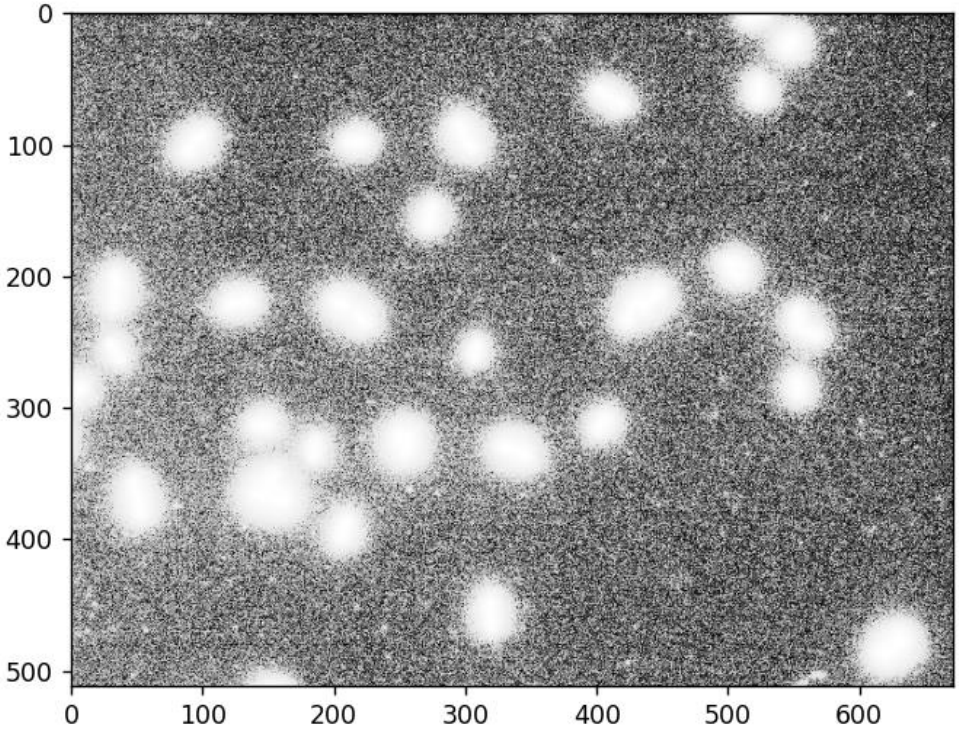
```python
import numpy
from scipy import misc
import matplotlib.pyplot as plt
import copy
from imageio import imread
import numpy as np

# Implement This Function

def get_hist (pic,M,N):
    hist_out = np.zeros(65535)
    for i in range(M):
        for j in range(N):
            itr = pic[i][j]
            hist_out[itr] +=1
    return hist_out

def get_cdf(a):
    cdf = np.zeros(65535)
    result = 0
    for i in range(len(a)):
        result += a[i]
        cdf[i] = result
    return cdf

def get_h(cdf,M,N):
    h = np.zeros(65535)
    cdf_min = 0
    for i in range (len(cdf)):
        if not(cdf[i] == 0):
            cdf_min = 0
            break
    for j in range(len(cdf)):
        result = int(((cdf[j]-cdf_min)/(M*N+1))*(65535-1))
        h[j] = result
    return h

```

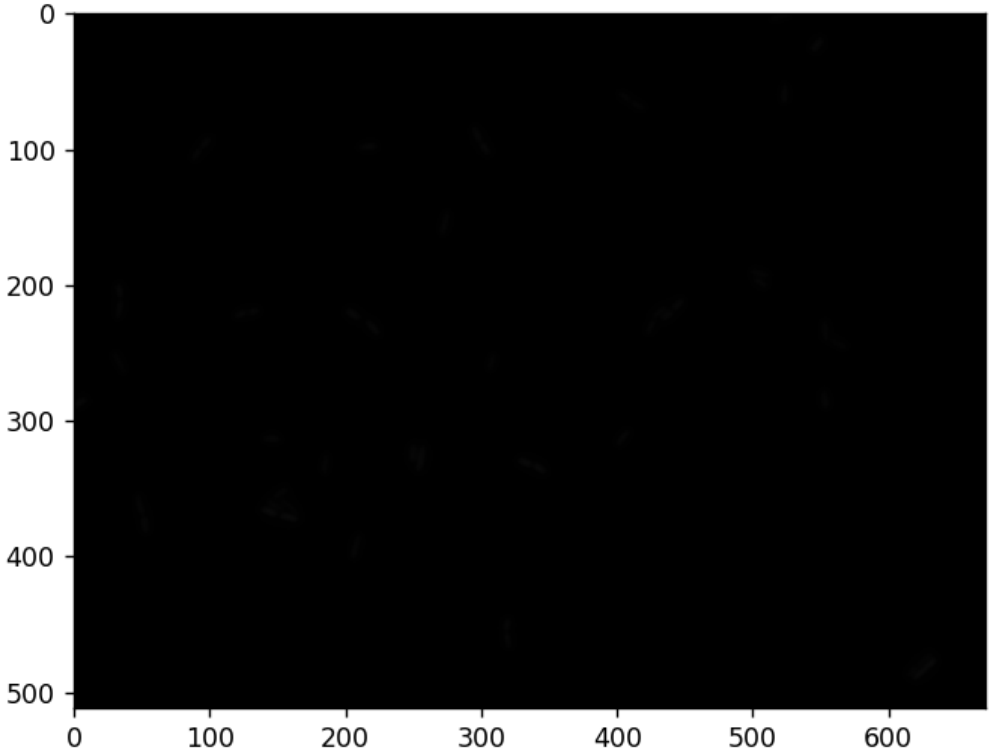
```
41
42  def histeq(pic):
43      # Follow the procedures of Histogram Equalizaion
44      # Modify the pixel value of pic directly
45      [M,N] = pic.shape
46      hist_cnt = get_hist(pic,M,N)
47      print ("hist generation complete")
48
49      cdf = get_cdf(hist_cnt)
50      print ("cdf generation complete")
51
52      h = get_h(cdf,M,N)
53      print ("h generation complete")
54
55
56      for k in range(M):
57          for l in range(N):
58              itr = pic[k][l]
59              pic[k][l] = h[itr]
60
61
62      return pic
63
64  # Histogram Equilization
65  eco_origin = imread('eco.tif')
66  eco_histeq = copy.deepcopy(eco_origin)
67  # Call to histeq to perform Histogram Equilization
68  eco_histeq = histeq(eco_histeq)
69  # Show the result in two windows
70  fig_eco_origin = plt.figure(1)
71  fig_eco_origin.suptitle('Original eco.tif', fontsize=14, fontweight='bold')
72  plt.imshow(eco_origin,cmap='gray',vmin = 0, vmax = 65535)
73  fig_eco_histeq = plt.figure(2)
74  fig_eco_histeq.suptitle('Histrogram Equalized eco.tif', fontsize=14, fontweight='bold')
75  plt.imshow(eco_histeq,cmap='gray',vmin = 0, vmax = 65535)
76  plt.show()
```

Question : the histogram is of size 256, with each index representing one brightness, and it would be dtype = int, as it increments at steps of 1

Histrogram Equalized eco.tif



Original eco.tif



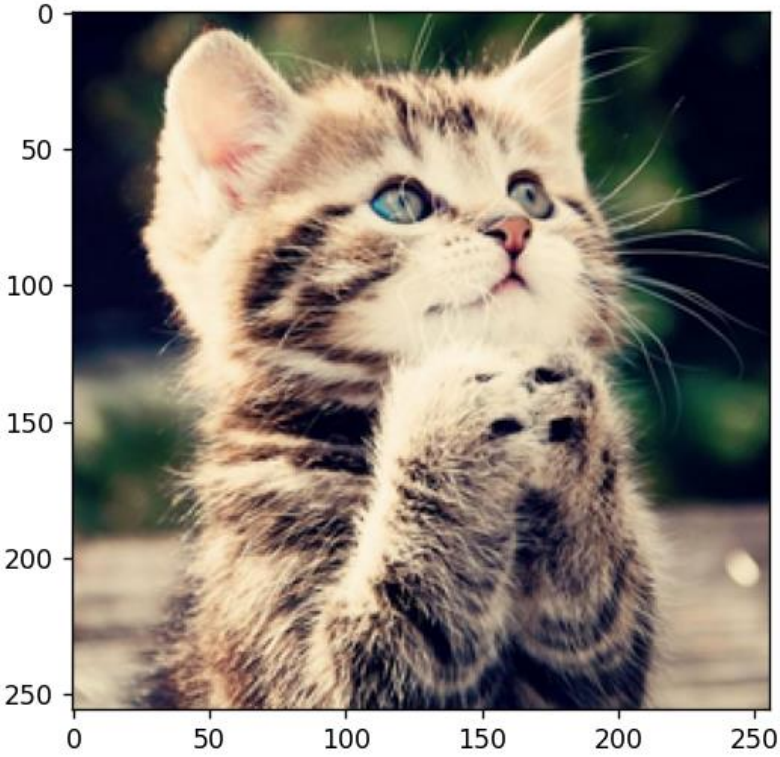
```python
import numpy
from scipy import misc
import matplotlib.pyplot as plt
from imageio import imread
import numpy as np


# Function Definition Here

def conv2_1pix(pic_,kernel,i,j,M,N):
    value_out = 0
    [l,w] = kernel.shape
    off_l = int((l-1)/2)
    off_w = int((w-1)/2)
    for x in range(l):
        for y in range(w):
            if (i-off_l+x<0) or (j-off_w+y<0) or (i-off_l+x>=M) or (j-off_w+y >= N):
                pic_data=0
            else:
                #print (i-off_l+x,j-off_w+y)
                pic_data = pic_[i-off_l+x][j-off_w+y]
            pic_data = float(pic_data)
            value_out += pic_data * kernel[x][y]
            #print (np.uint8(value_out))
    return int(value_out)

# Implement This funtion
def conv2(pic,kernel):
    # Create a new pic with same size but float type
    pic_conv = numpy.zeros(numpy.shape(pic), dtype=int)
    [M,N,T] = pic.shape
    print (M,N,T)
    print(kernel)
    # Perform 2-D Convolution with the given kernel
    for layer in range(3):
        pic_ = pic[:,:,layer]
        for i in range (M):
            for j in range (N):
                pic_conv[i][j][layer] = int(conv2_1pix(pic_,kernel,i,j,M,N))

        print ("layer ",layer, "completed")
        print (pic_)
        print (pic_conv[:,:,layer])
    return pic_conv

```

Question:

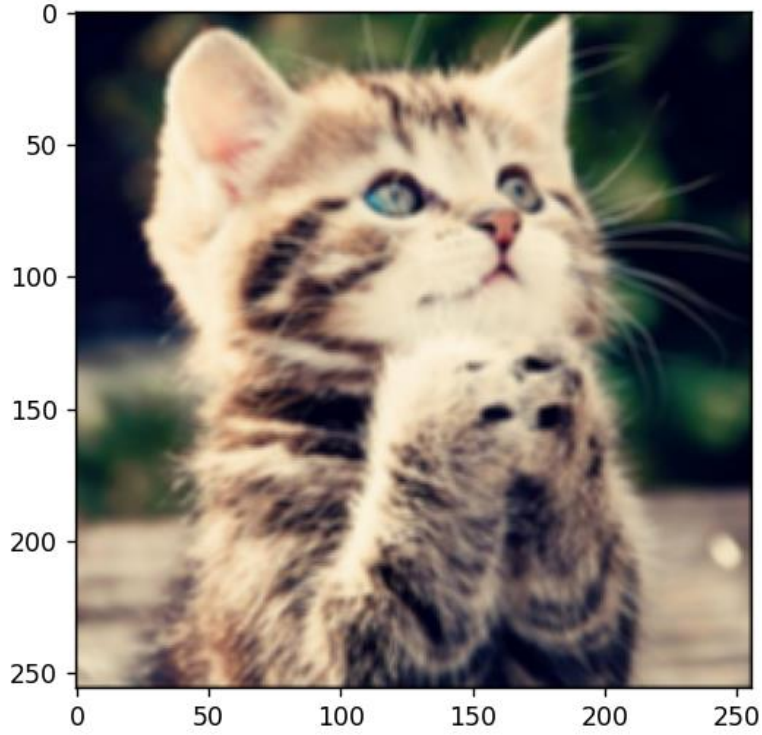
The histogram and cdf are stored as arrays, it would be more efficient to only store steps with non-zero value.

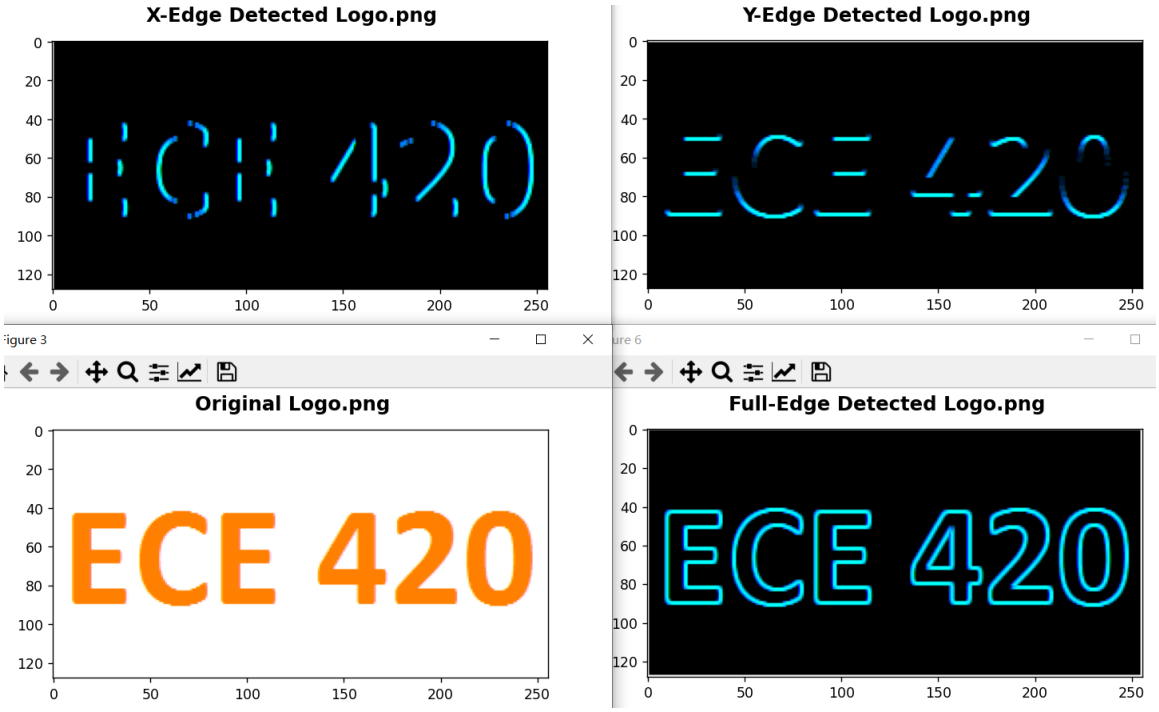
Normalizing CDF will make the pdf more uniform so dynamic range of the image will increase.

Original Kitten.png



Blurred Kitten.png




X-Edge Detected Logo.png
Y-Edge Detected Logo.png
Original Logo.png
Full-Edge Detected Logo.png
ECE 420
ECE 420