Mood classfication using CNN (HAPPY / SAD)

```
### STEPS -
- Create 3 folder in your desktop
- Training, Testing, Validation
- Inside training create 2 folder as happy or not happy
- paste all the photo in testing part
```

In [2]:

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
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
```

```
WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python311\site-packages\keras\src\losses.py:2976: Th
e name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy in
stead.
```

In [3]:

```python
img = image.load_img(r'D:\Data Science with AI\14th-feb-2024\image classification\training\happy\5.jpeg')
```

In [4]:

```python
plt.imshow(img)
```

Out[4]:

```
<matplotlib.image.AxesImage at 0x20e402c9710>
```

In [5]:
```
i1 = cv2.imread(r'D:\Data Science with AI\14th-feb-2024\image classification\training\happy\5.jpeg')
i1
# 3 dimension metrics are created for the image
# the value ranges from 0-255
```

Out[5]:
```
array([[[176, 184, 184],
        [176, 184, 184],
        [176, 184, 184],
        ...,
        [230, 227, 229],
        [233, 230, 232],
        [236, 233, 235]],

       [[175, 183, 183],
        [175, 183, 183],
        [176, 184, 184],
        ...,
        [227, 224, 226],
        [230, 227, 229],
        [232, 229, 231]],

       [[173, 181, 181],
        [174, 182, 182],
        [174, 182, 182],
        ...,
        [223, 220, 222],
        [226, 223, 225],
        [228, 225, 227]],

       ...,

       [[190, 182, 182],
        [190, 182, 182],
        [190, 182, 182],
        ...,
        [186, 185, 194],
        [186, 185, 195],
        [185, 186, 196]],

       [[191, 183, 183],
        [192, 184, 184],
        [194, 186, 186],
        ...,
        [198, 195, 204],
        [195, 194, 204],
        [186, 185, 195]],

       [[191, 183, 183],
        [192, 184, 184],
        [194, 186, 186],
        ...,
        [198, 195, 204],
        [197, 193, 204],
        [186, 185, 195]]], dtype=uint8)
```

In [6]:
```
i1.shape
# shape of your image height, weight, rgb
```

Out[6]:
```
(234, 384, 3)
```

In [7]:
```
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
```

In [8]:
```
train_dataset = train.flow_from_directory(r'D:\Data Science with AI\14th-feb-2024\image classification\training',
                                          target_size = (200,200),
                                          batch_size = 3,
                                          class_mode = 'binary')
validataion_dataset = validataion.flow_from_directory(r'D:\Data Science with AI\14th-feb-2024\image classification\validatio
                                           target_size = (200,200),
                                           batch_size = 3,
                                           class_mode = 'binary')
```

```
Found 20 images belonging to 2 classes.
Found 0 images belonging to 2 classes.
```

In [9]:
```
train_dataset.class_indices
```

Out[9]:
```
{'happy': 0, 'not happy': 1}
```

```
In [10]: train_dataset.classes
```

```
Out[10]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])
```

```
In [11]: # 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')
                                             ]
                                             )
```

```
WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python311\site-packages\keras\src\backend.py:873: Th
e name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python311\site-packages\keras\src\layers\pooling\max
_pooling2d.py:161: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.
```

```
In [12]: 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.o
ptimizers.legacy.RMSprop.
```

In [13]:

```
model_fit = model.fit(train_dataset,
                     steps_per_epoch = 3,
                     epochs = 30,
                     validation_data = validataion_dataset)
```

```
Epoch 1/30
WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python311\site-packages\keras\src\utils\tf_utils.py:
492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.


WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python311\site-packages\keras\src\utils\tf_utils.py:
492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.


WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\base_layer_
utils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outs
ide_functions instead.


WARNING:tensorflow:From C:\Users\Achal Raghorte\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\base_layer_
utils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outs
ide_functions instead.


3/3 [==============================] - 1s 100ms/step - loss: 3.5043 - accuracy: 0.4444
Epoch 2/30
3/3 [==============================] - 0s 99ms/step - loss: 0.9455 - accuracy: 0.3333
Epoch 3/30
3/3 [==============================] - 0s 110ms/step - loss: 0.7691 - accuracy: 0.2500
Epoch 4/30
3/3 [==============================] - 0s 103ms/step - loss: 0.8187 - accuracy: 0.5000
Epoch 5/30
3/3 [==============================] - 0s 100ms/step - loss: 0.6402 - accuracy: 0.5556
Epoch 6/30
3/3 [==============================] - 0s 99ms/step - loss: 0.6939 - accuracy: 0.8750
Epoch 7/30
3/3 [==============================] - 0s 104ms/step - loss: 0.5783 - accuracy: 0.8889
Epoch 8/30
3/3 [==============================] - 0s 104ms/step - loss: 0.7239 - accuracy: 0.7778
Epoch 9/30
3/3 [==============================] - 0s 102ms/step - loss: 1.5134 - accuracy: 0.5556
Epoch 10/30
3/3 [==============================] - 0s 100ms/step - loss: 0.4788 - accuracy: 0.8889
Epoch 11/30
3/3 [==============================] - 0s 98ms/step - loss: 0.4897 - accuracy: 0.7500
Epoch 12/30
3/3 [==============================] - 0s 101ms/step - loss: 0.2246 - accuracy: 1.0000
Epoch 13/30
3/3 [==============================] - 0s 103ms/step - loss: 1.2213 - accuracy: 0.5000
Epoch 14/30
3/3 [==============================] - 0s 107ms/step - loss: 0.4085 - accuracy: 1.0000
Epoch 15/30
3/3 [==============================] - 0s 105ms/step - loss: 0.2511 - accuracy: 1.0000
Epoch 16/30
3/3 [==============================] - 0s 93ms/step - loss: 0.0758 - accuracy: 1.0000
Epoch 17/30
3/3 [==============================] - 0s 103ms/step - loss: 0.4287 - accuracy: 0.5556
Epoch 18/30
3/3 [==============================] - 0s 92ms/step - loss: 0.1012 - accuracy: 1.0000
Epoch 19/30
3/3 [==============================] - 0s 95ms/step - loss: 0.0701 - accuracy: 1.0000
Epoch 20/30
3/3 [==============================] - 0s 100ms/step - loss: 0.0276 - accuracy: 1.0000
Epoch 21/30
3/3 [==============================] - 0s 104ms/step - loss: 0.1839 - accuracy: 0.8889
Epoch 22/30
3/3 [==============================] - 0s 101ms/step - loss: 0.0725 - accuracy: 1.0000
Epoch 23/30
3/3 [==============================] - 0s 110ms/step - loss: 0.0146 - accuracy: 1.0000
Epoch 24/30
3/3 [==============================] - 0s 102ms/step - loss: 0.0099 - accuracy: 1.0000
Epoch 25/30
3/3 [==============================] - 0s 104ms/step - loss: 0.0030 - accuracy: 1.0000
Epoch 26/30
3/3 [==============================] - 0s 107ms/step - loss: 0.0021 - accuracy: 1.0000
Epoch 27/30
3/3 [==============================] - 0s 102ms/step - loss: 0.0014 - accuracy: 1.0000
Epoch 28/30
3/3 [==============================] - 0s 102ms/step - loss: 0.0154 - accuracy: 1.0000
Epoch 29/30
3/3 [==============================] - 0s 98ms/step - loss: 0.0023 - accuracy: 1.0000
Epoch 30/30
3/3 [==============================] - 0s 98ms/step - loss: 0.0108 - accuracy: 1.0000
```

```
In [25]: dir_path = r'D:\Data Science with AI\14th-feb-2024\image classification\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.jpeg
10.jpeg
11.jpeg
12.jpeg
13.jpeg
14.jpeg
14.jpg
15.jpeg
16.jpeg
17.jpeg
18.jpeg
19.jpeg
2.jpeg
3.jpeg
4.jpeg
5.jpeg
6.jpeg
7.jpeg
8.jpeg
9.png
```

```
In [26]: dir_path = r'D:\Data Science with AI\14th-feb-2024\image classification\testing'
         for i in os.listdir(dir_path ):
             img = image.load_img(dir_path+ '//'+i, target_size = (200,200))
             plt.imshow(img)
             plt.show()
```

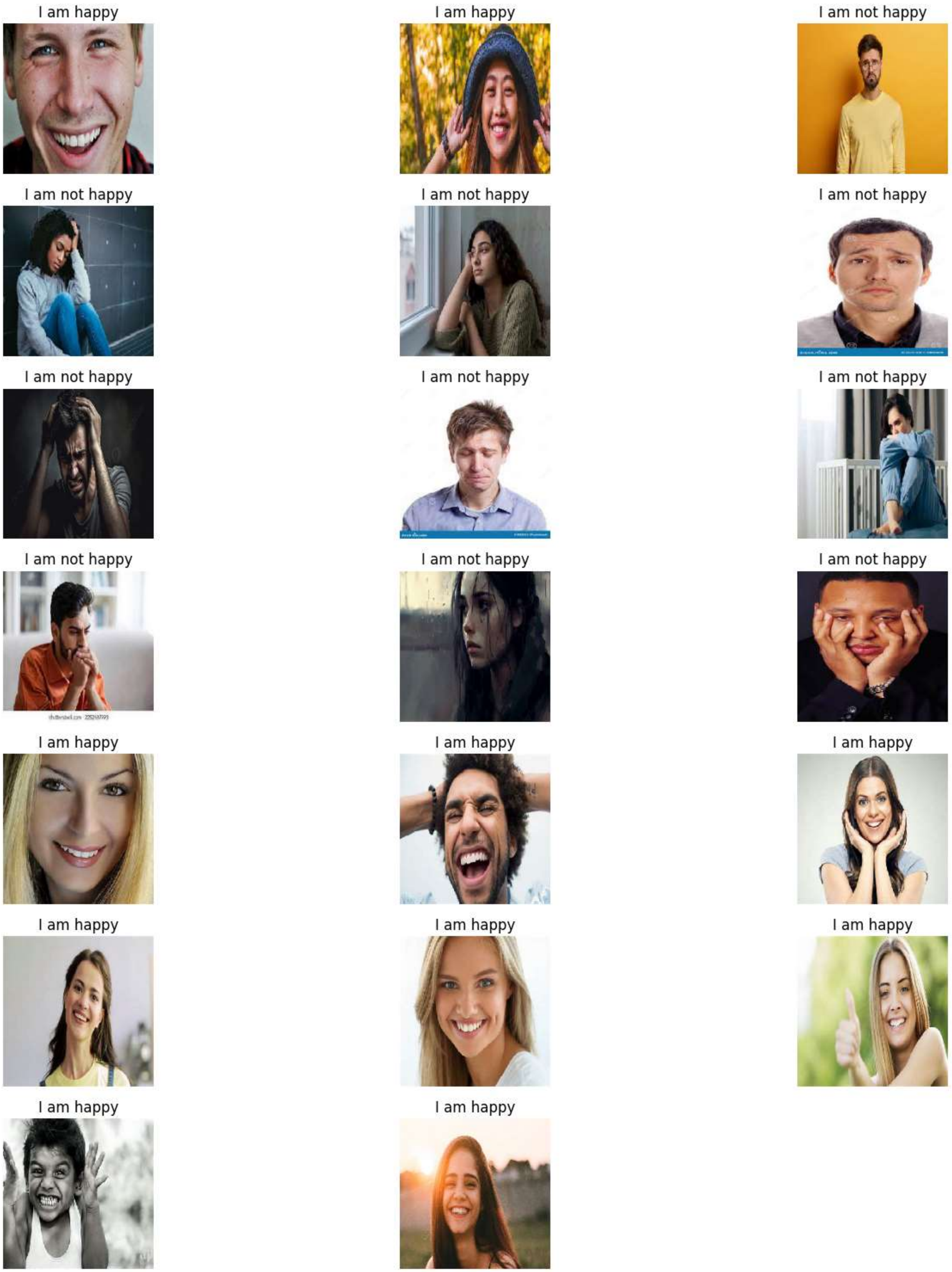
In [29]:

```
dir_path = r'D:\Data Science with AI\14th-feb-2024\image classification\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')
```

In [30]:

```
dir_path = r'D:\Data Science with AI\14th-feb-2024\image classification\testing'


plt.figure(figsize=(15, 15))
columns = 3
rows = len(os.listdir(dir_path)) // columns + 1

for i, filename in enumerate(os.listdir(dir_path)):
    img_path = os.path.join(dir_path, filename)
    img = image.load_img(img_path, target_size=(200, 200))
    plt.subplot(rows, columns, i + 1)
    plt.imshow(img)
    plt.axis('off')  # Disable axis

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

    val = model.predict(images)
    if val == 0:
        prediction = 'I am happy'
    else:
        prediction = 'I am not happy'
    plt.title(prediction)

plt.tight_layout()
plt.show()
```

```
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 41ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 37ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 29ms/step
```



I am happy
I am happy
I am not happy
I am not happy
I am not happy
I am not happy
I am not happy
I am not happy
I am not happy
I am not happy
I am not happy
I am not happy
I am happy
I am happy
I am happy
I am happy
I am happy
I am happy
I am happy
I am happy

```
In [ ]:
```