```
from jupyterthemes import get_themes
import jupyterthemes as jt
from jupyterthemes.stylefx import set_nb_theme

set_nb_theme('chesterish')
```

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
import math
from PIL import Image
```

Problem 5.1.1

```
def gaussian_pyramid(img, num_levels):
#     here we want to build a gussian pyramid
    lower= img.copy()
    plt.figure(figsize=(30,30))
    pyramid = []
    for i in range(num_levels):
#         here we execute a gussian filter
        lower = cv2.pyrDown(lower)
        pyramid.append(np.float32(lower))
        plt.subplot( 1 ,  num_levels , i+1)
        plt.imshow(lower , cmap=plt.cm.gray)
#         we return our final pyramid
    return pyramid
```

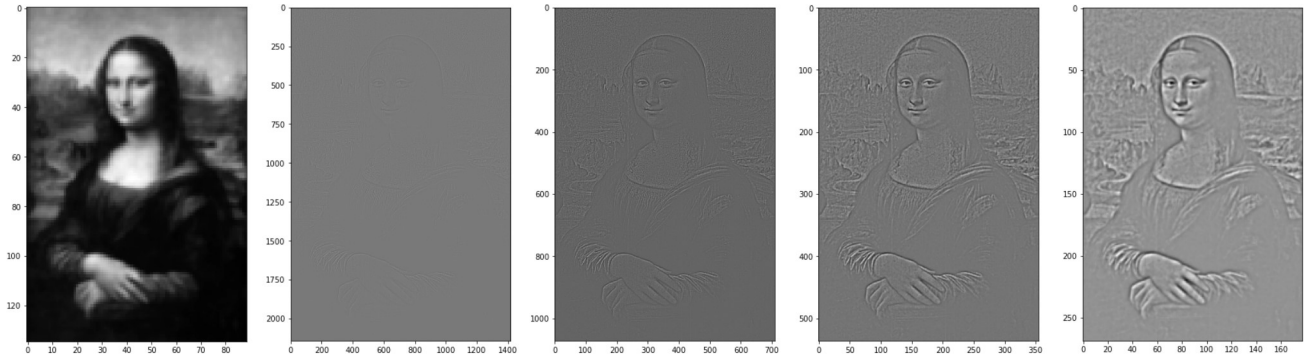
```
def Laplasina(gussian_pyramid):
    laplacian_top = gussian_pyramid[-1]
    num_levels = len(gussian_pyramid) - 1

    plt.figure(figsize=(30,30))
    plt.subplot( 1 ,  num_levels+1 , 1)
    plt.imshow(laplacian_top , cmap=plt.cm.gray )
    for i in range(num_levels,0,-1):
        size = (gussian_pyramid[i - 1].shape[1],gussian_pyramid[i - 1].shape[0])
        gaussian_expanded = cv2.pyrUp(gussian_pyramid[i], dstsize=size)
        laplacian = np.subtract(gussian_pyramid[i-1], gaussian_expanded)
        plt.subplot( 1 ,  num_levels+1 , i+1)
        plt.imshow(laplacian , cmap=plt.cm.gray )
```

Gussian Pyramid

```
monaLisa = cv2.imread("mona lisa.jpg"  , cv2.IMREAD_GRAYSCALE)
gussian_pyramid = gaussian_pyramid(monaLisa,5)
```

Laplasina Pyramid

```
Laplasina(gussian_pyramid)
```



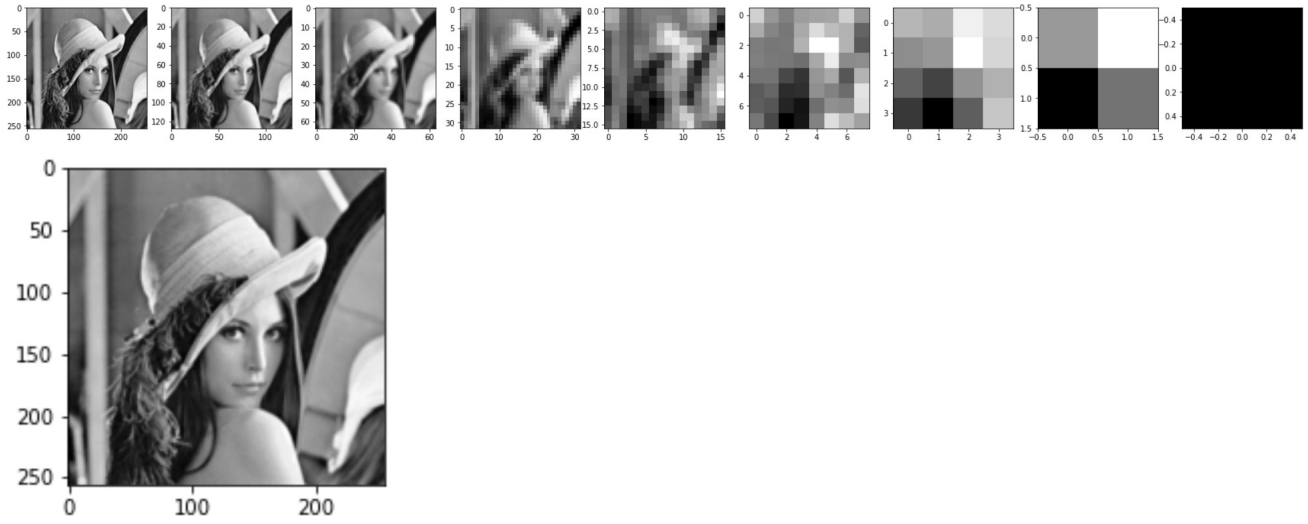
Problem 5.1.2

```
monaLisa1 = cv2.imread("Lena.bmp"  , cv2.IMREAD_GRAYSCALE)
print(monaLisa1.shape)
```

```
(512, 512)
```

Problem 5.1.3

```
def Recunstruct(gussian_pyramid):
    laplacian_top = gussian_pyramid[-1]
    num_levels = len(gussian_pyramid) - 1
    laplasina_p = [laplacian_top]
    gussian_p = gussian_pyramid[num_levels]
    plt.figure(figsize=(30,30))
    plt.subplot( 1 ,  num_levels+1 , 1)

    for i in range(num_levels,0,-1):
        size = (gussian_pyramid[i - 1].shape[1],gussian_pyramid[i - 1].shape[0])
        gaussian_expanded = cv2.pyrUp(gussian_pyramid[i], dstsize=size)
        laplacian = np.subtract(gussian_pyramid[i-1], gaussian_expanded)
        laplasina_p.append(laplacian)
    for j in range(num_levels , 0  ,-1):

         gaussian_expanded = cv2.pyrUp(gussian_p)
         gussian_p  = np.add(laplasina_p[num_levels-j+1] , gaussian_expanded)
    plt.imshow(gussian_p ,cmap=plt.cm.gray )
    return gussian_p



Lena = cv2.imread("Lena.bmp"  , cv2.IMREAD_GRAYSCALE)
pyramid = gaussian_pyramid(Lena,9)
a =Recunstruct(pyramid)
```

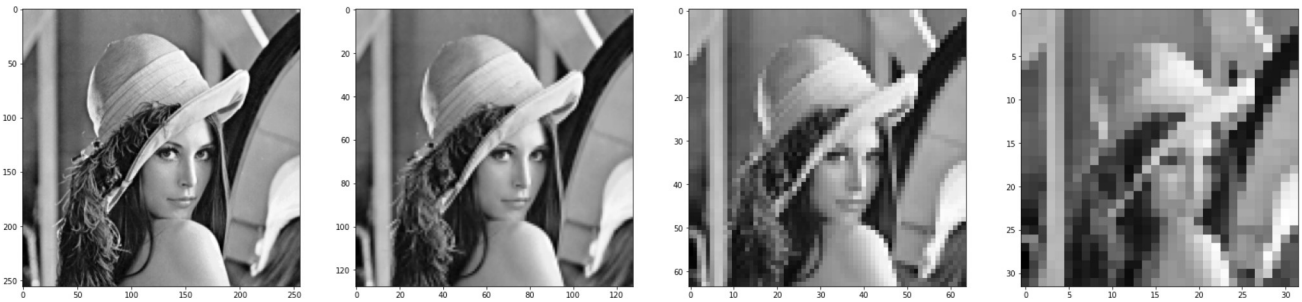


Problem 5.1.4 :

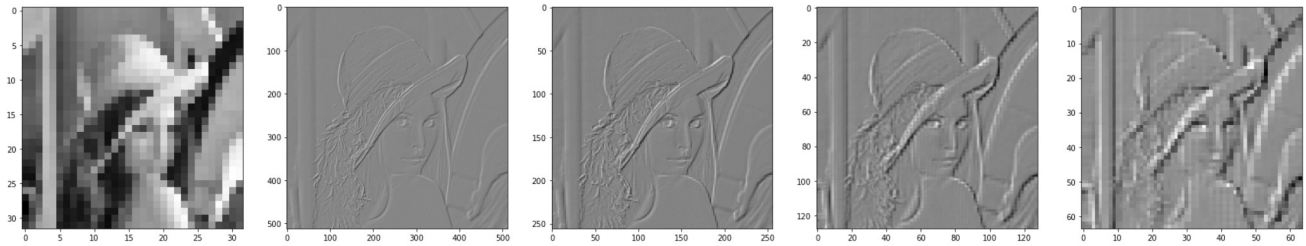
Problem 3.1.4 :

```
def box_down_sampeling(imag , level):
    picture = imag.copy()
    plt.figure(figsize=(30,30))
    plt.subplot( 1 , level+1 , 1)
    pyramid = [picture]
    kernel  = np.ones((2 ,2 ) , np.float32) / 4
    for i in range(level) :
        picture = cv2.filter2D(picture ,-1 , kernel)
        picture = picture[::2 , ::2]
        pyramid.append(np.float32(picture))
        plt.subplot(1 , level , i+1)
        plt.imshow(picture , cmap=plt.cm.gray)
    return pyramid


def Laplasina_1(gussian_pyramid):
    laplacian_top = gussian_pyramid[-1]
    num_levels = len(gussian_pyramid) - 1

    plt.figure(figsize=(30,30))
    plt.subplot( 1 ,  num_levels+1 , 1)
    plt.imshow(laplacian_top  , cmap=plt.cm.gray)
    for i in range(num_levels,0,-1):
        size = (gussian_pyramid[i - 1].shape[1],gussian_pyramid[i - 1].shape[0])
        gaussian_expanded = cv2.resize(gussian_pyramid[i], size, interpolation = cv2.INTER_
        laplacian = np.subtract(gussian_pyramid[i-1], gaussian_expanded)
        plt.subplot( 1 ,  num_levels+1 , i+1)
        plt.imshow(laplacian , cmap=plt.cm.gray )



Lena1 = cv2.imread("Lena.bmp"  , cv2.IMREAD_GRAYSCALE)
a = box_down_sampeling(Lena1,4)
print(len(a[0]), len(a[1]) , len(a[2]) , len(a[3]))
```

```
512 256 128 64
```

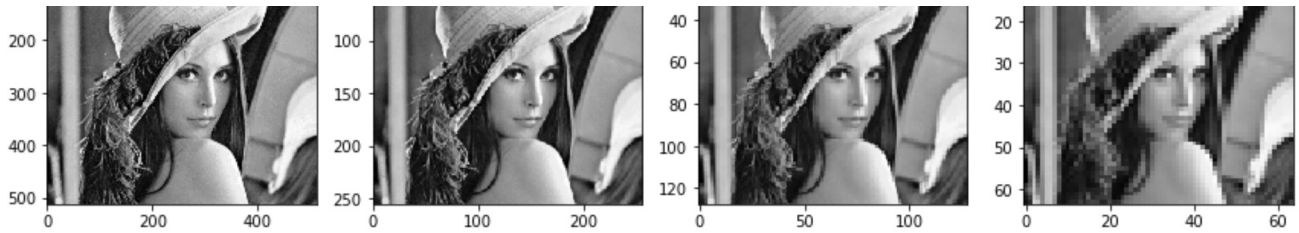


Laplasina_1(a)



Problem 5.1.5 :

```
import pywt
Lena2 = cv2.imread("Lena.bmp"  , cv2.IMREAD_GRAYSCALE)
pyramid_wavlet = [Lena2]
LH_arr = []
HL_arr = []
HH_arr = []
LL= Lena2.copy()
for kk in range(0, 3) :

    coeffs2 = pywt.dwt2(LL, 'haar' )
    LL, (LH, HL, HH) = coeffs2
    LH_arr.append(LH)
    HL_arr.append(HL)
    HH_arr.append(HH)


    pyramid_wavlet.append(np.float32(LL))


fig = plt.figure(figsize=(12, 3))


ax = fig.add_subplot(1, 4, 1)
ax.imshow(pyramid_wavlet[0], interpolation="nearest", cmap=plt.cm.gray)
ax = fig.add_subplot(1, 4, 2)
ax.imshow(pyramid_wavlet[1], interpolation="nearest", cmap=plt.cm.gray)
ax = fig.add_subplot(1, 4, 3)
ax.imshow(pyramid_wavlet[2], interpolation="nearest", cmap=plt.cm.gray)
ax = fig.add_subplot(1, 4, 4)
ax.imshow(pyramid_wavlet[3], interpolation="nearest", cmap=plt.cm.gray)
fig.tight_layout()
plt.show()
```

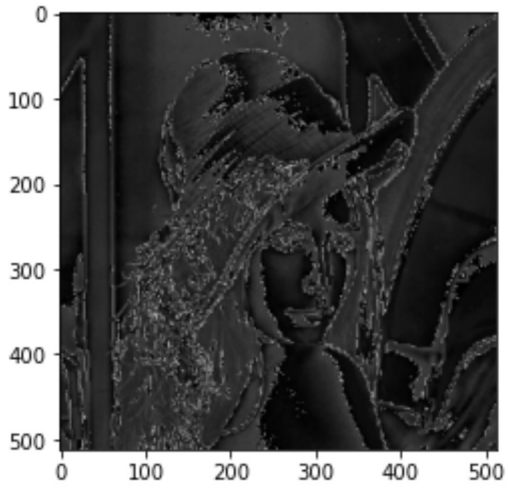


Problem 5.1.6 :

```
def coefficientQuantizer(coeff,step):
    coefficient = step * np.sign(coeff) * np.floor(np.abs(coeff)/step)
    return coefficient
```

```
def psnr(img1, img2):
    mse = np.mean((img1 - img2) ** 2)
    if mse == 0:
        return 100
    PIXEL_MAX = 255.0
    return 20 * math.log10(PIXEL_MAX / math.sqrt(mse))


cA  = coefficientQuantizer(pyramid_wavlet[3]  , 2)
for so in range(2 , -1 , -1):


    # (cH, cV, cD)

    cH = coefficientQuantizer(LH_arr[so],2)
    cV = coefficientQuantizer(HL_arr[so] , 2 )
    cD = coefficientQuantizer(HH_arr[so] , 2 )

    # inverse wavelet transform
    coefficients = [cA, (cH, cV, cD)]
    print(cA.shape)
    print(cH.shape)
    cA= pywt.idwt2(coefficients, 'haar')
    cA= cA.astype('uint8')

plt.imshow(cA , cmap='gray')

Lena3 = cv2.imread("Lena.bmp"  , cv2.IMREAD_GRAYSCALE)
print(f'PSNR: {psnr(Lena3,cA)}')
```

```
(64, 64)
(64, 64)
(128, 128)
(128, 128)
(256, 256)
(256, 256)
PSNR: 33.407473746088314
```




```
$ jupyter nbconvert --to pdf Untitled.pdf
```

```
  Input In [48]
    $ jupyter nbconvert --to pdf Untitled.pdf
    ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

Colab paid products - Cancel contracts here