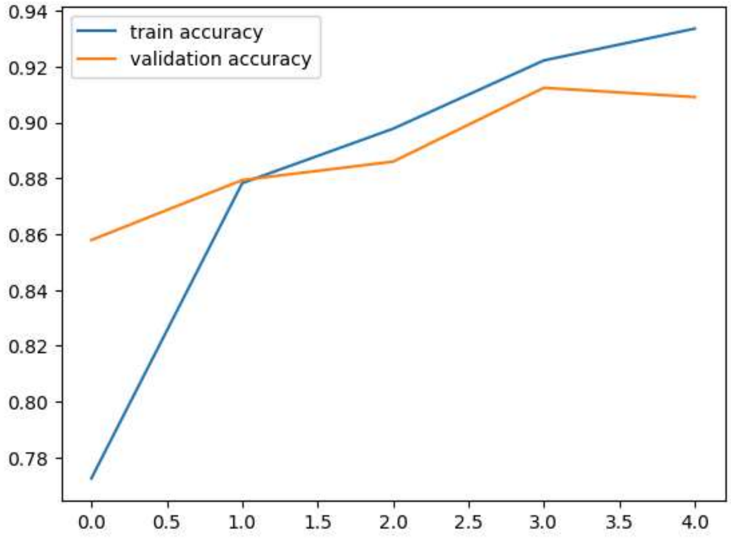
Model Evaluation

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

```
48/48 ━━━━━━━━━━━━━━━━━━━━ 1s 26ms/step - acc: 0.9331 - loss: 0.1790
Test Accuracy = 0.9285241365432739
```

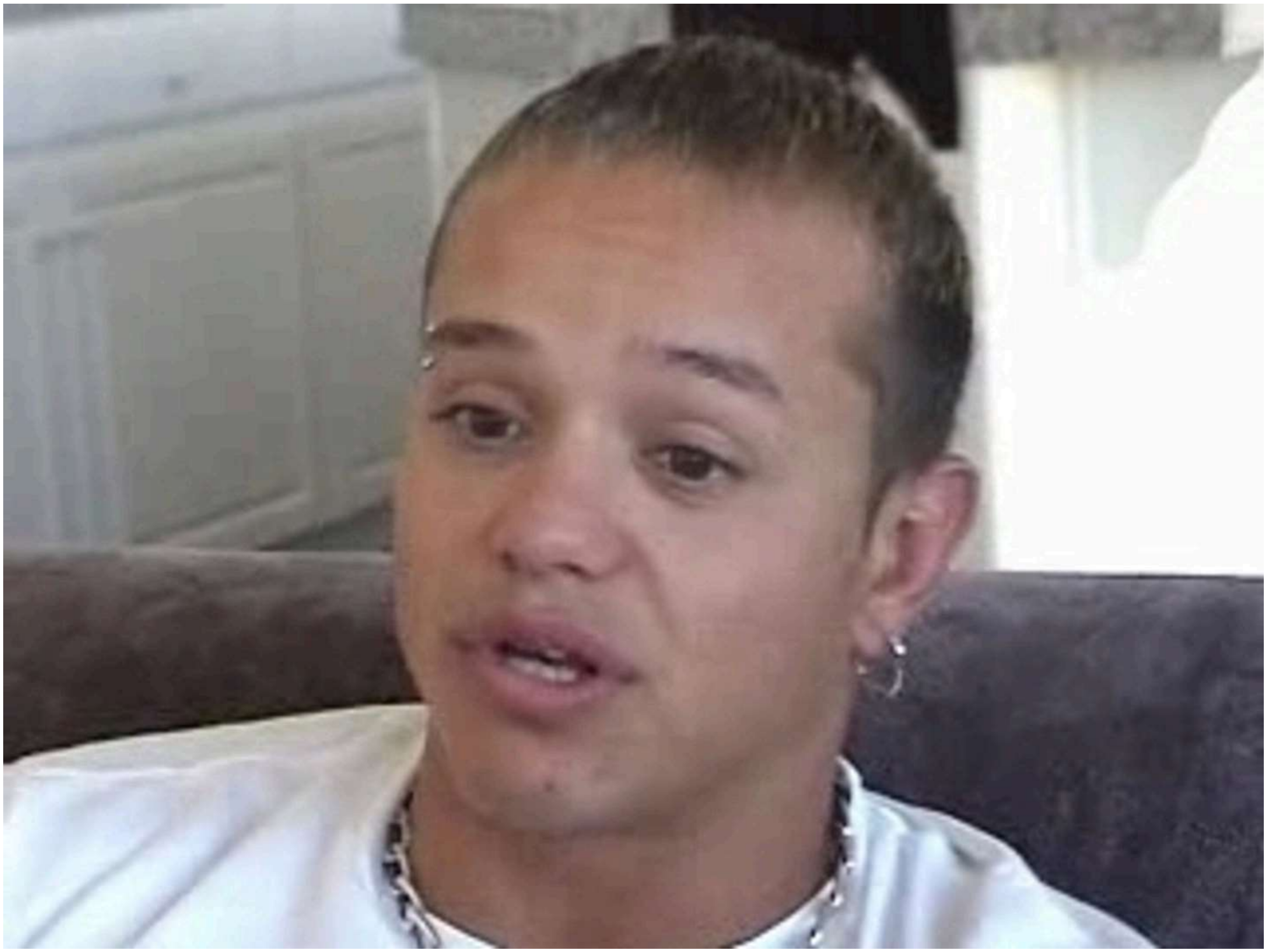
```
h = history

# 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 = "/content/test.jpeg"
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')
```



```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 595ms/step
[[0.21021703 0.79409456]]
1
The person in the image is wearing a mask
```

```
input_image_path = "/content/test.webp"

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:
```