**Libraries Used:**

* from PIL import Image,ImageFilter
* import NumPy as np

**Task 1**

**Code: (For 3x3)**

resolution=im.size

original\_image= np.array(im)

new\_image=np.array(im)

filter = np.zeros((3,3))

#Code adds replication rows and columns to the image on the borders

#Should be added according to filter used

original\_image= np.concatenate([[original\_image[0]],original\_image],axis=0)

original\_image=np.concatenate([original\_image,[original\_image[resolution[1]-1]]],axis=0)

original\_image=np.transpose(original\_image)

original\_image=np.concatenate([[original\_image[0]],original\_image],axis=0)

original\_image=np.concatenate([original\_image,[original\_image[resolution[0]-1]]],axis=0)

original\_image=np.transpose(original\_image)

#---------------------------------------------------------------

print("Resolution:",resolution)

sum = 0

for x in range(1,resolution[1]-1):

for y in range(1,resolution[0]-1):

for i in range(x-7,x+8): #change the values of in this case 7,8 according to the filter (15x15)

for j in range(y-7,y+8):#change the values of in this case 7,8 according to the filter (15x15)

sum= sum+int(original\_image[i][j]) #looping to calculating sum of filter

new\_image[x][y]=(sum/225) #new resultant image intensities are assigned

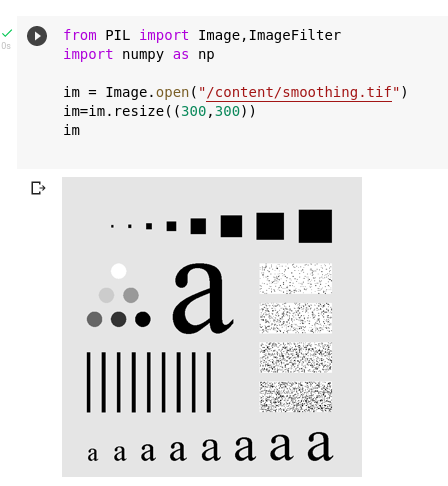
sum=0 #reinitializing the sum variable

#Display Message Prompt

print("Image Processed...Successfully.")

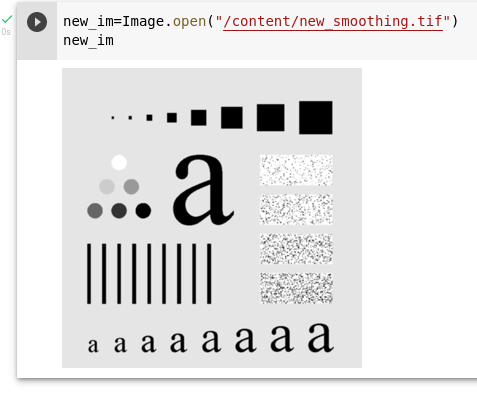
print("")

**Original image:**



**Processed Image:**

**3x3 Filter**



**5x5 Filter**



**15x15 Filter**



**35x35 Filter**



**Findings:**

* By increasing the size of the filter, the image gets blurred.
* As the filter size increases the image gets more and more blurred.

**Why?**

As the size of the box filter increases more intensities are weighted to form the resultant intensity pixel. Hence, the details of the image are lost with the use of a filter that has a larger NxN size.

**Weighted Averaging Filter**

**Code:**

resolution=im.size

original\_image= np.array(im)

new\_image=np.array(im)

filter = np.zeros((3,3))

#Code to add replication row and column to the image on the borders

original\_image= np.concatenate([[original\_image[0]],original\_image],axis=0)

original\_image=np.concatenate([original\_image,[original\_image[resolution[1]-1]]],axis=0)

original\_image=np.transpose(original\_image)

original\_image=np.concatenate([[original\_image[0]],original\_image],axis=0)

original\_image=np.concatenate([original\_image,[original\_image[resolution[0]-1]]],axis=0)

original\_image=np.transpose(original\_image)

#---------------------------------------------------------------

print("Resolution:",resolution)

sum = 0

for x in range(1,resolution[1]-1):

for y in range(1,resolution[0]-1):

for i in range(x-1,x+2): #change the values of in this case 1,2 according to the filter (3x3)

if i==x:

sum= sum+2\*(int(original\_image[i][y-1]))+4\*(int(original\_image[i][y]))+2\*(int(original\_image[i][y+1])) #looping to calculating sum of filter

else:

sum= sum+1\*(int(original\_image[i][y-1]))+2\*(int(original\_image[i][y]))+1\*(int(original\_image[i][y+1])) #looping to calculating sum of filter

new\_image[x][y]=(sum/16) #new resultant image intensities are assigned

sum=0 #reintializing the sum variable

#Display Message Prompt

print("Image Processed...Successfully.")

print("")

**Screenshot:**

****

**Task 2**

**Code:**

resolution=im.size

original\_image= np.array(im)

new\_image=np.array(im)

filter = np.zeros((7,7))

#Code to addes replication row and coloums to the image on the borders

original\_image= np.concatenate([[original\_image[0]],original\_image],axis=0)

original\_image= np.concatenate([[original\_image[0]],original\_image],axis=0)

original\_image= np.concatenate([[original\_image[0]],original\_image],axis=0)

original\_image=np.concatenate([original\_image,[original\_image[resolution[1]-1]]],axis=0)

original\_image=np.concatenate([original\_image,[original\_image[resolution[1]-1]]],axis=0)

original\_image=np.concatenate([original\_image,[original\_image[resolution[1]-1]]],axis=0)

original\_image=np.transpose(original\_image)

original\_image=np.concatenate([[original\_image[0]],original\_image],axis=0)

original\_image=np.concatenate([[original\_image[0]],original\_image],axis=0)

original\_image=np.concatenate([[original\_image[0]],original\_image],axis=0)

original\_image=np.concatenate([original\_image,[original\_image[resolution[0]-1]]],axis=0)

original\_image=np.concatenate([original\_image,[original\_image[resolution[0]-1]]],axis=0)

original\_image=np.concatenate([original\_image,[original\_image[resolution[0]-1]]],axis=0)

original\_image=np.transpose(original\_image)

#---------------------------------------------------------------

print("Resolution:",resolution)

sum = 0

for x in range(1,resolution[1]-1):

for y in range(1,resolution[0]-1):

for i in range(x-3,x+4): #change the values of in this case 1,2 according to the filter (3x3)

if (i==(x-3)) or (i==(x+3)):

sum=sum+1\*(int(original\_image[i][y-3]))+1\*(int(original\_image[i][y-2]))+2\*(int(original\_image[i][y-1]))+2\*(int(original\_image[i][y]))+2\*(int(original\_image[i][y+1]))+1\*(int(original\_image[i][y+2]))+1\*(int(original\_image[i][y+3])) #looping to calculating sum of filter

elif (i==(x-2)) or (i==(x+2)):

sum=sum+1\*(int(original\_image[i][y-3]))+2\*(int(original\_image[i][y-2]))+2\*(int(original\_image[i][y-1]))+4\*(int(original\_image[i][y]))+2\*(int(original\_image[i][y+1]))+2\*(int(original\_image[i][y+2]))+1\*(int(original\_image[i][y+3])) #looping to calculating sum of filter

elif (i==(x-1)) or (i==(x+1)):

sum=sum+2\*(int(original\_image[i][y-3]))+2\*(int(original\_image[i][y-2]))+4\*(int(original\_image[i][y-1]))+8\*(int(original\_image[i][y]))+4\*(int(original\_image[i][y+1]))+2\*(int(original\_image[i][y+2]))+2\*(int(original\_image[i][y+3])) #looping to calculating sum of filter

else:

sum=sum+2\*(int(original\_image[i][y-3]))+4\*(int(original\_image[i][y-2]))+8\*(int(original\_image[i][y-1]))+16\*(int(original\_image[i][y]))+8\*(int(original\_image[i][y+1]))+4\*(int(original\_image[i][y+2]))+2\*(int(original\_image[i][y+3])) #looping to calculating sum of filter

new\_image[x][y]=(sum/140) #new resultant image intensities are assigned

sum=0 #reintializing the sum variable

#Display Message Prompt

print("Image Processed...Successfully.")

print("")

**Screenshot:**

****

**Note:**

The role of sigma in the Gaussian filter is to control the variation around its mean value. So as the Sigma becomes larger the more variance allowed around mean and as the Sigma becomes smaller the less variance allowed around mean.

**Task 3**

* **Using Prewit Operator for getting Horizontal and Veritical Edges.**

**Code:**

resolution=im.size

original\_image= np.array(im)

new\_image=np.array(im)

filter = np.zeros((3,3))

point=0

print("Resolution:",resolution)

hsum = 0

vsum = 0

sum = 0

for x in range(1,resolution[1]-1):

for y in range(1,resolution[0]-1):

#Horizantal edges

for i in range(x-1,x+2): #change the values of in this case 1,2 according to the filter (3x3)

if (i==(x-1)):

hsum = hsum - (int((original\_image[i][y-1])[0])+int((original\_image[i][y])[0])+int((original\_image[i][y+1])[0]))

elif (i==(x+1)):

hsum = hsum + int((original\_image[i][y-1])[0])+int((original\_image[i][y])[0])+int((original\_image[i][y+1])[0])

#Vertical Edges

for j in range(x-1,x+2):

vsum = vsum + -1\*(int((original\_image[i][y-1])[0])) + int((original\_image[i][y+1])[0])

sum = abs(abs(vsum)+abs(hsum))

new\_image[x][y]=sum #new resultant image intensities are assigned

hsum=0 #reintializing the sum variable

vsum=0 #reintializing the sum variable

#Display Message Prompt

print("Image Processed...Successfully.")

print("")

**Screenshot:**

**Original:**



**Outputs:**

