```
!pip install kaggle
```

```
Requirement already satisfied: kaggle in
/usr/local/lib/python3.10/dist-packages (1.5.16)
Requirement already satisfied: six>=1.10 in
/usr/local/lib/python3.10/dist-packages (from kaggle) (1.16.0)
Requirement already satisfied: certifi in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2024.2.2)
Requirement already satisfied: python-dateutil in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2.8.2)
Requirement already satisfied: requests in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2.31.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-
packages (from kaggle) (4.66.2)
Requirement already satisfied: python-slugify in
/usr/local/lib/python3.10/dist-packages (from kaggle) (8.0.4)
Requirement already satisfied: urllib3 in
/usr/local/lib/python3.10/dist-packages (from kaggle) (2.0.7)
Requirement already satisfied: bleach in
/usr/local/lib/python3.10/dist-packages (from kaggle) (6.1.0)
Requirement already satisfied: webencodings in
/usr/local/lib/python3.10/dist-packages (from bleach->kaggle) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in
/usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle)
(1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests->kaggle)
(3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests->kaggle) (3.6)
```

```
!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)
```

```python
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
```

Therefore, we have all the required files now, after importing them directly from Kaggle :)

```
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_3536.jpg', 'with_mask_1265.jpg', 'with_mask_3039.jpg',
'with_mask_1615.jpg', 'with_mask_2969.jpg']
['with_mask_2357.jpg', 'with_mask_3477.jpg', 'with_mask_1559.jpg',
'with_mask_2794.jpg', 'with_mask_754.jpg']
```

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

```
['without_mask_1199.jpg', 'without_mask_2045.jpg',
'without_mask_2369.jpg', 'without_mask_496.jpg',
'without_mask_2807.jpg']
['without_mask_1620.jpg', 'without_mask_3453.jpg',
'without_mask_2403.jpg', 'without_mask_864.jpg',
'without_mask_1431.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
```

```
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]
```

```
img = mpimg.imread('/content/data/with_mask/with_mask_1545.jpg')
imgplot = plt.imshow(img)
plt.show()
```

```
img = mpimg.imread('/content/data/without_mask/without_mask_2925.jpg')
imgplot = plt.imshow(img)
plt.show()
```

```
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/python3.10/dist-packages/PIL/Image.py:996: UserWarning:
Palette images with Transparency expressed in bytes should be
converted to RGBA images
  warnings.warn(
```

```
type(data)
```

```
list
```

```
len(data)
```

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

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       ...,

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]],

       [[255, 255, 255],
        [255, 255, 255],
        [255, 255, 255],
        ...,
        [255, 255, 255],
        [255, 255, 255],
        [255, 255, 255]]], dtype=uint8)
```

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

```
numpy.ndarray
```

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

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

```
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)
```

```
X_train_scaled = X_train/255

X_test_scaled = X_test/255

X_train[0]
```

```
array([[[254, 254, 254],
        [254, 254, 254],
        [254, 254, 254],
        ...,
        [254, 254, 254],
        [254, 254, 254],
        [254, 254, 254]],

       [[254, 254, 254],
        [254, 254, 254],
        [254, 254, 254],
        ...,
        [254, 254, 254],
        [254, 254, 254],
        [254, 254, 254]],

       [[254, 254, 254],
        [254, 254, 254],
        [254, 254, 254],
        ...,
        [254, 254, 254],
        [254, 254, 254],
        [254, 254, 254]],

       ...,

       [[224, 199, 187],
        [164, 131, 118],
        [183, 142, 127],
        ...,
        [229, 189, 167],
        [220, 180, 158],
        [211, 171, 149]],

       [[181, 146, 131],
        [174, 135, 120],
        [186, 141, 123],
        ...,
        [228, 191, 170],
        [219, 181, 160],
        [210, 172, 151]],

       [[167, 129, 113],
        [176, 134, 117],
        [188, 142, 123],
        ...,
        [228, 192, 170],
        [219, 183, 161],
        [210, 174, 152]]], dtype=uint8)
```

```
X_train_scaled[0]
```

```
array([[[0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        ...,
        [0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843]],

       [[0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        ...,
        [0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843]],

       [[0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        ...,
        [0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843],
        [0.99607843, 0.99607843, 0.99607843]],

       ...,

       [[0.87843137, 0.78039216, 0.73333333],
        [0.64313725, 0.51372549, 0.4627451 ],
        [0.71764706, 0.55686275, 0.49803922],
        ...,
        [0.89803922, 0.74117647, 0.65490196],
        [0.8627451 , 0.70588235, 0.61960784],
        [0.82745098, 0.67058824, 0.58431373]],

       [[0.70980392, 0.57254902, 0.51372549],
        [0.68235294, 0.52941176, 0.47058824],
        [0.72941176, 0.55294118, 0.48235294],
        ...,
        [0.89411765, 0.74901961, 0.66666667],
        [0.85882353, 0.70980392, 0.62745098],
        [0.82352941, 0.6745098 , 0.59215686]],
       [[0.65490196, 0.50588235, 0.44313725],
        [0.69019608, 0.5254902 , 0.45882353],
        [0.7372549 , 0.55686275, 0.48235294],
        ...,
        [0.89411765, 0.75294118, 0.66666667],
        [0.85882353, 0.71764706, 0.63137255],
        [0.82352941, 0.68235294, 0.59607843]]])
```

CNN Architecture

```
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(32, activation='relu'))
model.add(keras.layers.Dropout(0.5))

model.add(keras.layers.Dense(16, 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'])
```

```
history = model.fit(X_train_scaled, Y_train, validation_split=0.1,
epochs=25)

Epoch 1/25
170/170 [==============================] - 6s 21ms/step - loss: 0.7223
- acc: 0.5144 - val_loss: 0.6927 - val_acc: 0.5140
Epoch 2/25
170/170 [==============================] - 3s 17ms/step - loss: 0.6640
- acc: 0.5983 - val_loss: 0.5637 - val_acc: 0.8099
Epoch 3/25
170/170 [==============================] - 3s 19ms/step - loss: 0.5526
- acc: 0.7603 - val_loss: 0.5638 - val_acc: 0.7008
Epoch 4/25
170/170 [==============================] - 3s 17ms/step - loss: 0.5202
- acc: 0.7690 - val_loss: 0.3962 - val_acc: 0.8893
Epoch 5/25
170/170 [==============================] - 3s 17ms/step - loss: 0.4710
- acc: 0.8023 - val_loss: 0.3491 - val_acc: 0.8909
Epoch 6/25
170/170 [==============================] - 3s 17ms/step - loss: 0.3614
- acc: 0.8685 - val_loss: 0.2790 - val_acc: 0.9074
Epoch 7/25
170/170 [==============================] - 3s 19ms/step - loss: 0.3043
- acc: 0.8931 - val_loss: 0.2750 - val_acc: 0.8975
Epoch 8/25
170/170 [==============================] - 3s 18ms/step - loss: 0.2682
- acc: 0.9027 - val_loss: 0.2509 - val_acc: 0.9140
Epoch 9/25
170/170 [==============================] - 3s 18ms/step - loss: 0.2479
- acc: 0.9169 - val_loss: 0.2424 - val_acc: 0.9256
Epoch 10/25
170/170 [==============================] - 3s 18ms/step - loss: 0.2152
- acc: 0.9268 - val_loss: 0.2549 - val_acc: 0.9157
Epoch 11/25
170/170 [==============================] - 3s 19ms/step - loss: 0.1856
- acc: 0.9367 - val_loss: 0.2477 - val_acc: 0.9223
Epoch 12/25
170/170 [==============================] - 3s 17ms/step - loss: 0.1613
- acc: 0.9468 - val_loss: 0.3027 - val_acc: 0.9223
Epoch 13/25
170/170 [==============================] - 3s 17ms/step - loss: 0.1550
- acc: 0.9496 - val_loss: 0.2888 - val_acc: 0.9124
Epoch 14/25
170/170 [==============================] - 3s 17ms/step - loss: 0.1808
- acc: 0.9408 - val_loss: 0.2519 - val_acc: 0.9157
Epoch 15/25
170/170 [==============================] - 3s 19ms/step - loss: 0.1421
- acc: 0.9551 - val_loss: 0.3272 - val_acc: 0.9223
Epoch 16/25
```

```
170/170 [==============================] - 3s 17ms/step - loss: 0.1284
- acc: 0.9628 - val_loss: 0.3126 - val_acc: 0.9355
Epoch 17/25
170/170 [==============================] - 3s 17ms/step - loss: 0.1025
- acc: 0.9676 - val_loss: 0.3933 - val_acc: 0.9074
Epoch 18/25
170/170 [==============================] - 3s 18ms/step - loss: 0.0943
- acc: 0.9704 - val_loss: 0.3561 - val_acc: 0.9223
Epoch 19/25
170/170 [==============================] - 3s 19ms/step - loss: 0.0992
- acc: 0.9697 - val_loss: 0.3492 - val_acc: 0.9223
Epoch 20/25
170/170 [==============================] - 3s 18ms/step - loss: 0.1593
- acc: 0.9514 - val_loss: 0.3300 - val_acc: 0.9273
Epoch 21/25
170/170 [==============================] - 3s 17ms/step - loss: 0.0959
- acc: 0.9693 - val_loss: 0.3850 - val_acc: 0.9207
Epoch 22/25
170/170 [==============================] - 3s 18ms/step - loss: 0.0816
- acc: 0.9739 - val_loss: 0.4665 - val_acc: 0.9223
Epoch 23/25
170/170 [==============================] - 3s 19ms/step - loss: 0.0934
- acc: 0.9733 - val_loss: 0.3976 - val_acc: 0.9008
Epoch 24/25
170/170 [==============================] - 3s 18ms/step - loss: 0.0849
- acc: 0.9744 - val_loss: 0.4947 - val_acc: 0.9107
Epoch 25/25
170/170 [==============================] - 3s 18ms/step - loss: 0.0619
- acc: 0.9809 - val_loss: 0.6253 - val_acc: 0.9025
```

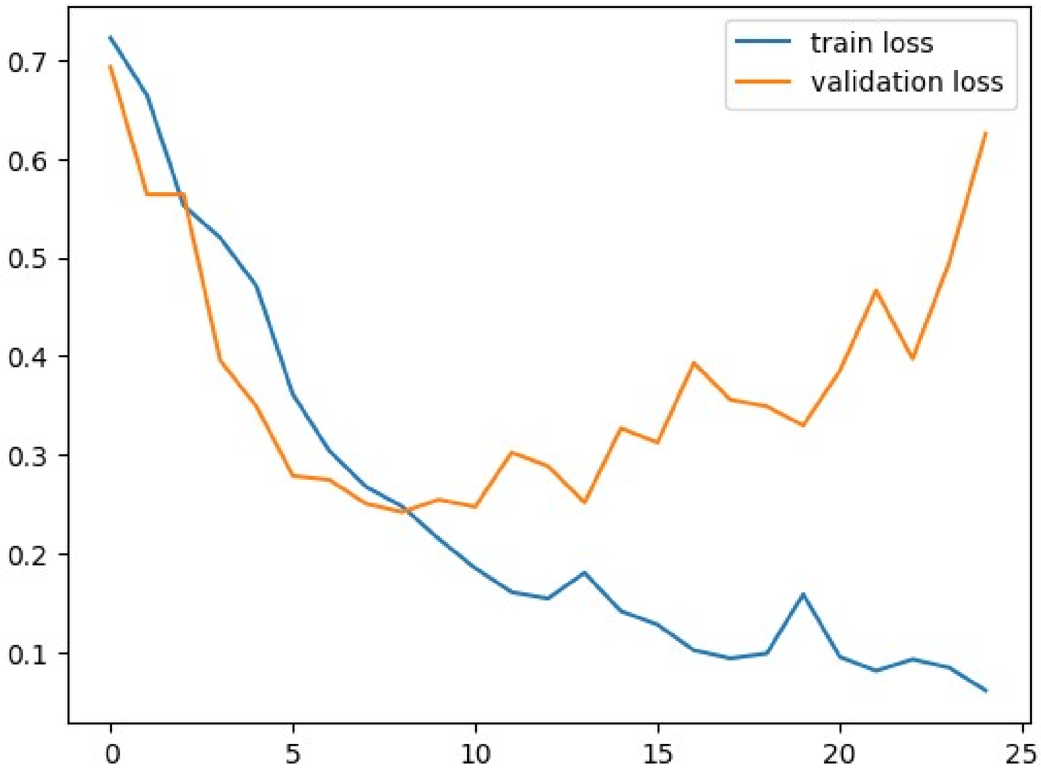
```
loss, accuracy = model.evaluate(X_test_scaled, Y_test)
print('Test Accuracy =', accuracy)
```

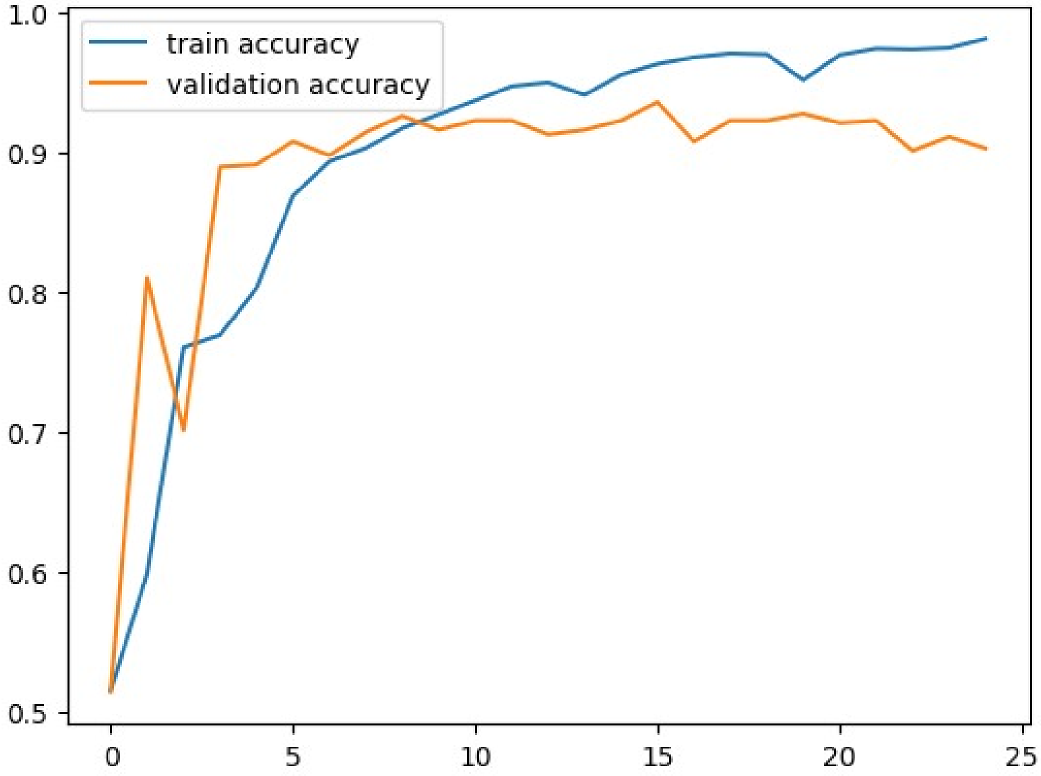
```
48/48 [==============================] - 0s 8ms/step - loss: 0.6268 -
acc: 0.9226
Test Accuracy = 0.9225678443908691
```

```
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()
```


train loss
validation loss
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
5
10
15
20
25

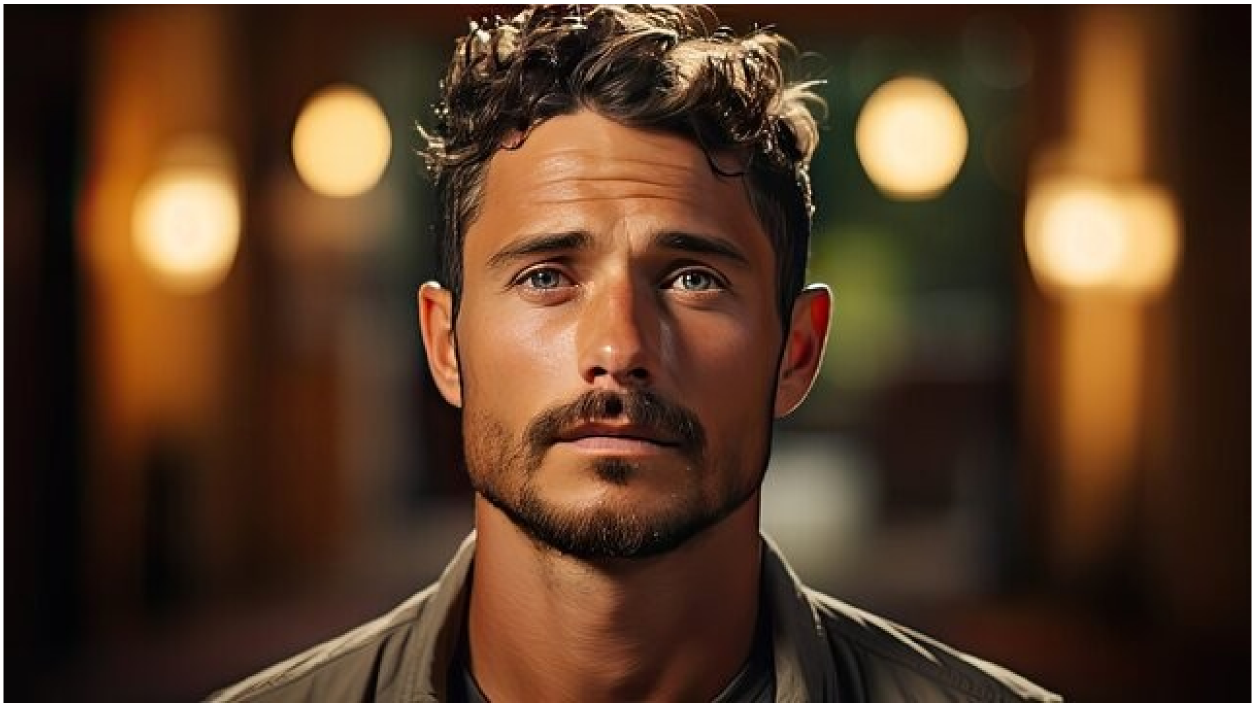

train accuracy
validation accuracy
1.0
0.9
0.8
0.7
0.6
0.5
0
5
10
15
20
25

```python
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/Withoutmask.jpg
```



```
1/1 [==============================] - 0s 97ms/step
[[1. 1.]]
0
The person in the image is not wearing a mask
```