Gradient of Image

import library

In [3]:

```
import numpy as np
import matplotlib.image as img
import matplotlib.pyplot as plt
from matplotlib import cm
import matplotlib.colors as colors
```

load input image ('test.jpeg')

In [4]:

```
I0 = img.imread('test.jpeg')
```

check the size of the input image

In [5]:

```
# ++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#
num_row     = 512
num_column  = 510
num_channel = 3
#
# ++++++++++++++++++++++++++++++++++++++++++++++++++

print('number of rows of I0 = ', num_row)
print('number of columns of I0 = ', num_column)
print('number of channels of I0 = ', num_channel)
```

```
number of rows of I0 =  512
number of columns of I0 =  510
number of channels of I0 =  3
```

convert the color image into a grey image

- take the average of the input image with 3 channels with respect to the channels into an image with 1 channel

In [41]:

```
# ++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#

from matplotlib import pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread('test.jpeg')

R, G, B = img[:,:,0], img[:,:,1], img[:,:,2]

I = 0.2989 * R + 0.5870 * G + 0.1140 * B
num_row     = I.shape[0]
num_column  = I.shape[1]
```

```
#plt.imshow(imgGray, cmap='gray')
#plt.show()

#
# ++++++++++++++++++++++++++++++++++++++++++++++++++

print('number of rows of I = ', num_row)
print('number of columns of I = ', num_column)
```

```
number of rows of I =  510
number of columns of I =  512
```

normalize the converted image

- normalize the converted grey scale image so that its maximum value is 1 and its minimum value is 0

In [44]:

```
# ++++++++++++++++++++++++++++++++++++++++++++++++++
# complete the blanks
#

I=I/np.max(I)

#
# ++++++++++++++++++++++++++++++++++++++++++++++++++

print('maximum value of I = ', np.max(I))
print('minimum value of I = ', np.min(I))
```

```
maximum value of I =  1.0
minimum value of I =  0.0
```

define a function to compute the derivative of input matrix in x(row)-direction

- forward difference : $I[x+1,y] - I[x,y]$

In [45]:

```
def compute_derivative_x_forward(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    dx=D.shape[0]

    for i in range(dx):
        if i !=dx-1:
            D[i,:]=I[i+1,:]-I[i,:]


    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return D
```

- backward difference : $I[x,y] - I[x-1,y]$

In [46]:

```
def compute_derivative_x_backward(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    dx=D.shape[0]

    for i in range(dx):
        if i !=0:
            D[i,:]=I[i,:]-I[i-1,:]

    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return D
```

- central difference : $\frac{1}{2}(I[x+1,y] - I[x-1,y])$

In [47]:

```
def compute_derivative_x_central(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    D=1/2*(compute_derivative_x_forward(I)+compute_derivative_x_backward(I))


    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return D
```

define a function to compute the derivative of input matrix in y(column)-direction

- forward difference : $I[x,y+1] - I[x,y]$

In [48]:

```
def compute_derivative_y_forward(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    dy=D.shape[0]

    for i in range(dy):
        if i !=dy-1:
            D[:,i]=I[:,i+1]-I[:,i]



    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    return D
```

- backward difference : $I[x, y] - I[x, y-1]$

In [49]:

```
def compute_derivative_y_backward(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    dy=D.shape[0]

    for i in range(dy):
        if i !=0:
            D[:,i]=I[:,i]-I[:,i-1]



    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return D
```

- central difference : $\frac{1}{2}(I[x, y+1] - I[x, y-1])$

In [50]:

```
def compute_derivative_y_central(I):

    D = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #

    D=1/2*(compute_derivative_y_forward(I)+compute_derivative_y_backward(I))


    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return D
```

compute the norm of the gradient of the input image

- L_2^2-norm of the gradient $\left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)$ is defined by $\left(\frac{\partial I}{\partial x}\right)^2 + \left(\frac{\partial I}{\partial y}\right)^2$

In [52]:

```
def compute_norm_gradient_central(I):

    norm_gradient = np.zeros(I.shape)

    # ++++++++++++++++++++++++++++++++++++++++++++++++++
    # complete the blanks
    #
    norm_gradient = (compute_derivative_y_central(I))**2+(compute_derivative_x_centra
    #
    # ++++++++++++++++++++++++++++++++++++++++++++++++++

    return norm_gradient
```

---

functions for presenting the results

In [20]:

```
def function_result_01():

    plt.figure(figsize=(8,6))
    plt.imshow(I0)
    plt.show()
```

In [21]:

```
def function_result_02():

    plt.figure(figsize=(8,6))
    plt.imshow(I, cmap='gray', vmin=0, vmax=1, interpolation='none')
    plt.show()
```

In [22]:

```
def function_result_03():

    D = compute_derivative_x_forward(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [23]:

```
def function_result_04():

    D = compute_derivative_x_backward(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [24]:

```
def function_result_05():

    D = compute_derivative_x_central(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [25]:

```
def function_result_06():

    D = compute_derivative_y_forward(I)
    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [26]:

```
def function_result_07():

    D = compute_derivative_y_backward(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [27]:

```
def function_result_08():

    D = compute_derivative_y_central(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [28]:

```
def function_result_09():

    D = compute_norm_gradient_central(I)

    plt.figure(figsize=(8,6))
    plt.imshow(D, cmap='gray')
    plt.show()
```

In [29]:

```
def function_result_10():

    D = compute_norm_gradient_central(I)

    plt.figure(figsize=(8,6))
    im = plt.imshow(D, cmap=cm.jet, norm=colors.LogNorm())
    plt.colorbar(im)
    plt.show()
```

In [30]:

```
def function_result_11():

    D = compute_derivative_x_forward(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [31]:

```
def function_result_12():

    D = compute_derivative_x_backward(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [32]:

```
def function_result_13():

    D = compute_derivative_x_central(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [33]:

```
def function_result_14():

    D = compute_derivative_y_forward(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [34]:

```
def function_result_15():

    D = compute_derivative_y_backward(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [35]:

```
def function_result_16():

    D = compute_derivative_y_central(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]
    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

In [36]:

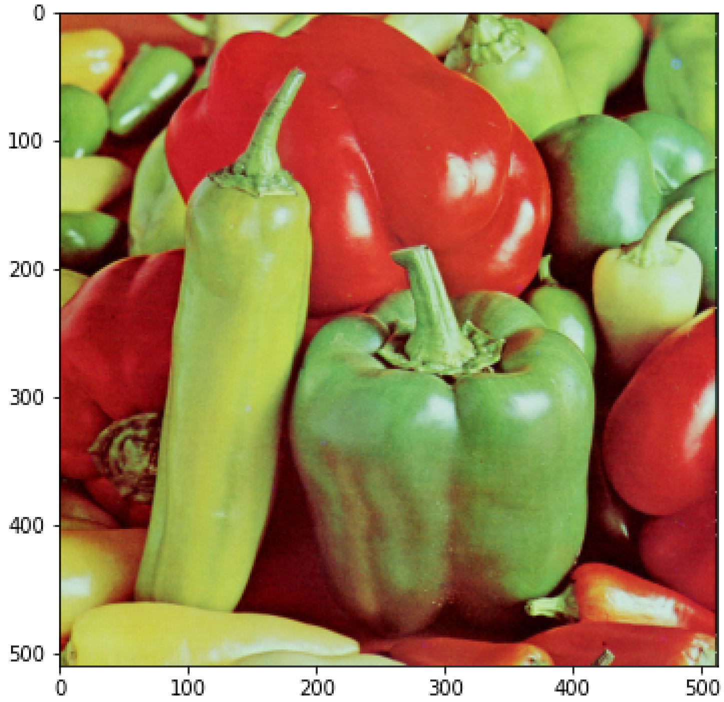
```
def function_result_17():

    D = compute_norm_gradient_central(I)

    value1 = D[0, 0]
    value2 = D[-1, -1]
    value3 = D[100, 100]
    value4 = D[200, 200]

    print('value1 = ', value1)
    print('value2 = ', value2)
    print('value3 = ', value3)
    print('value4 = ', value4)
```

results

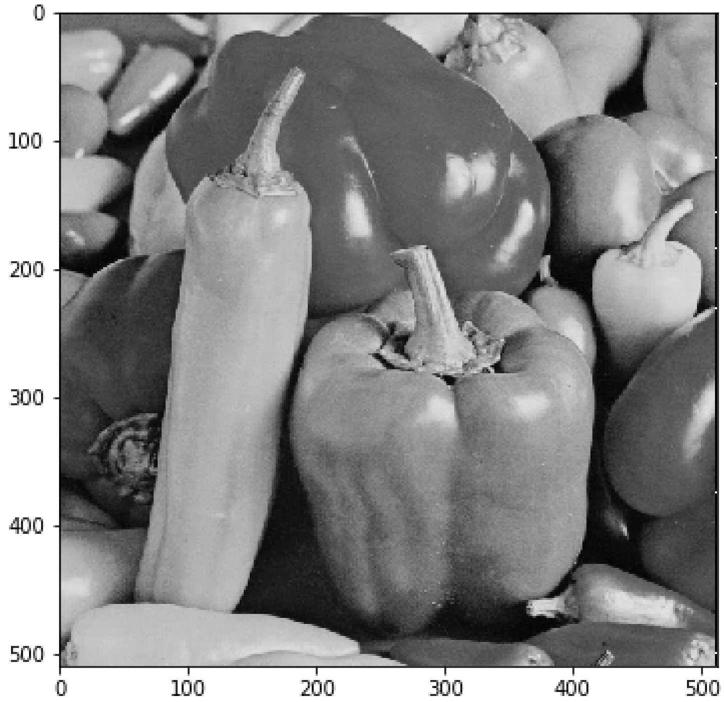
In [53]:

```
number_result = 17

for i in range(number_result):
    title = '## [RESULT {:02d}]'.format(i+1)
    name_function = 'function_result_{:02d}()'.format(i+1)

    print('**************************************************')
    print(title)
    print('**************************************************')
    eval(name_function)
```

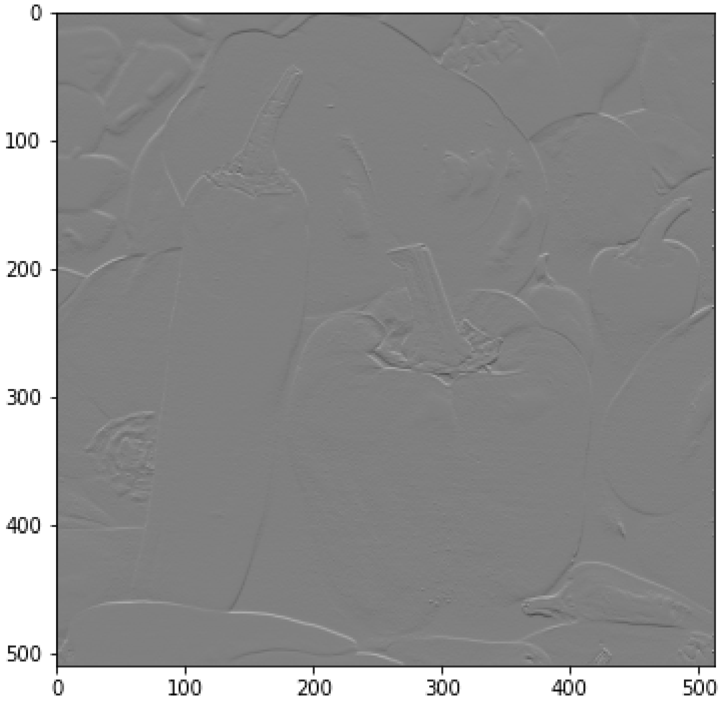
```
**************************************************
## [RESULT 01]
**************************************************
```



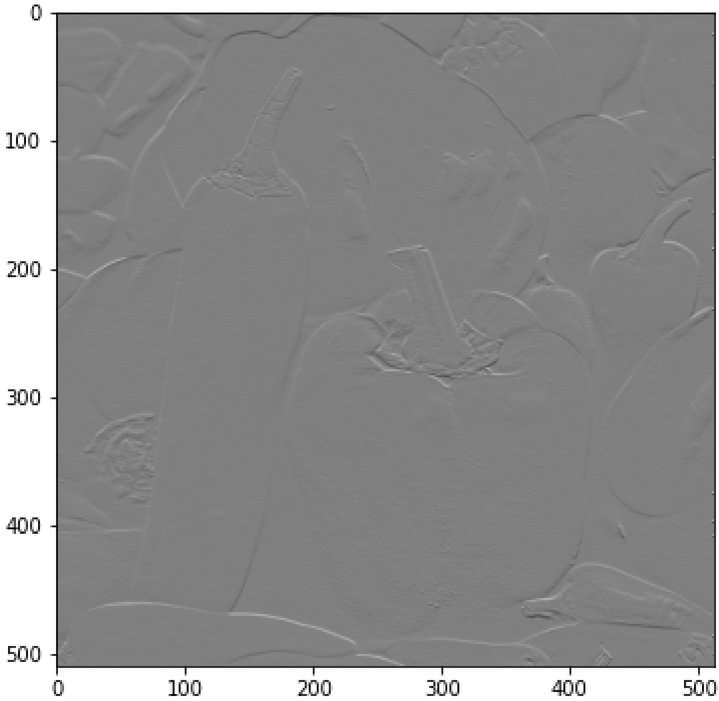
```
**************************************************
## [RESULT 02]
**************************************************
```



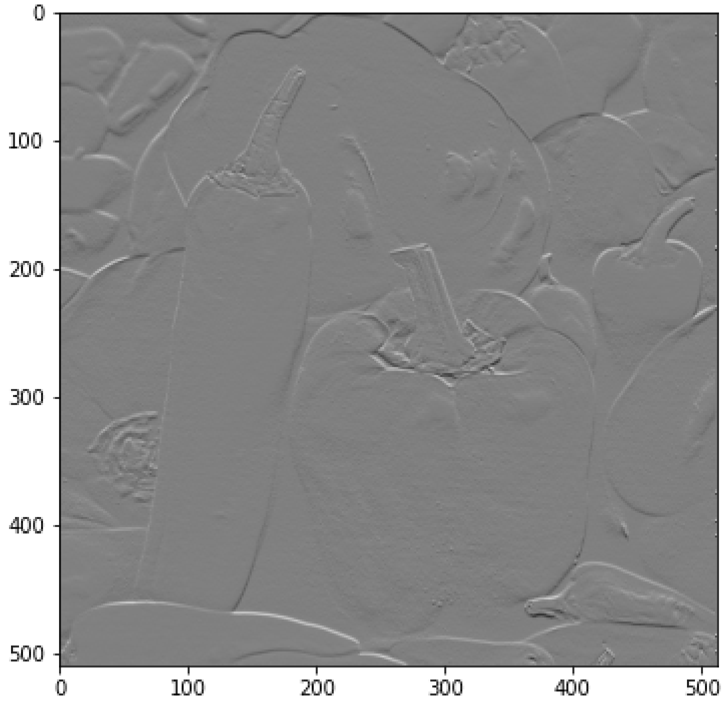
```
**************************************************
## [RESULT 03]
**************************************************
```



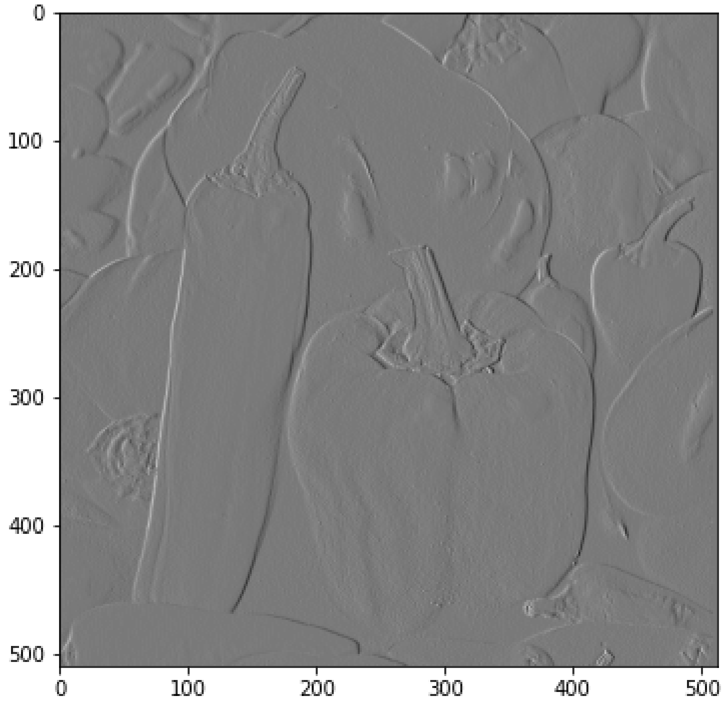
```
**************************************************
## [RESULT 04]
**************************************************
```



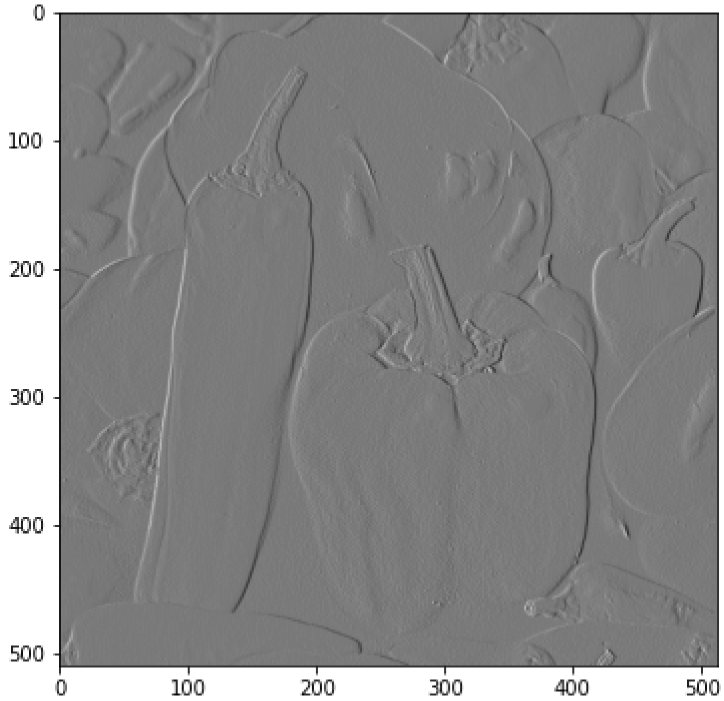
```
**************************************************
## [RESULT 05]
**************************************************
```



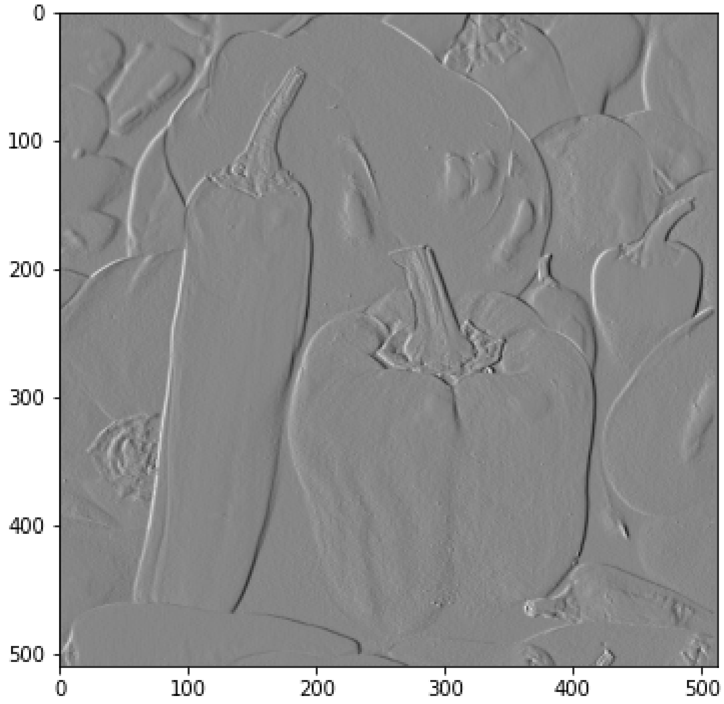
```
**************************************************
## [RESULT 06]
**************************************************
```



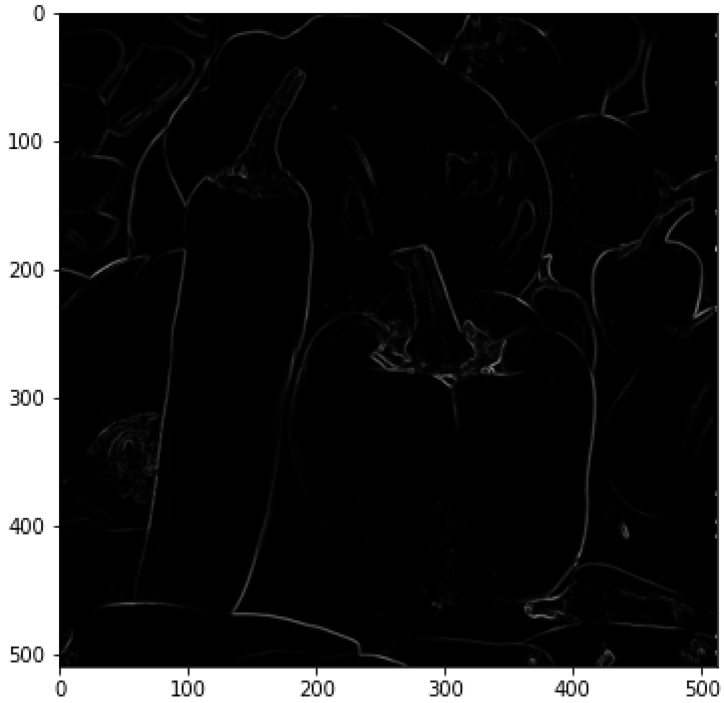
```
**************************************************
## [RESULT 07]
**************************************************
```



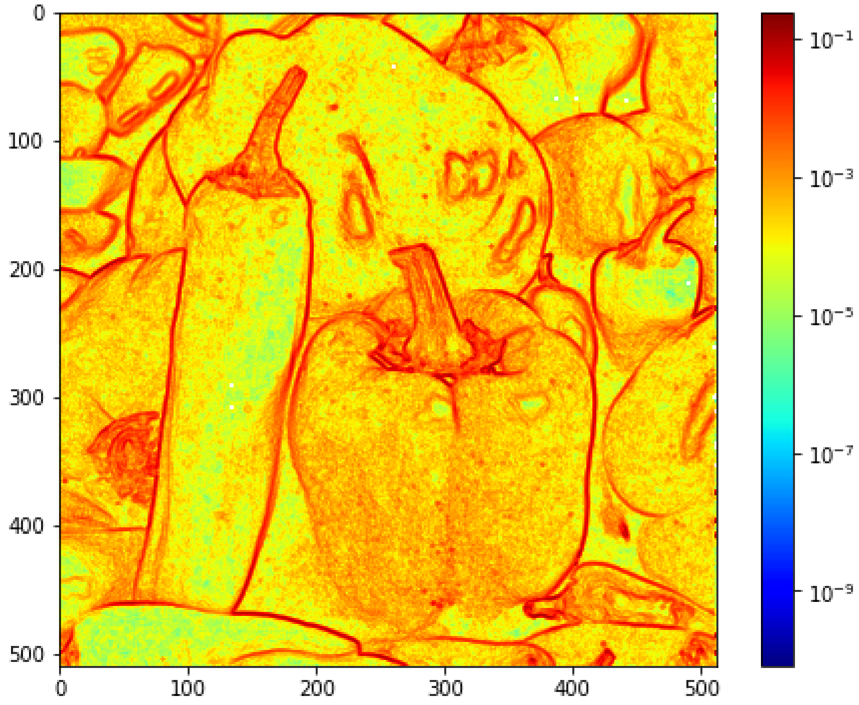
```
************************************************
## [RESULT 08]
************************************************
```



```
************************************************
## [RESULT 09]
************************************************
```



```
**************************************************
## [RESULT 10]
**************************************************
```



```
**************************************************
## [RESULT 11]
**************************************************
value1 =  -0.00970690470722857
value2 =  0.0
value3 =  0.00450604816945471З
value4 =  0.013631983167580242
**************************************************
## [RESULT 12]
**************************************************
value1 =  0.0
value2 =  0.002346646568255109
value3 =  0.016831242038654104
value4 =  -0.01254345005721369
**************************************************
## [RESULT 13]
```

```
**************************************************
value1 =  -0.004853452353614285
value2 =  0.0011733232841275546
value3 =  0.010668645104054408
value4 =  0.0005442665551832759
**************************************************
## [RESULT 14]
**************************************************
value1 =  -0.03718245392969094
value2 =  0.0
value3 =  -0.004281511227828494
value4 =  0.002177458767833873
**************************************************
## [RESULT 15]
**************************************************
value1 =  0.0
value2 =  0.0
value3 =  0.001275778077422074
value4 =  0.013741111261587502
**************************************************
## [RESULT 16]
**************************************************
value1 =  -0.01859122696484547
value2 =  0.0
value3 =  -0.00150286657520321
value4 =  0.007959285014710688
**************************************************
## [RESULT 17]
**************************************************
value1 =  0.0003691897198072014
value2 =  1.3766875290758703e-06
value3 =  0.00011607859629912712
value4 =  6.364644402848918e-05
```

```
In [ ]:
```