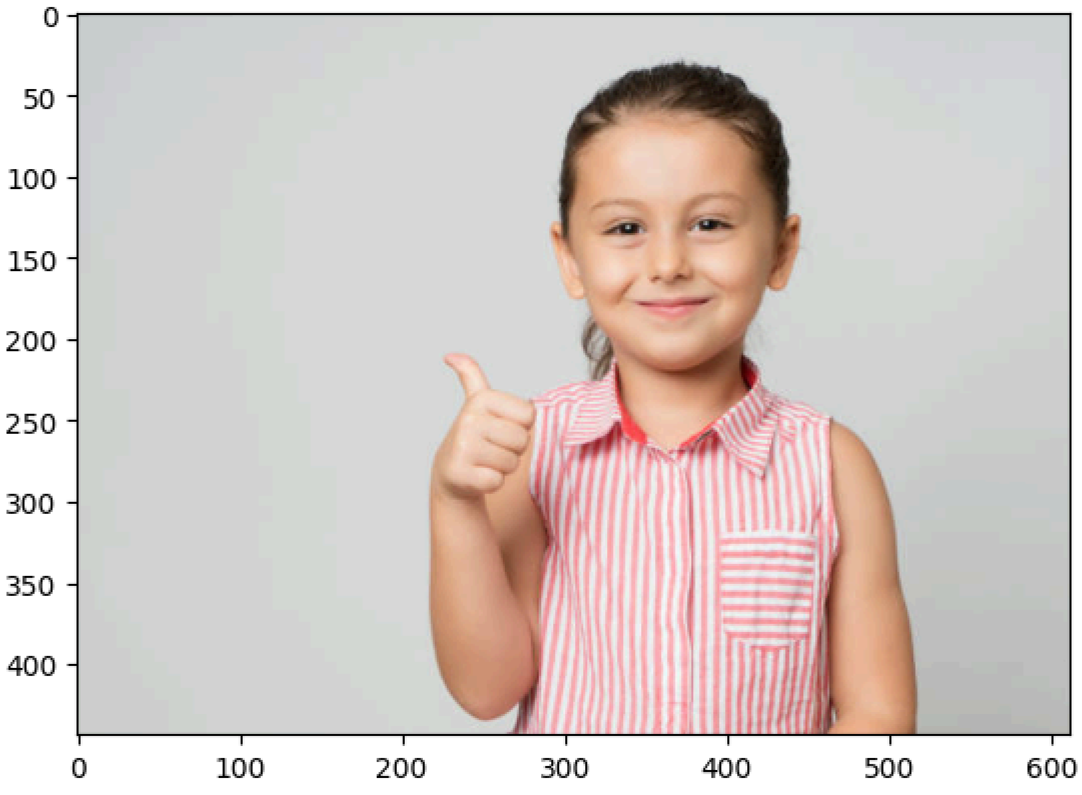
In [1]:
```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import cv2
import os
#image data generator is the package to lable the images & it will automatically
```

In [2]:
```
img=image.load_img(r'C:\Ds & AI ( my work)\AVSCODE\CNN - Happy or Sad\training\H
```

In [3]:
```
plt.imshow(img)
```

Out[3]:
```
<matplotlib.image.AxesImage at 0x26eb666e390>
```



In [4]:
```
i1= cv2.imread(r'C:\Ds & AI ( my work)\AVSCODE\CNN - Happy or Sad\training\Happy
i1
```

```
Out[4]:  array([[[206, 205, 201],
                 [206, 205, 201],
                 [206, 205, 201],
                 ...,
                 [213, 212, 208],
                 [213, 212, 208],
                 [213, 212, 208]],

                [[206, 205, 201],
                 [206, 205, 201],
                 [206, 205, 201],
                 ...,
                 [213, 212, 208],
                 [213, 212, 208],
                 [213, 212, 208]],

                [[206, 205, 201],
                 [206, 205, 201],
                 [206, 205, 201],
                 ...,
                 [213, 212, 208],
                 [213, 212, 208],
                 [213, 212, 208]],

                ...,

                [[206, 207, 205],
                 [206, 207, 205],
                 [206, 207, 205],
                 ...,
                 [184, 186, 186],
                 [184, 186, 186],
                 [184, 186, 186]],

                [[206, 207, 205],
                 [206, 207, 205],
                 [206, 207, 205],
                 ...,
                 [184, 186, 186],
                 [184, 186, 186],
                 [184, 186, 186]],

                [[206, 207, 205],
                 [206, 207, 205],
                 [206, 207, 205],
                 ...,
                 [184, 186, 186],
                 [184, 186, 186],
                 [184, 186, 186]]], shape=(444, 612, 3), dtype=uint8)
```

```
In [5]:  i1.shape   # shape of the image- heigjt, weight,rgb
```

```
Out[5]:  (444, 612, 3)
```

```
In [6]:  train=ImageDataGenerator(rescale=1/200)
         validation=ImageDataGenerator(rescale=1/200)     # resize the image using 200,20
```

```
In [7]:  train_dataset=train.flow_from_directory(r'C:\Ds & AI ( my work)\AVSCODE\CNN - Ha
                                                 target_size=(200,200),
                                                 batch_size=3,
                                                 class_mode='binary')
         validation_dataset=validation.flow_from_directory(r'C:\Ds & AI ( my work)\AVSCOD
                                                           target_size=(200,200),
                                                           batch_size=3,
                                                           class_mode='binary')
```

```
Found 6 images belonging to 2 classes.
Found 0 images belonging to 2 classes.
```

In [8]:
```
train_dataset.class_indices
```

Out[8]:
```
{'Happy': 0, 'Sad': 1}
```

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

Out[9]:
```
array([0, 0, 1, 1, 1, 1], dtype=int32)
```

In [10]:
```
# now we are applying maxpooling
model = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(16, (3,3), activation='relu', input_shape=(200, 200,
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.summary()
```

```
C:\Users\91918\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\co
nvolutional\base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_di
m` argument to a layer. When using Sequential models, prefer using an `Input(shap
e)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Model: "sequential"

Layer (type)	Output Shape
conv2d (Conv2D)	(None, 198, 198, 16)
max_pooling2d (MaxPooling2D)	(None, 99, 99, 16)
conv2d_1 (Conv2D)	(None, 97, 97, 32)
max_pooling2d_1 (MaxPooling2D)	(None, 48, 48, 32)
conv2d_2 (Conv2D)	(None, 46, 46, 64)
max_pooling2d_2 (MaxPooling2D)	(None, 23, 23, 64)
flatten (Flatten)	(None, 33856)
dense (Dense)	(None, 512)
dense_1 (Dense)	(None, 1)

```
Total params: 17,358,881 (66.22 MB)
Trainable params: 17,358,881 (66.22 MB)
Non-trainable params: 0 (0.00 B)
```

In [11]:
```
import tensorflow as tf
print(tf.__version__)
```

```
2.20.0
```

In [12]:
```
model.compile(loss='binary_crossentropy',
             optimizer = tf.keras.optimizers.RMSprop(learning_rate = 0.001),
             metrics = ['accuracy']
             )
```

In [13]:
```
model_fit = model.fit(train_dataset,epochs = 15)
```

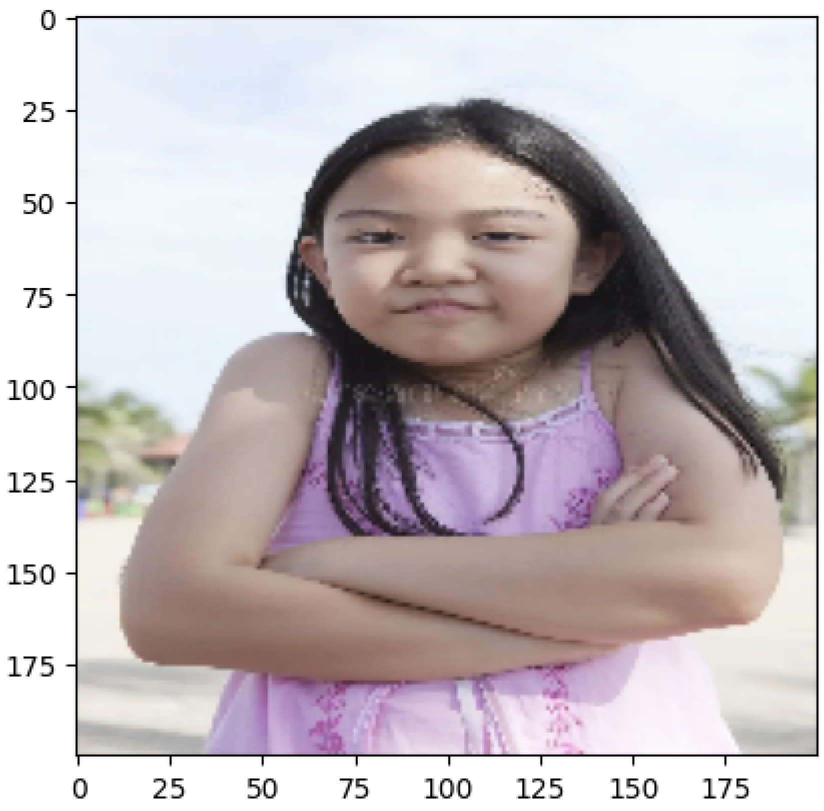
```
Epoch 1/15
C:\Users\91918\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers
\data_adapters\py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class sho
uld call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include
`workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments t
o `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
```

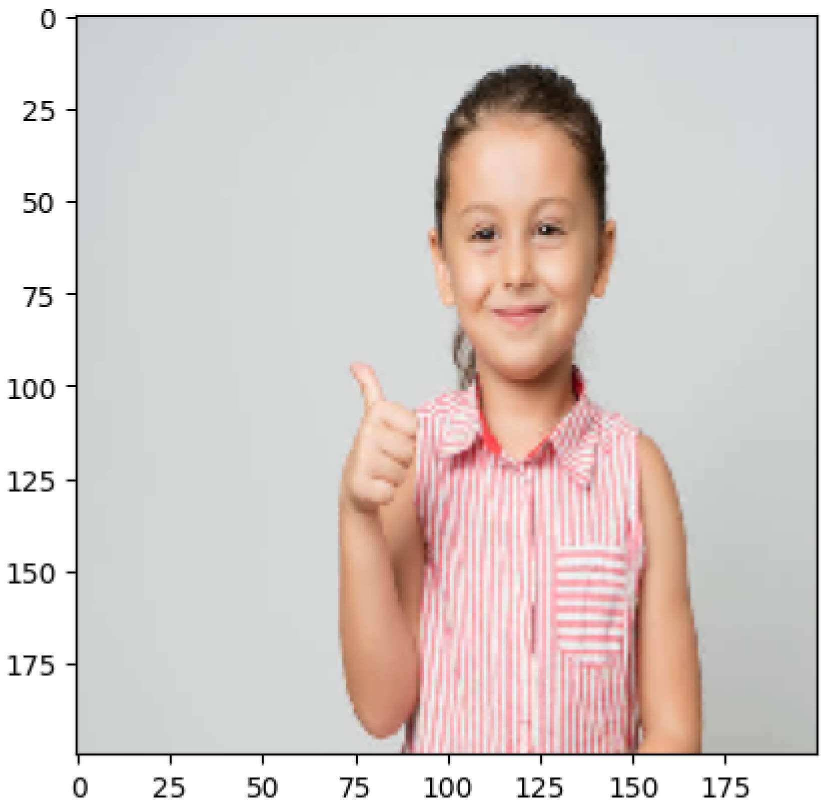
```
2/2 ━━━━━━━━━━━━━━━━━━━━ 3s 237ms/step - accuracy: 0.3333 - loss: 25.6950
Epoch 2/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 221ms/step - accuracy: 0.6667 - loss: 3.0745
Epoch 3/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 256ms/step - accuracy: 0.6667 - loss: 1.2100
Epoch 4/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 255ms/step - accuracy: 0.6667 - loss: 0.5879
Epoch 5/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 277ms/step - accuracy: 0.6667 - loss: 1.0606
Epoch 6/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 295ms/step - accuracy: 0.6667 - loss: 0.5021
Epoch 7/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 232ms/step - accuracy: 1.0000 - loss: 0.4044
Epoch 8/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 244ms/step - accuracy: 0.1667 - loss: 1.4280
Epoch 9/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 234ms/step - accuracy: 0.8333 - loss: 0.3436
Epoch 10/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 234ms/step - accuracy: 1.0000 - loss: 0.1595
Epoch 11/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 221ms/step - accuracy: 1.0000 - loss: 0.1045
Epoch 12/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 225ms/step - accuracy: 1.0000 - loss: 0.0578
Epoch 13/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 218ms/step - accuracy: 1.0000 - loss: 0.0383
Epoch 14/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 223ms/step - accuracy: 1.0000 - loss: 0.0199
Epoch 15/15
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 220ms/step - accuracy: 1.0000 - loss: 0.0137
```

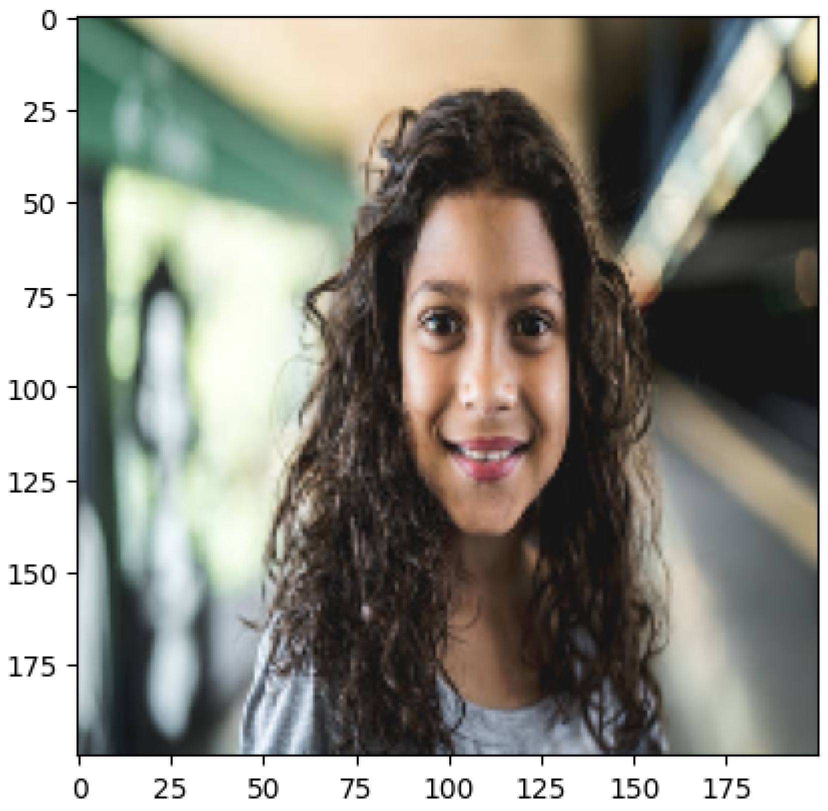
In [14]:
```python
dir_path = r'C:\Ds & AI ( my work)\AVSCODE\CNN - Happy or Sad\testing'
for i in os.listdir(dir_path ):
    print(i)
    #img = image.load_img(dir_path+ '//'+i, target_size = (200,200))
   # plt.imshow(img)
   # plt.show()
```

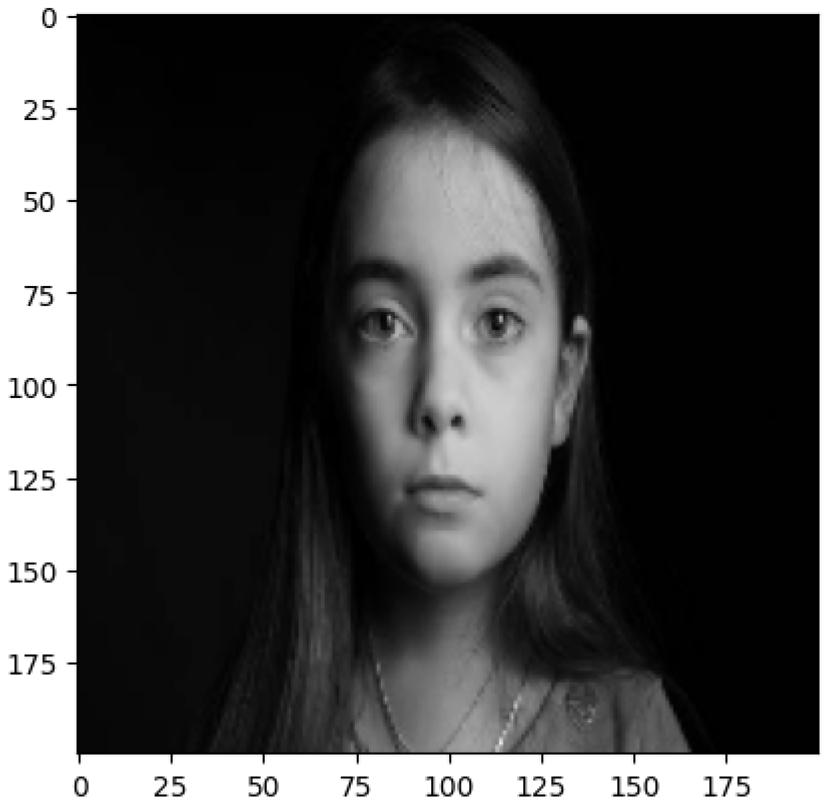
```
pic-3.webp
pic-4.jpg
pic-5.jpg
pic-6.jpg
pic-7.jpeg
pic-8.jpeg
pic-9.jpeg
```

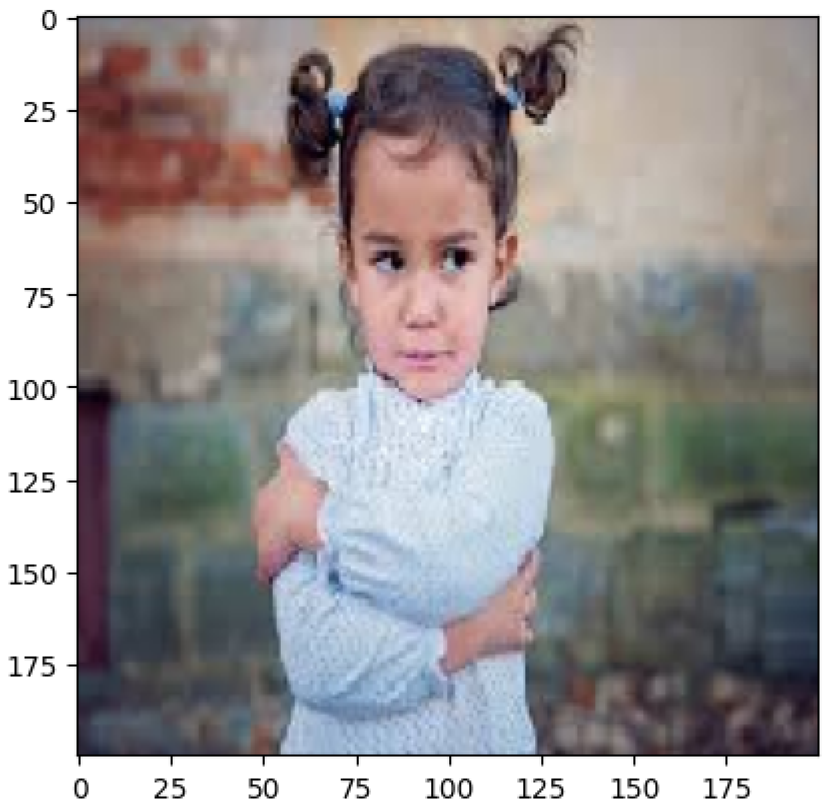
In [15]:
```python
dir_path = r'C:\Ds & AI ( my work)\AVSCODE\CNN - Happy or Sad\testing'
for i in os.listdir(dir_path ):
    img = image.load_img(dir_path+ '//'+i, target_size = (200,200))
    plt.imshow(img)
    plt.show()
```

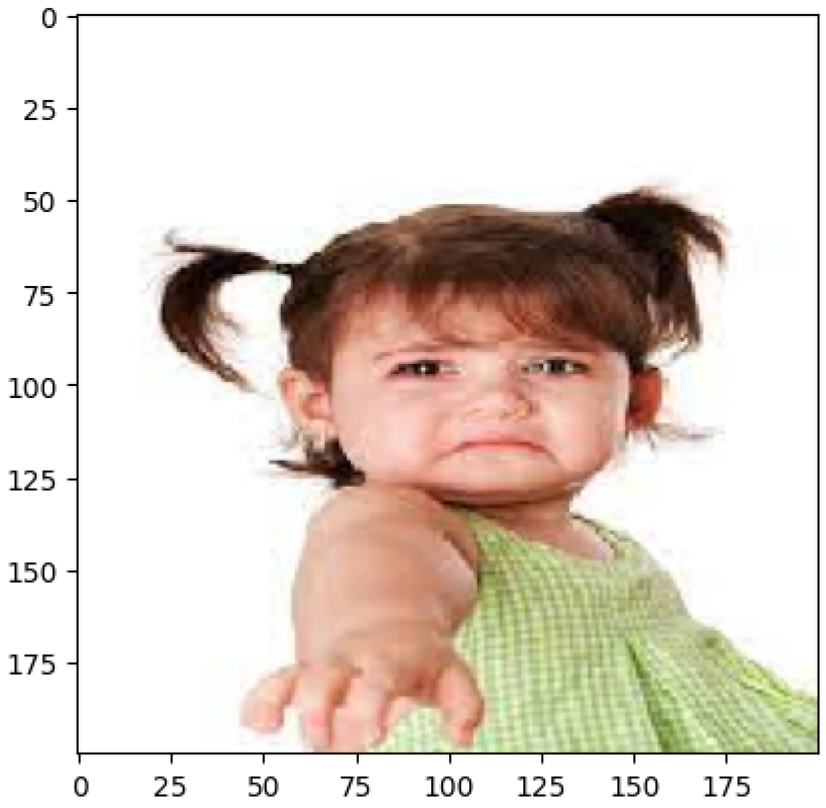

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175

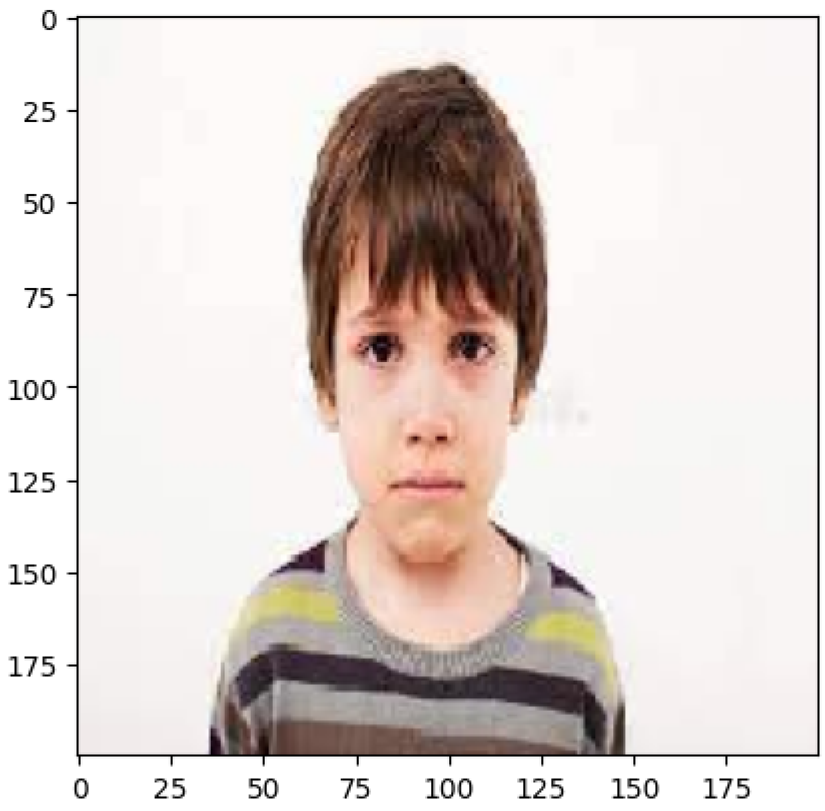

0
25
50
75
100
125
150
175
0
25
50
75
100
125
150
175


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



```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```