uophc6vnt

May 7, 2024

0.1 Mood classfication using CNN in GPU (HAPPY / SAD)

[1]:
```python
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

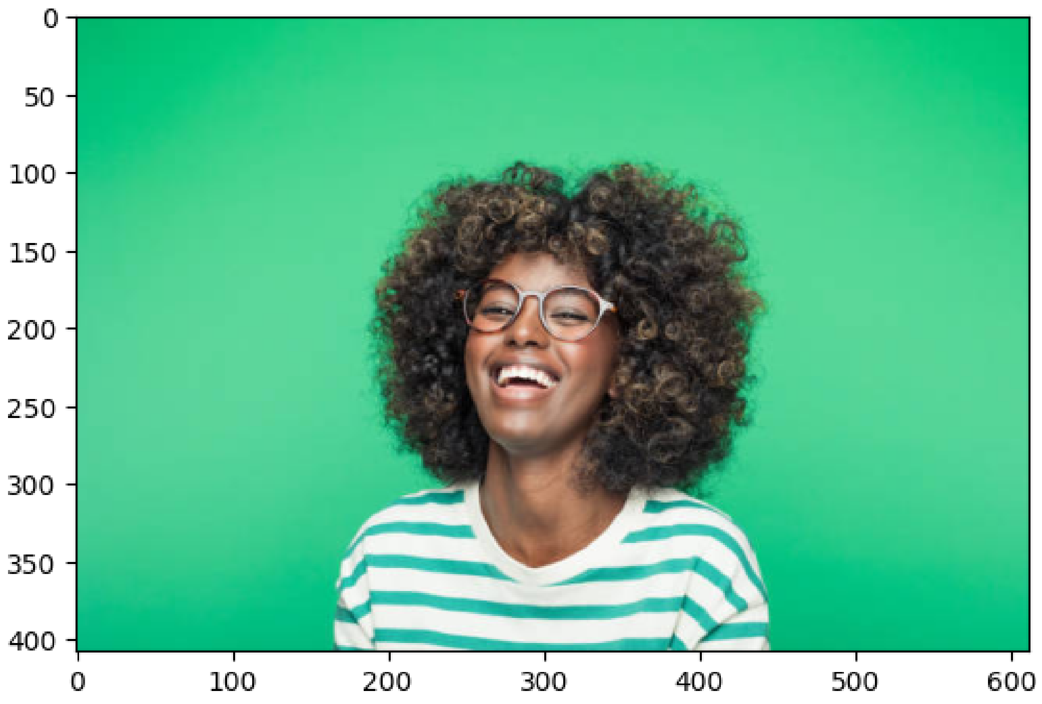
Steps * Create 3 folder. * Traing, testing and validation. * Please collect 20 pics (collect Happy & sad images) * Inside training creat 2 folder as happy and not happy * paste all the images in testing part

[2]:
```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator  #for data␣
 ↪augmentation in image processing tasks
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import cv2
import os
#image data generator is the package to lable the images & it will␣
 ↪automatically lable all the images
```

[3]:
```python
img = image.load_img('/content/drive/MyDrive/Img Class/1. Training/Happy/snsns.
 ↪jpg')
```

[4]:
```python
plt.imshow(img)
```

[4]:
```
<matplotlib.image.AxesImage at 0x798a84990a00>
```



```
[10]: i1 = cv2.imread(r'/content/drive/MyDrive/Img Class/1. Training/Happy/snsns.jpg')
      i1
      # 3 dimension metrics are created for the image
      # the value ranges from 0-255
```

```
[10]: array([[[109, 180,   0],
              [109, 180,   0],
              [109, 180,   0],
              …,
              [115, 194,   1],
              [115, 194,   1],
              [115, 194,   1]],

             [[109, 180,   0],
              [109, 180,   0],
              [109, 180,   0],
              …,
              [115, 194,   1],
              [115, 194,   1],
              [115, 194,   1]],

             [[110, 181,   0],
              [110, 181,   0],
              [110, 181,   0],
              …,
              [115, 194,   1],
              [115, 194,   1],
              [115, 194,   1]],

             …,

             [[127, 182,   1],
              [127, 182,   1],
              [127, 182,   1],
              …,
              [125, 189,   0],
              [125, 189,   0],
              [125, 189,   0]],

             [[127, 182,   1],
              [127, 182,   1],
              [127, 182,   1],
              …,
              [124, 188,   0],
              [124, 188,   0],
              [124, 188,   0]],

             [[127, 182,   1],
              [127, 182,   1],
              [127, 182,   1],
              …,
              [124, 188,   0],
              [124, 188,   0],
              [124, 188,   0]]], dtype=uint8)
```

[11]:
```
i1.shape
# shape of your image height, weight, rgb
```

[11]:
```
(408, 612, 3)
```

[12]:
```
train = ImageDataGenerator(rescale = 1/255)
validataion = ImageDataGenerator(rescale = 1/255)
# to scale all the images i need to divide with 255
# we need to resize the image using 200, 200 pixel
```

[13]:
```
train_dataset = train.flow_from_directory(r'/content/drive/MyDrive/Img Class/1.␣
 ↪Training',
                                          target_size = (200,200),
                                          batch_size = 3,
                                          class_mode = 'binary')
```

```
validataion_dataset = validataion.flow_from_directory(r'/content/drive/MyDrive/
 ↪Img Class/3. Validations',
                                                       target_size = (200,200),
                                                       batch_size = 3,
                                                       class_mode = 'binary')
```

```
Found 16 images belonging to 2 classes.
Found 0 images belonging to 2 classes.
```

[14]:
```
train_dataset.class_indices
```

[14]:
```
{'Happy': 0, 'Not Happy': 1}
```

[15]:
```
train_dataset.classes
```

[15]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1], dtype=int32)
```

[16]:
```
# now we are applying maxpooling

model = tf.keras.models.Sequential([ tf.keras.layers.Conv2D(16,(3,3),activation␣
 ↪= 'relu',input_shape = (200,200,3)),
                                    tf.keras.layers.MaxPool2D(2,2), #3 filtr we␣
 ↪applied hear
                                    #
                                    tf.keras.layers.Conv2D(32,(3,3),activation␣
 ↪= 'relu'),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    #
                                    tf.keras.layers.Conv2D(64,(3,3),activation␣
 ↪= 'relu'),
                                    tf.keras.layers.MaxPool2D(2,2),
                                    ##
                                    tf.keras.layers.Flatten(),
                                    ##
                                    tf.keras.layers.Dense(512, activation =␣
 ↪'relu'),
                                    #
                                    tf.keras.layers.Dense(1,activation=␣
 ↪'sigmoid')
                                    ]
                                    )
```

[17]:
```
model.compile(loss='binary_crossentropy',
              optimizer = tf.keras.optimizers.RMSprop(lr = 0.001),
              metrics = ['accuracy']
              )
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate`
or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.RMSprop.
```

[27]:
```
model_fit = model.fit(train_dataset,
                      epochs = 10,
                      validation_data = validataion_dataset)
```

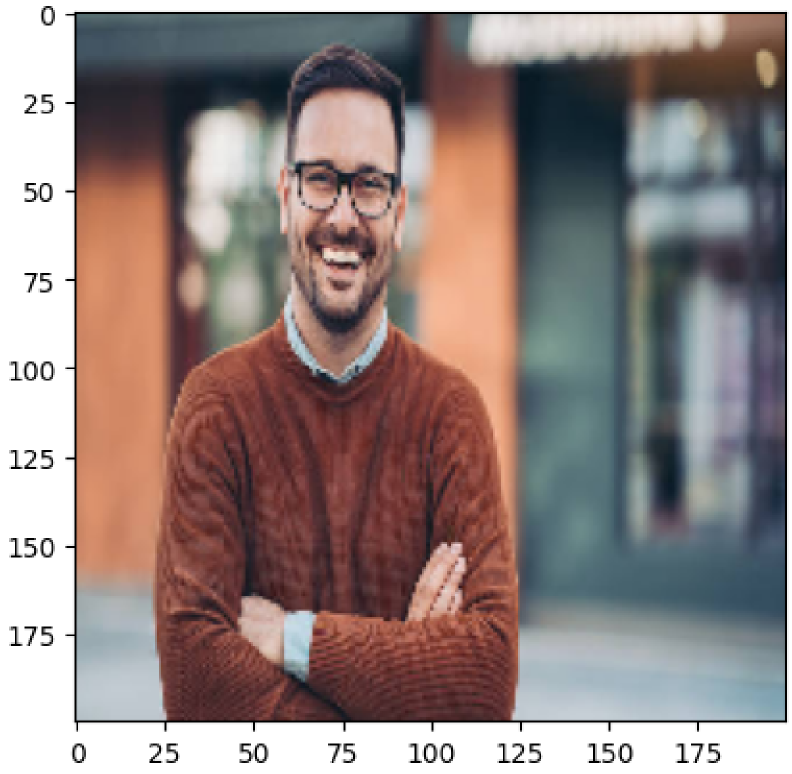
```
Epoch 1/10
6/6 [==============================] - 0s 47ms/step - loss: 3.3804e-06 -
accuracy: 1.0000
Epoch 2/10
6/6 [==============================] - 0s 83ms/step - loss: 3.2062e-06 -
accuracy: 1.0000
Epoch 3/10
6/6 [==============================] - 0s 55ms/step - loss: 3.0661e-06 -
accuracy: 1.0000
Epoch 4/10
6/6 [==============================] - 0s 37ms/step - loss: 2.9047e-06 -
accuracy: 1.0000
Epoch 5/10
6/6 [==============================] - 0s 37ms/step - loss: 2.7797e-06 -
accuracy: 1.0000
Epoch 6/10
6/6 [==============================] - 0s 33ms/step - loss: 2.6686e-06 -
accuracy: 1.0000
Epoch 7/10
6/6 [==============================] - 0s 23ms/step - loss: 2.5652e-06 -
accuracy: 1.0000
Epoch 8/10
6/6 [==============================] - 0s 26ms/step - loss: 2.4572e-06 -
accuracy: 1.0000
Epoch 9/10
6/6 [==============================] - 0s 26ms/step - loss: 2.3601e-06 -
accuracy: 1.0000
Epoch 10/10
6/6 [==============================] - 0s 25ms/step - loss: 2.2783e-06 -
accuracy: 1.0000
```

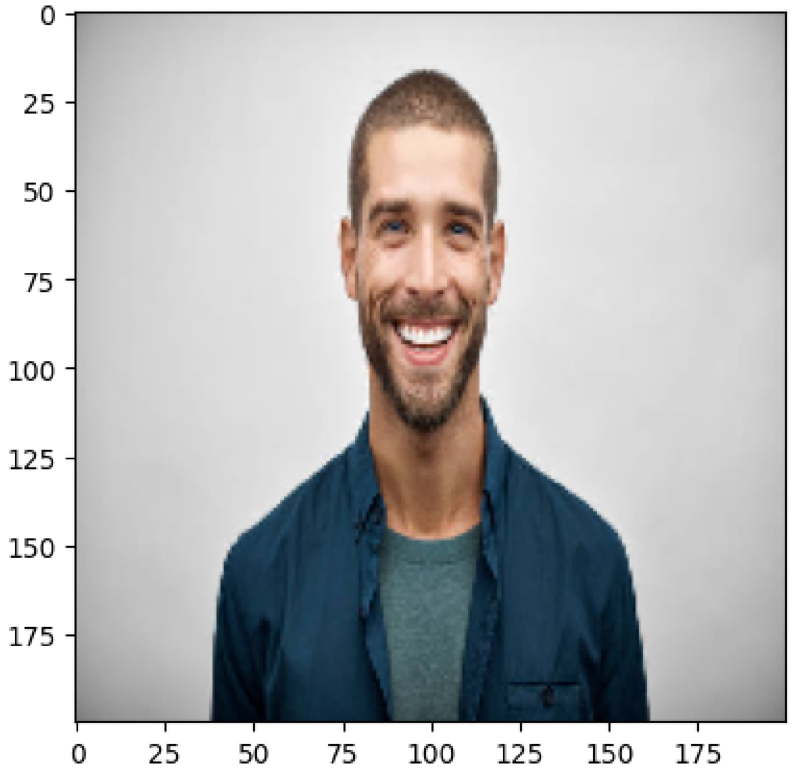
[28]:
```
dir_path = r'/content/drive/MyDrive/Img Class/2. Testing'
for i in os.listdir(dir_path ):
    print(i)
    #img = image.load_img(dir_path+ '//'+i, target_size = (200,200))
   # plt.imshow(img)
   # plt.show()
```

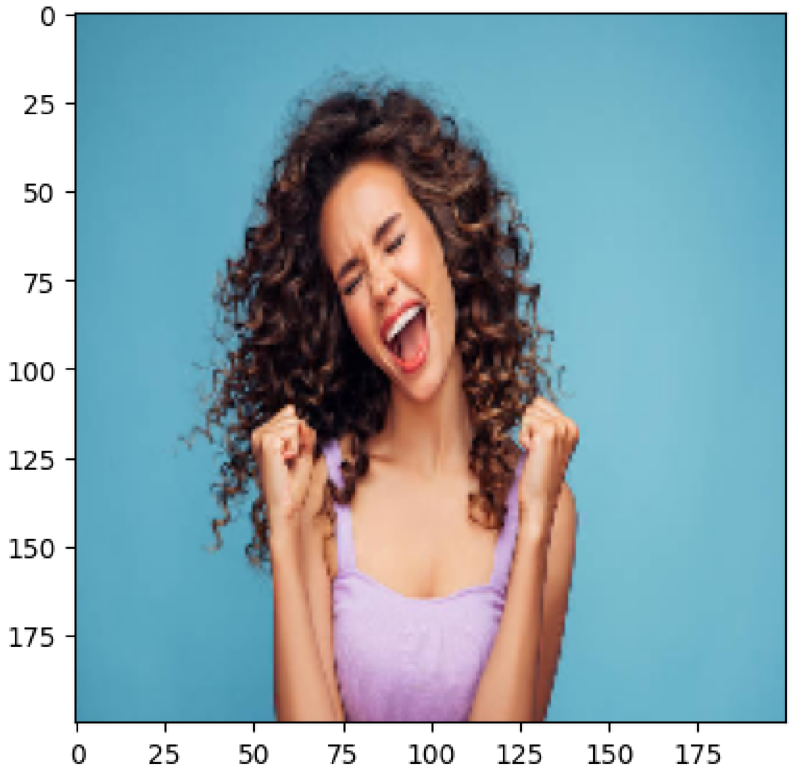
```
1.jpg
2.jpg
3.jpg
```

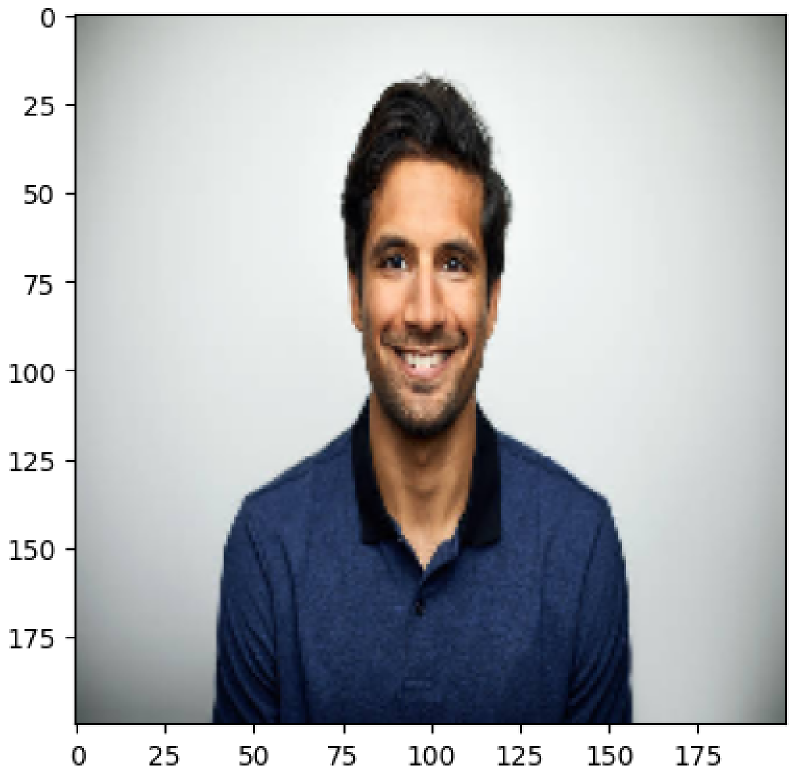
```
4.jpg
5.jpg
avds.jpg
adv.jpg
djdj.jpg
asx.jpg
asxaw.jpg
```

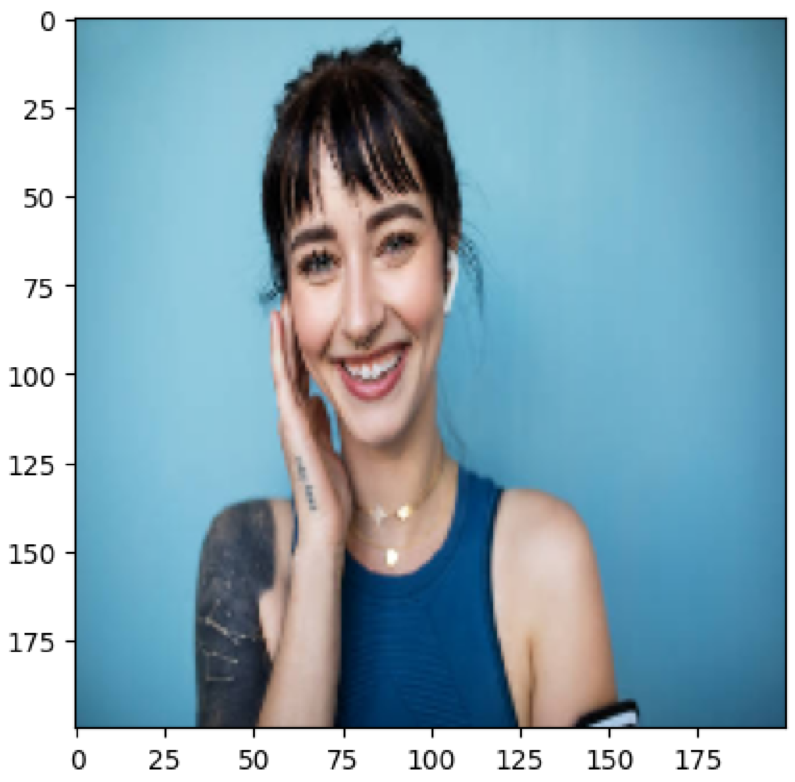
[29]:
```
dir_path = r'/content/drive/MyDrive/Img Class/2. Testing'
for i in os.listdir(dir_path ):
    img = image.load_img(dir_path+ '//'+i, target_size = (200,200))
    plt.imshow(img)
    plt.show()
```

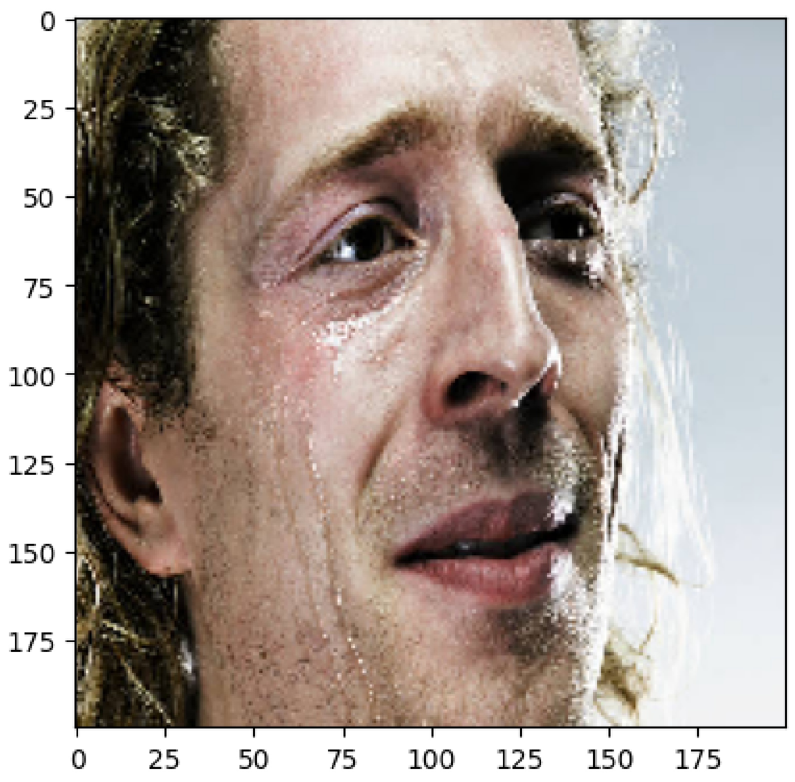


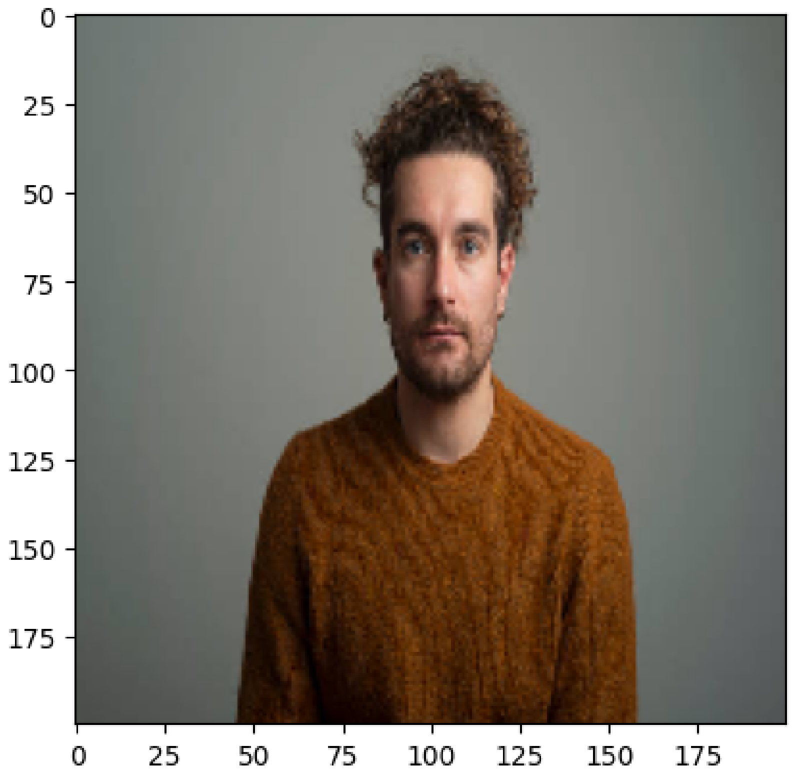

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

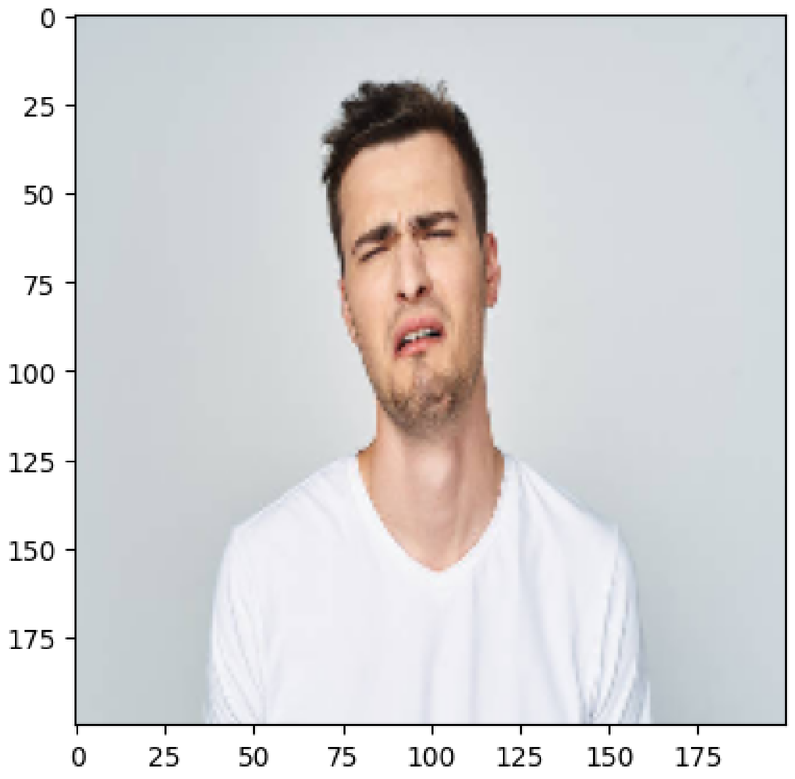

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

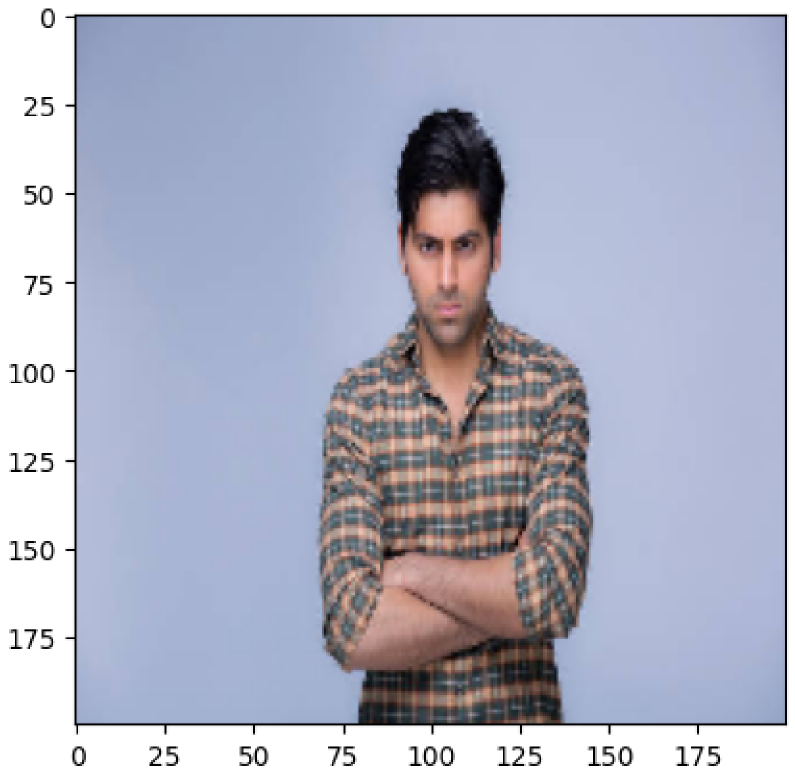

0
25
50
75
100
125
150
175

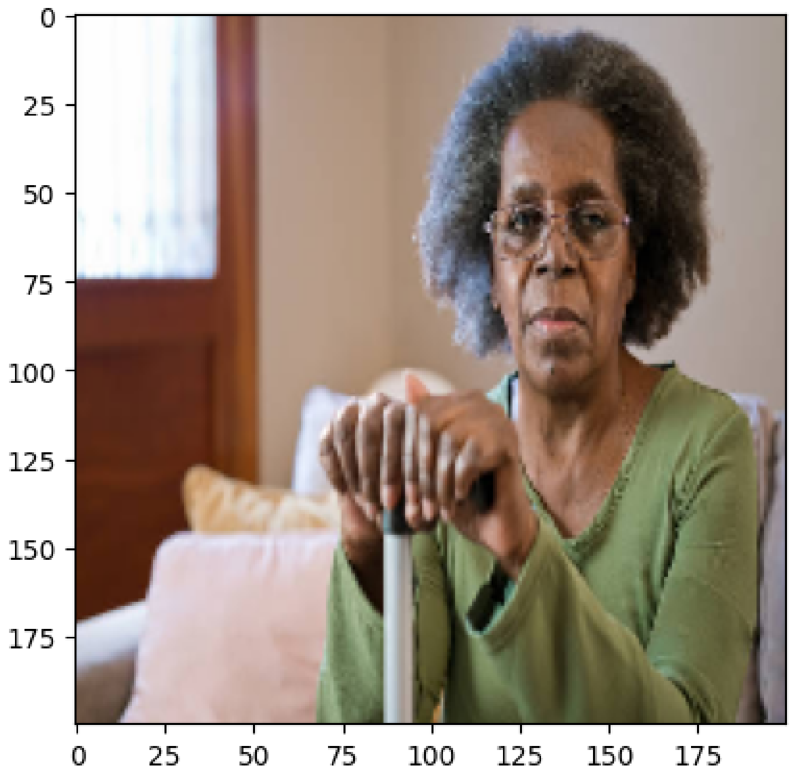

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

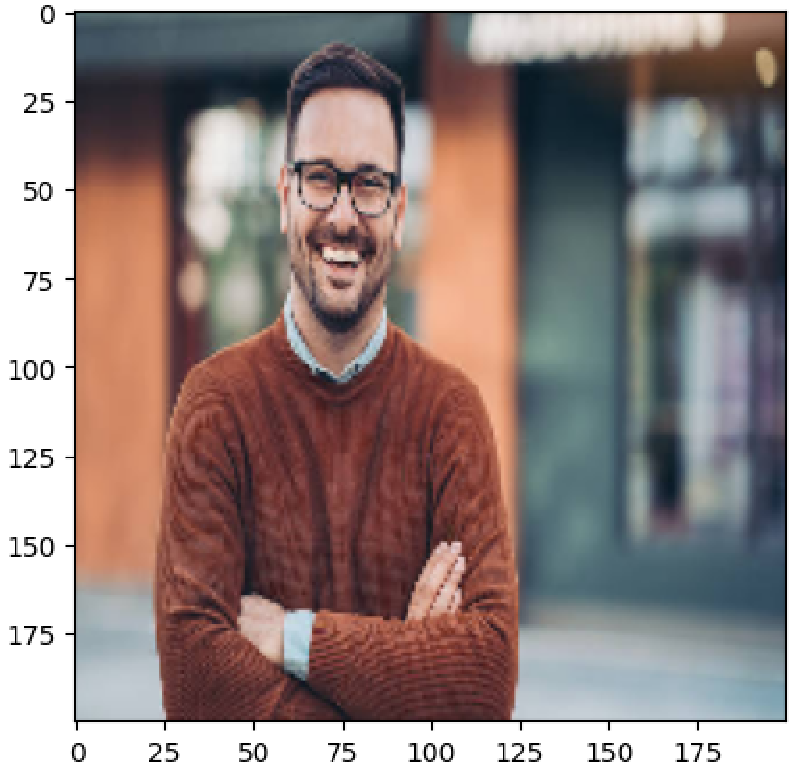

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

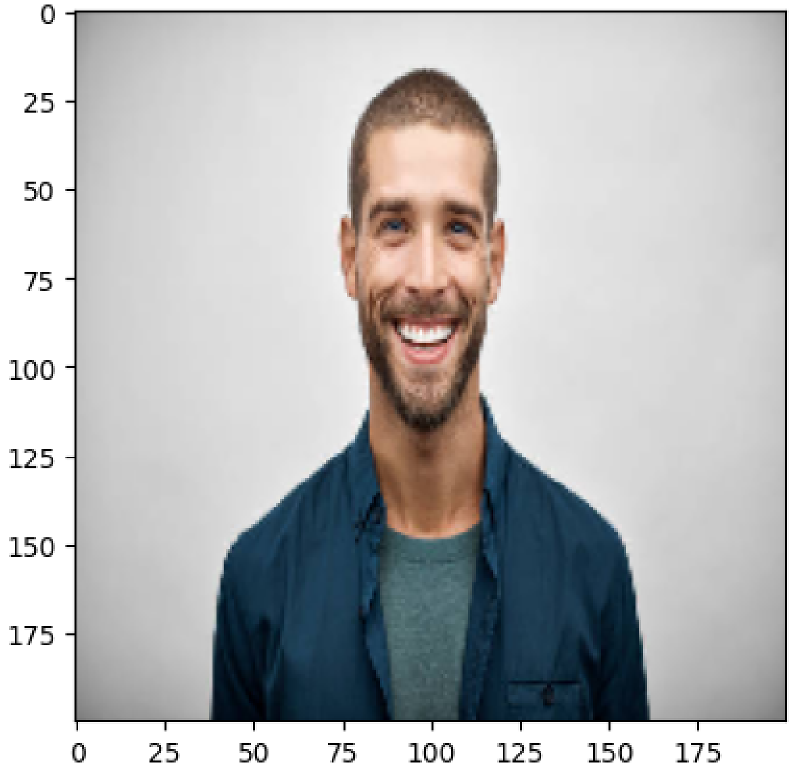

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175



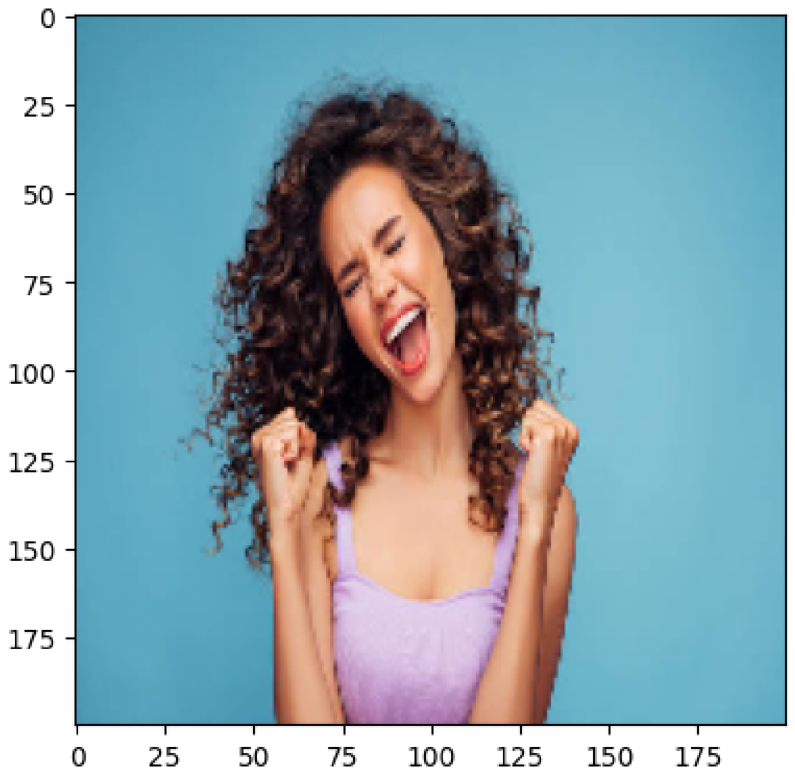
[31]:
```
dir_path = r'/content/drive/MyDrive/Img Class/2. Testing'
for i in os.listdir(dir_path ):
    img = image.load_img(dir_path+ '//'+i, target_size = (200,200))
    plt.imshow(img)
    plt.show()

    x= image.img_to_array(img)
    x=np.expand_dims(x,axis = 0)
    images = np.vstack([x])

    val = model.predict(images)
    if val == 0:
        print( ' i am happy')
    else:
        print('i am not happy')
```



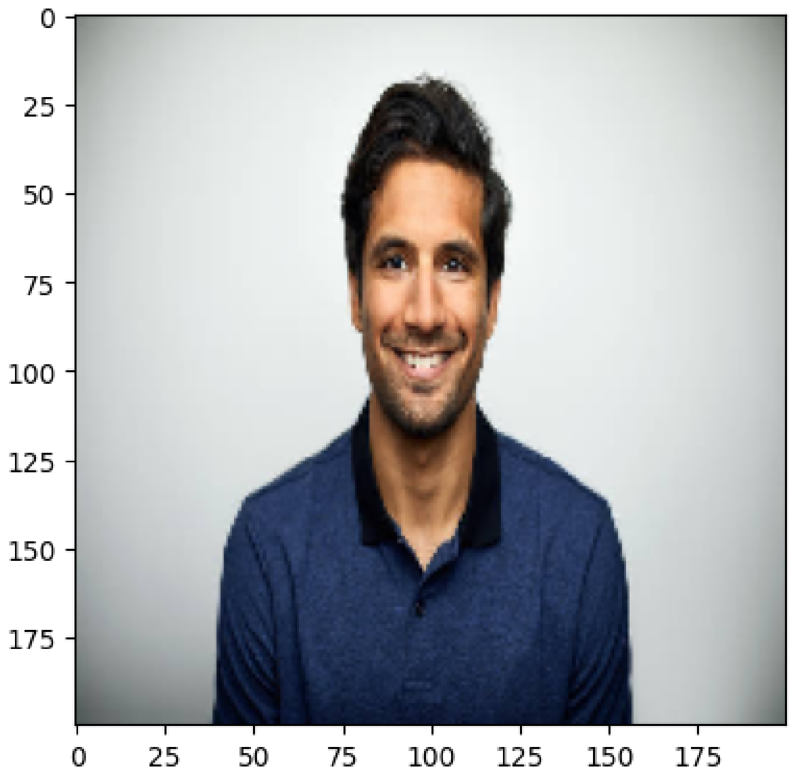
```
1/1 [==============================] - 0s 44ms/step
 i am happy
```



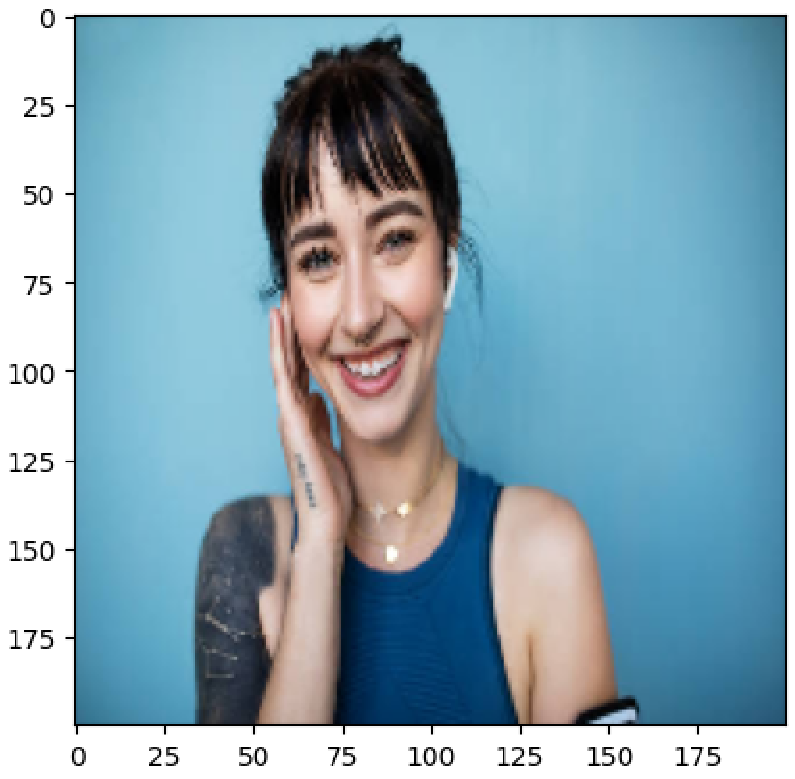
```
1/1 [==============================] - 0s 21ms/step
 i am happy
```



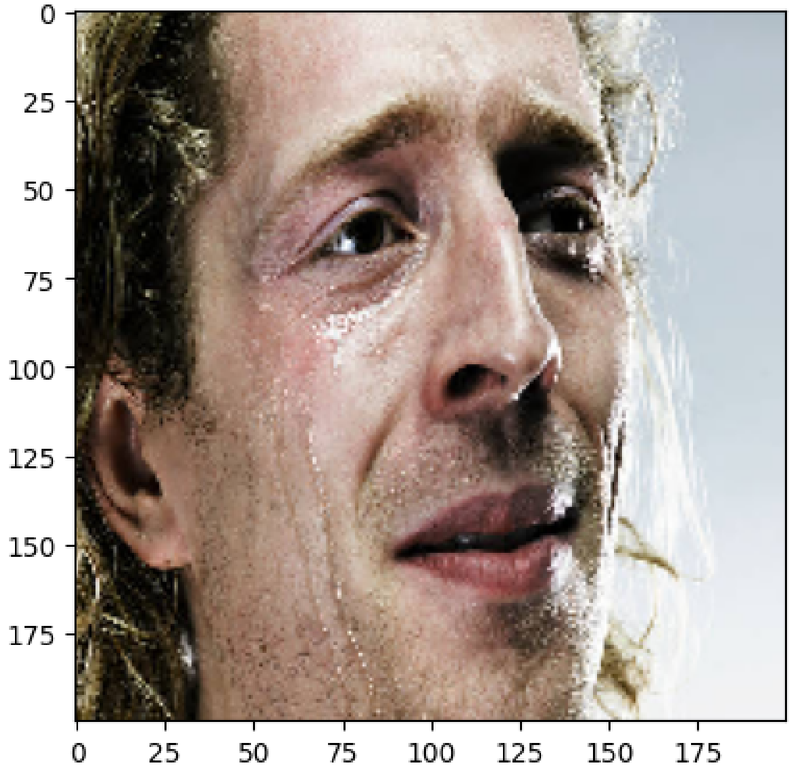
```
1/1 [==============================] - 0s 19ms/step
 i am happy
```



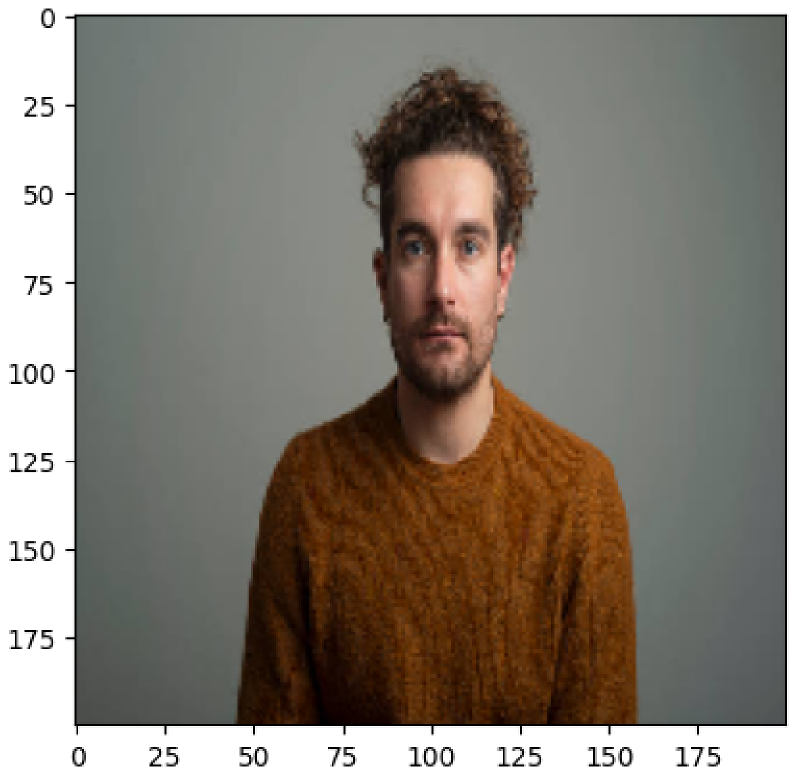
```
1/1 [==============================] - 0s 18ms/step
 i am happy
```



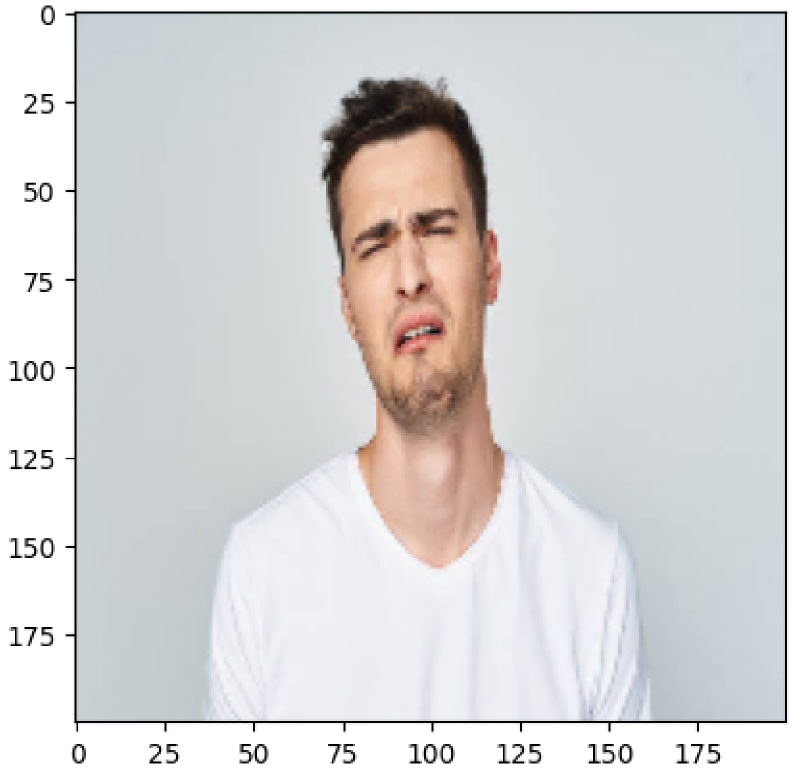
```
1/1 [==============================] - 0s 22ms/step
 i am happy
```



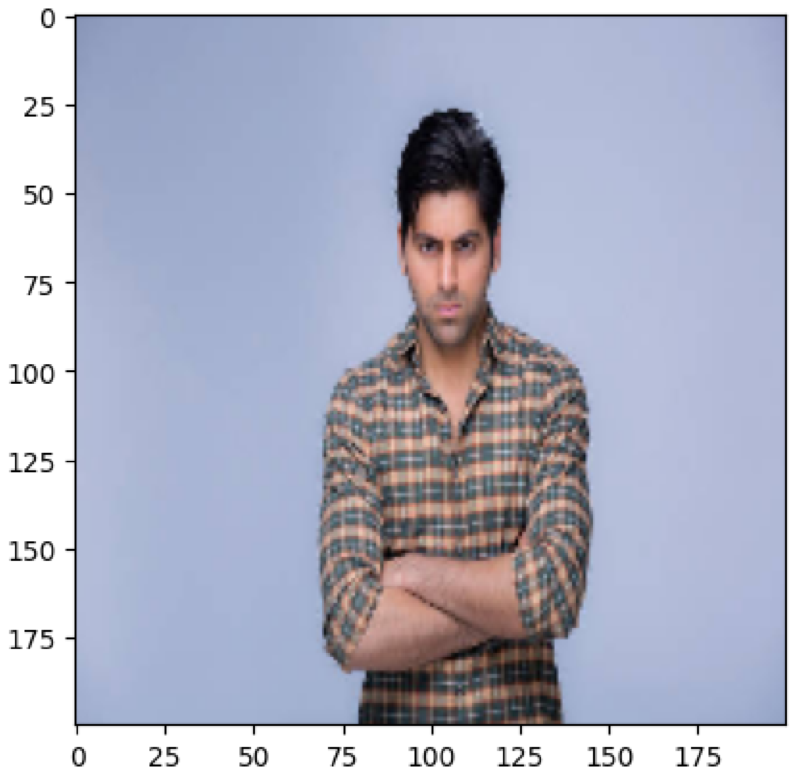
```
1/1 [==============================] - 0s 19ms/step
i am not happy
```



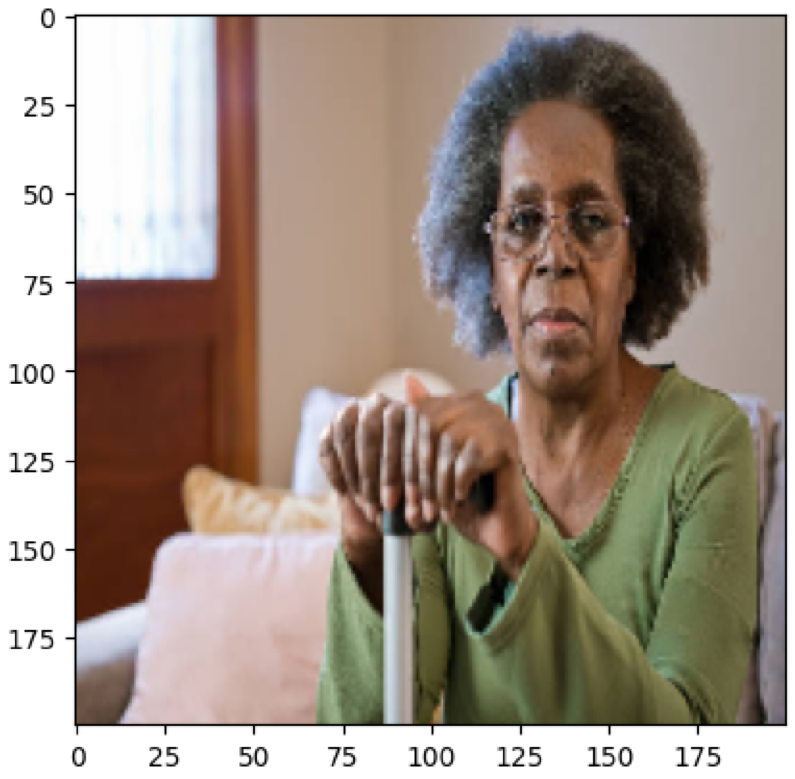
```
1/1 [==============================] - 0s 19ms/step
i am not happy
```



```
1/1 [==============================] - 0s 18ms/step
i am not happy
```



```
1/1 [==============================] - 0s 19ms/step
i am not happy
```



```
1/1 [==============================] - 0s 19ms/step
i am not happy
```