Wavelet

زهرا نیازی

اطلاعات گزارش

تاریخ: 18/11/1401

واژگان کلیدی:

چکیده

موجک و تبدیل آن برای رفع ایرادات تبدیل فوریه برای تحلیل سیگنال ها و به خصوص فشرده سازی و حذف نویز معرفی شده است.

1-مقدمه

در این تمرین ما ابتدا با مفهوم هرم ها آشنا شده و استفاده از آنها را تست میکنیم. بهبود اعمال فیلتر گوسی با استفاده از هرم ها را بررسی میکنیم و ابعاد هرم ها را می سنجیم. همچنین ساخت هرم با توابع فیلتر و درونیابی جدید را امتحان کرده و در نهایت تبدیل موجک و ضرایب آن را بررسی خواهیم کرد.

2-شرح تکنیکال

1-2

در این بخش از ما خواسته شده یک هرم گوسین با 5 سطح برای تصویر mona lisa تشکیل دهیم و در ادامه هرم لاپلاسین آن (که ناشی از اختلاف گوسین هاست) را ساخته و نشان دهیم. یک ساختار قدرتمند، اما از لحاظ مفهومی ساده برای نمایش تصاویر در بیش از یک resolution تصویر، هرم است. هرم تصویر که در اصل برای کاربردهای بینایی ماشین و فشرده سازی تصویر ابداع شده است، مجموعه ای از تصاویر با وضوح کاهشی است که به شکل یک هرم مرتب شده اند. همانطور که در شکل زیر مشاهده می شود، پایه هرم حاوی نمایشی با resolution بالا از تصویر در حال پردازش است. راس هرم شامل یک تقریب از تصویر با رزولوشن کم است. با بالا رفتن از هرم، اندازه و وضوح کاهش می یابد.

2-2

در این تمرین از ما خواسته شده بررسی کنیم که چگونه seperability و cascading می تواند به سرعت بخشیدن به هموارسازی گاوسی کمک کند.

در مرحله بعد شبه کد یک الگوریتم سریع برای محاسبه یک هرم گوسی 3 مرحله ای (فیلتر شده با σ، √2σ، σ2) از یک تصویر دو بعدی را ارائه کنیم

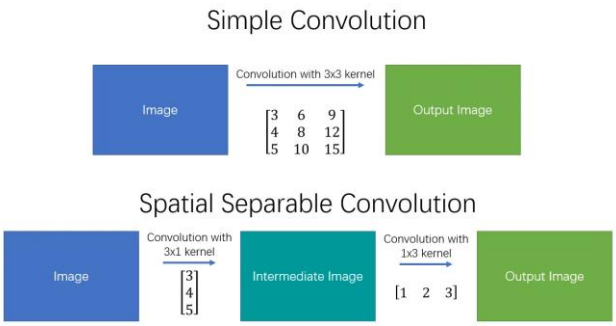
فیلتر گوسی فیلتر بسیار جالبی است زیرا وزن‌ها همگن نیستند و به پیکسل‌های نزدیک‌تر بیشتر از پیکسل‌های دیگر تأثیر می‌گذارد.

ما فرمول کلی را برای ایجاد یک فیلتر Gaussian پیاده سازی را بررسی میکنیم. همچنین می‌توانیم به لطف کانولوشن، کرنل های دوبعدی ایجاد کنیم، زیرا هسته گوسی قابل جداسازی است.

$$G_\sigma * f = \left(g_{\sigma\rightarrow} * g_{\sigma\uparrow}\right) * f = g_{\sigma\rightarrow} * \left(g_{\sigma\uparrow} * f\right)$$

وقتی یک کرنل جدایی پذیر است یعنی میتوان کانولوشن دوبعدی را از روی دو کانولوشن یک بعدی cascaded به دست آورد.

* zahra.niazi2001@gmail.com

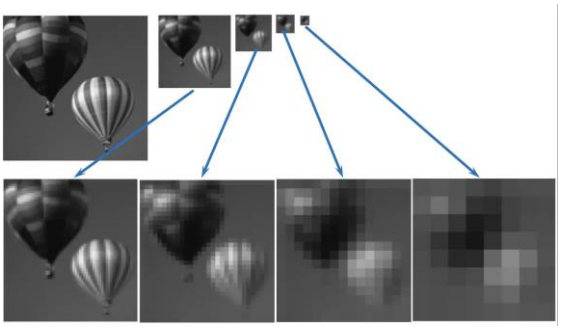


در این بخش، اندازه هسته براساس انتخاب انحراف معیار فیلتر گاوسی محدود می شود. در واقع، ما انتخاب کردیم که از استاندارد $3\sigma \pm$ به عنوان مرجع استفاده کنیم، بنابراین اگر σ برابر یک باشد، فیلتر دارای عرض $2*3\sigma + 1$ خواهد بود و در مرکز $\lceil 3\sigma+1 \rceil$ قرار می گیرد. که اینجا $\lceil \rceil$ به معنی تابع ceiling استفاده می شود. در واقع، برای داشتن یک هسته 5×5، باید $\sigma = ½$ داشته باشید که قرار است بر روی 3 متمرکز شود. برای پیاده سازی الگوریتم ساخت هرم گوسی نکات زیر را در نظر میگیریم:

- هرم گاوسی روشی معرف برای ساختن نمایش چند مقیاسی از یک تصویر است. این شامل دو مرحله است:
  1. هموارسازی برای حذف اجزای با فرکانس بالا که می‌تواند منجر به aliasing شود.
  2. down-sampling: اندازه تصویر را به نصف کاهش میدهیم.

- صاف کردن، که به آن blur کردن نیز می گویند، با کانولوشن تصویر با فیلتر گوسی همانطور که قبلا توضیح داده شد، انجام می شود.
- مرحله دوم، downsampling، به این معنی است که به سادگی هر دومین پیکسل را از تصویر در آخرین مرحله برمیداریم و نسخه کوچک شده تصویر را بازسازی میکنیم.

در هر سطح از هرم، ما اطلاعات ساختار تصویر مربوط به اجزای فرکانس بالا را از دست می دهیم و کیفیت تصویر را همانطور که در نتایج نشان داده شده کاهش می دهد. با این حال، با انجام این کار، می‌توانیم مقیاس تصویر را در هر سطح بدون معرفی مصنوعات کاهش دهیم.



پس در پیاده سازی این بخش طبق مراحل زیر پیش میرویم:

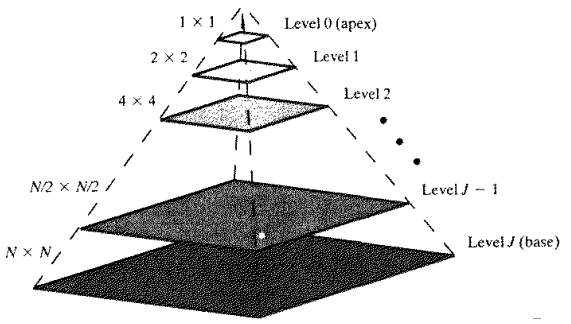
- به جای استفاده از فیلتر دوبعدی، فیلتر یک بعدی با سیگمای مربوطه را می یابیم
- این فیلتر یک بعدی را با transposeش کانولو کرده تا به کرنل مطلوب برسیم
- تصویر را با کرنل جدید کانولو کرده و توسط آن آن smoothing را انجام دهیم.

3-2

در این تمرین از ما خواسته شده با توجه به اندازه تصویر $N \times N$، که در آن $N=2^J$، حداکثر تعداد سطوحی که می توانیم در یک نمایش هرم تقریبی داشته باشیم را محاسبه کنیم. سپس تعداد کل پیکسل های هرم (یعنی شامل پیکسل ها در تمام سطوح هرم) را بیابیم. در مرحله آخر خواسته شده تا از آنجایی که این عدد بزرگتر از عدد پیکسل اصلی است، برخی از مزایای استفاده از هرم تقریبی را بیان کنیم.

از آنجایی که سطح J اندازه $2^J x 2^J$ یا N X N است، که در آن J = log2 N می باشد، سطح متوسط j اندازه $2^j x 2^j$ خواهد داشت.

هرمی که تا اخرین سطح ممکن پر شده باشد دارای J + 1 سطح رزولوشن می باشد.



تعداد تمامی المان های موجود در یک هرمی که کامل ساخته شده باشد مطابق فرمول زیر به دست می آید:

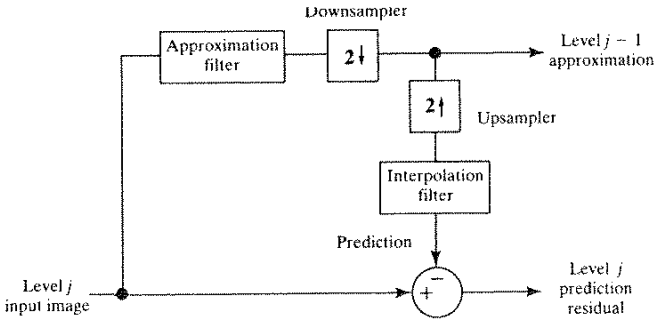
$$N^2\left(1+\frac{1}{(4)^1}+\frac{1}{(4)^2}+\cdots+\frac{1}{(4)^P}\right)\leq\frac{4}{3}N^2.$$

همانطور که مشاهده میشود تنها 1/3N2 تعداد پیکسلهای اضافه ایست که باید نگه داریم. این مقدار با توجه به مزیتهایی که هرم ها برای ما به وجود می آوردند مقدار ناچیزی است.

از جمله این مزیت ها میتوان به موارد زیر اشاره کرد:

- Alternative representation: که میتوان از هریک از تصاویر در سطح های مختلف هرم به عنوان scaleهای مختلف آن تصویر استفاده کنیم.
- Detail manipulation: سطوح هرم لاپلاس (که از هرم گوسی به دست می آید) را می توان به تصویر اصلی اضافه کرد یا از آن حذف کرد تا جزئیات را در مقیاس های مختلف تقویت یا کاهش داد. با این حال، دستکاری جزئیات این فرم در بسیاری از موارد برای تولید halo artifacts شناخته شده است که منجر به توسعه جایگزین هایی مانند bilateral filter می شود.
- Image Compression: برخی از فرمت‌های فشرده‌سازی فایل تصویر از الگوریتم هایی استفاده میکنند که اینها را می توان نوعی هرم تصویری دید. از آنجایی که این فرمت‌های فایل ابتدا ویژگی‌های «مقیاس بزرگ» را ذخیره می‌کنند، و جزئیات ریز را بعداً در فایل ذخیره می‌کنند، یک بیننده خاص که یک «thumbnail» کوچک یا روی یک صفحه کوچک نمایش می‌دهد می‌تواند به سرعت قدر کافی از تصویر را دانلود کند تا در پیکسل‌های موجود آن را نمایش دهد. بنابراین یک فایل می‌تواند رزولوشن‌های بیننده زیادی را پشتیبانی کند، نه اینکه برای هر رزولوشن فایلی متفاوت ذخیره یا تولید کند.

شکل زیر یک سیستم ساده برای ساخت هرم های تصویری را نشان می دهد. خروجی approximation سطح j - 1 برای ایجاد هرم های approximation استفاده می شود که حاوی یک یا چند approximation از تصویر اصلی است.



هم تصویر اصلی که در قاعده هرم قرار دارد و هم approximation رزولوشن آن مستقیماً قابل دسترسی می باشند و میتوانند دستکاری شوند. خروجی باقیمانده approximation سطح j شکل زیر برای ساخت هرم‌های prediction residual استفاده می‌شود. این هرم‌ها حاوی تقریبی با رزولوشن پایین از تصویر اصلی در سطح J – P می باشند و اطلاعاتی برای ساختن P سطح تقریب دیگر با رزولوشن های بالاتر را دارند را دارند.

اطلاعات در سطح j تفاوت بین تقریب سطح j از هرم تقریبی مربوطه و برآورد آن تقریب بر اساس prediction residual سطح j - 1 است. این تفاوت را می توان کدگذاری کرد و بنابراین ذخیره و منتقل کردن آن کارآمدتر از تقریب می باشد.

4-2

در این تمرین از ما خواسته شده برای تصویر لنا یک هرم approximation 3سطحی و هرم prediction residual مربوطه را به صورت دستی محاسبه کنیم. همچنین قید شده که برای approximation یا down-sampling از فیلتر میانگین گیر 2×2 و برای prediction یا upsample کردن از pixel replication برای فیلترهای interpolation استفاده کنیم.

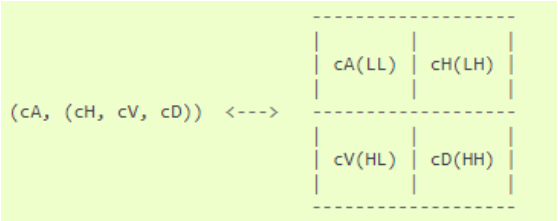
برای ایجاد این هرم ها، ابتدا توابع مربوطه برای downsample و upsample کردن تصاویر را ایجاد میکنیم. این توابع مشابه توابعی که در تکلیف اول درس بینایی پیاده سازی شد، استفاده میشوند.

کوچک کردن تصویر به فاکتور 2 به این معناست که طول و عرض تصویر هر دو تقسیم بر 2 میشوند. Averaging به این معناست که برای هر پیکسلی که در تصویر کوچک شده قرار میدهیم، میانگین پیکسل های اطراف آن در تصویر اصلی را محاسبه کرده و قرار میدهیم. روش pixel replication بیان میکند که برای هر 4 پیکسل جدیدی که باید در عکس بزرگ نمایی شده مقداردهی شوند، مقدار یکی از پیکسل ها نکرار شود

5-2

در این تمرین از ما خواسته شده برای تصویر لنا در مقیاس خاکستری، تبدیل موجک (با 3 سطح) را با استفاده از فیلترهای تجزیه و تحلیل Haar محاسبه کنیم. همچنین باید تفاوت بین هرم های ایجاد شده در این بخش و بخش 2-4 را هم بررسی کنیم.

برای محاسبه تبدیل موجک از تابع pywt.dwt2() استفاده میکنیم. پارامترهای این تابع نیز باید تصویر ورودی و 'haar' باشد. خروجی این تابع به شکل زیر است:



Approximation, horizontal detail, vertical detail and diagonal detail coefficients

این مقادیر 4 تصویر با اندازه نصف تصویر ورودی میباشد. که این 4 تصویر عبارتند از:

- cA/LL: تصویر اصلی با نصف اندازه تصویر قبلی
- cH/LH: لبه های افقی
- cV/HL: لبه های عمودی
- cD/HH: لبه های اریب

در مرحله بعد برای ساختن سطح بعدی هرم تصویر cA/LL را به عنوان ورودی تابع **pywt.dwt2()** می دهیم و همین کار را یک بار دیگر تکرار میکنیم.

درباره مقایسه دو روش ذکر شده، به نظر می رسد که استفاده از فیلتر میانگین گیر نتواند بزرگترین فرکانس تصویر را کاهش دهد و به همین دلیل ممکن است با پدیده aliasing در این حالت مواجه شویم.

6-2

در این تمرین از ما خواسته شده بر روی تمام ضرایب موجک (کل زیر باند) ایجاد شده در تمرین قبلی را کوانتیزه کنیم. همچنین این کار را طبق فرمول داده شده و با اندازه گام γ=2 انجام دهیم.

$$c'(u,v) = \gamma \times sgn[c(u,\ v)] \times floor\left[\frac{|c(u,\ v)|}{\gamma}\right]$$

که اینجا c ضرایب موجک می باشند.

سپس تصویر را از ضرایب موجک کوانتیزه شده با استفاده از فیلتر سنتز هار بازسازی کنیم.

در نهایت باید مقادیر PSNR را گزارش کنیم.

برای انجام این کار در مرحله اول با استفاده از تابع pywt.wavdec2() با پارامترهای تصویر، 'haar' و تعداد سطوح مطلوب (اینجا 3) ضرایب موجک را استخراج کرده و تصویر را decompose میکنیم. سپس ضرایب را به تابع pywt.coeffs_to_array() می دهیم تا بتوانیم هرم موجک را نمایش دهیم.

در مرحله بعد به ترتیب ضرایب مربوط به هر سطح را استخراج کرده و تابع ذکرشده را بر آنها اعمال می کنیم.

حال که ضرایب جدید کوانتیزه شده را به دست آوردیم از تابع pywt.coeffs_to_array() استفاده کرده و هرم جدید را نمایش میدهیم.

در مرحله آخر هم با استفاده از تابع pywt.wavrec2() با پارامترهای ضرایب و 'haar'، تصویر reconstructed را به دست می آوریم.

نسبت سیگنال به نویز پیک (*PSNR*) نسبت بین حداکثر توان ممکن یک تصویر و توان نویز مخرب است که بر کیفیت نمایش آن تأثیر می گذارد. برای تخمین *PSNR* یک تصویر، باید آن تصویر را با یک تصویر تمیز ایده آل با حداکثر توان ممکن مقایسه کرد.

$$PSNR = 10log_{10}(\frac{(L-1)^2}{MSE}) = 20log_{10}(\frac{L-1}{RMSE})$$

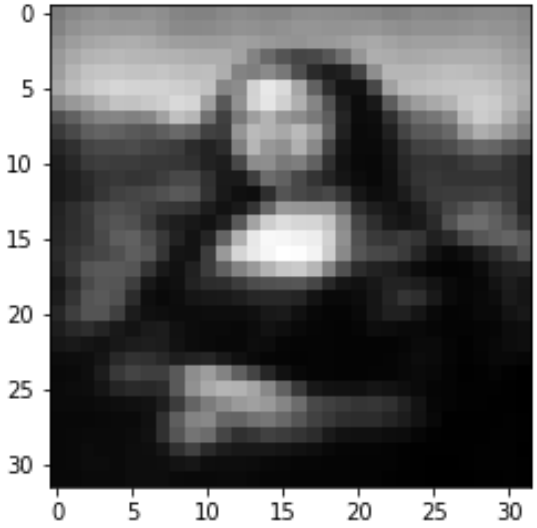
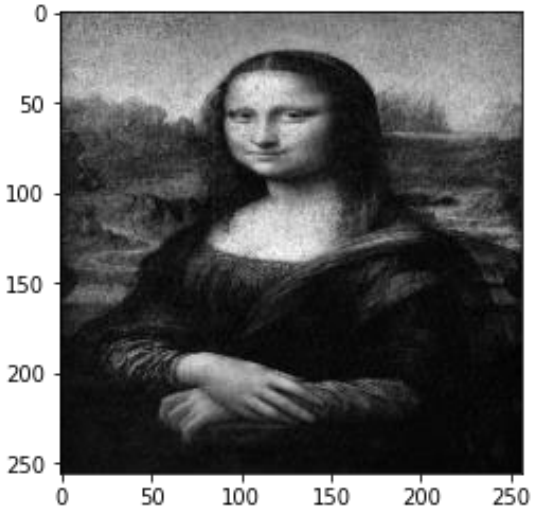
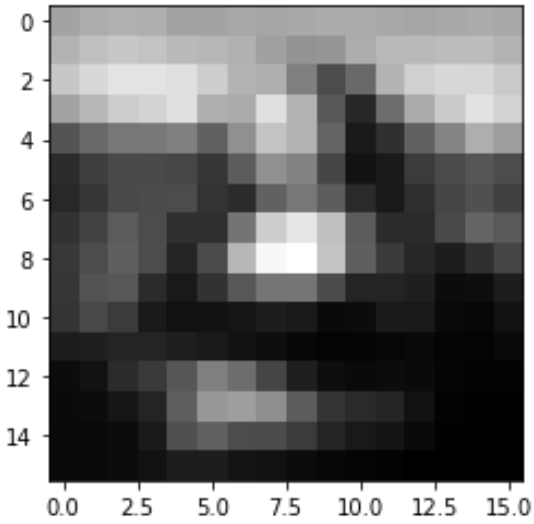
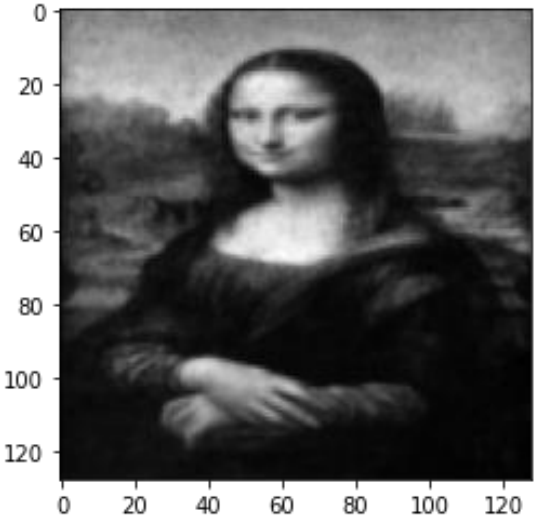
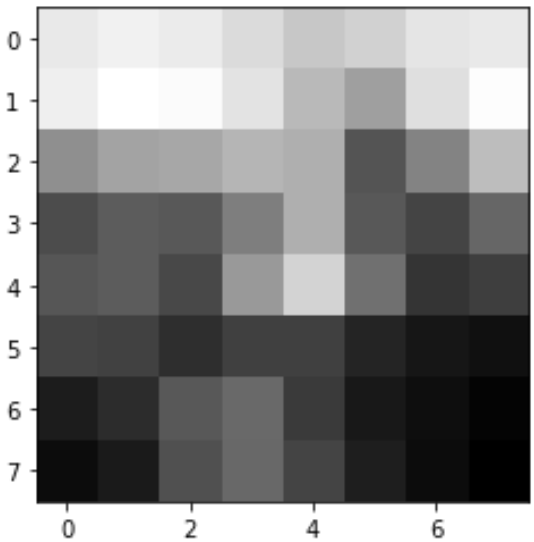
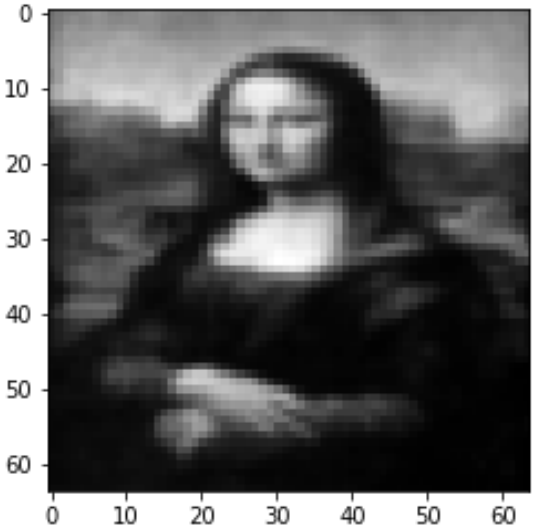
در اینجا، L تعداد حداکثر سطوح شدت ممکن (حداقل سطح شدت فرض کنید 0 باشد) در یک تصویر است و *MSE* میانگین مربعات خطا است.

PSNR بیشتر برای تخمین راندمان کمپرسورها، فیلترها و... استفاده می شود.

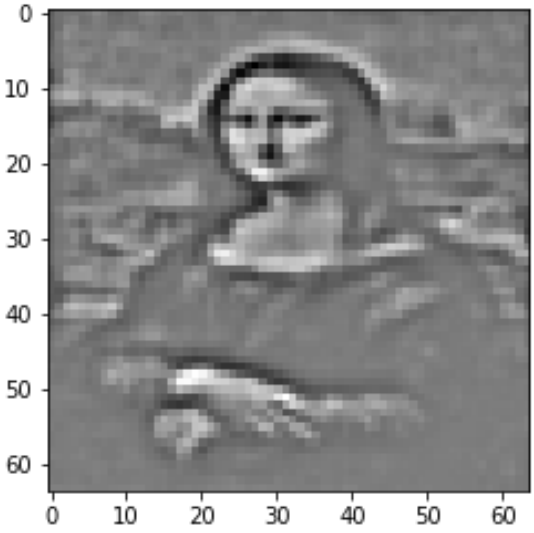
3-نتایج

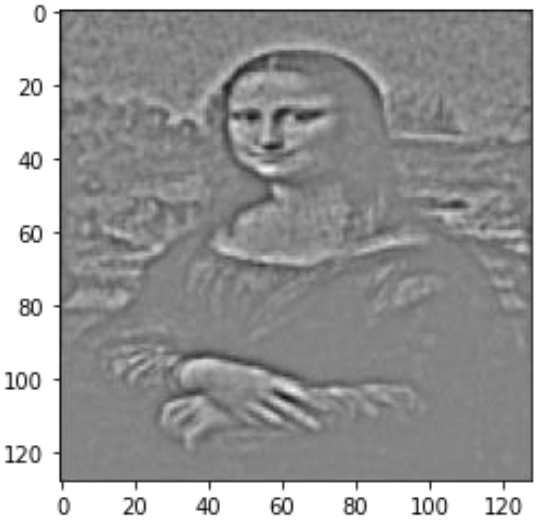
1-3

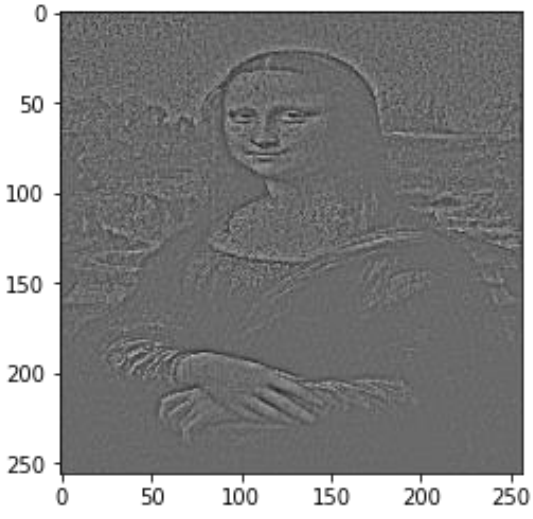
در این تمرین از ما خواسته شده یک هرم گوسین با 5 سطح برای تصویر mona lisa تشکیل دهیم. 5 سطح هرم در تصاویر زیر نمایش داده شده اند:

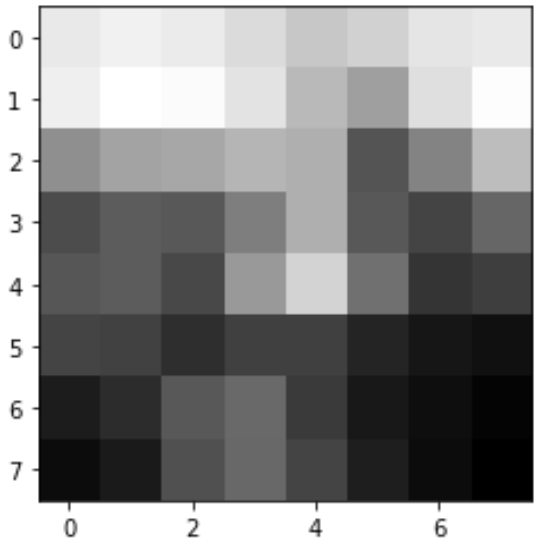







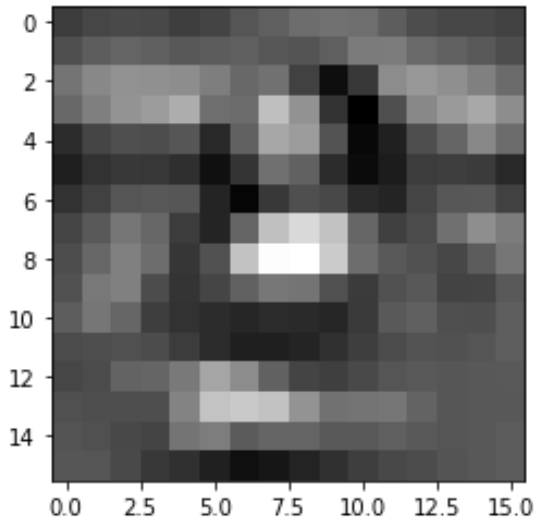
در ادامه هرم لاپلاسین آن (که ناشی از اختلاف گوسین هاست) را ساخته و در تصاویر زیر نمایش میدهیم:

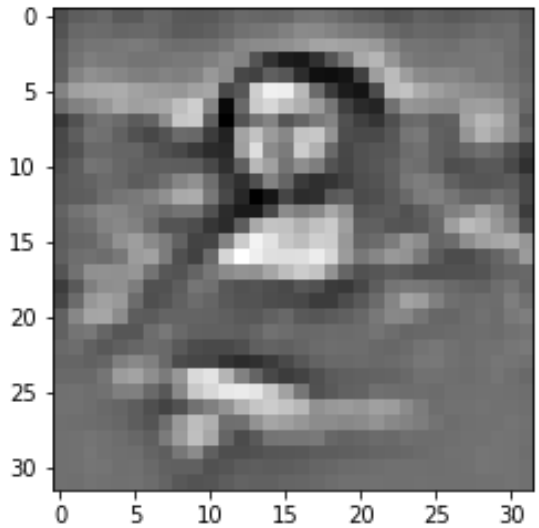






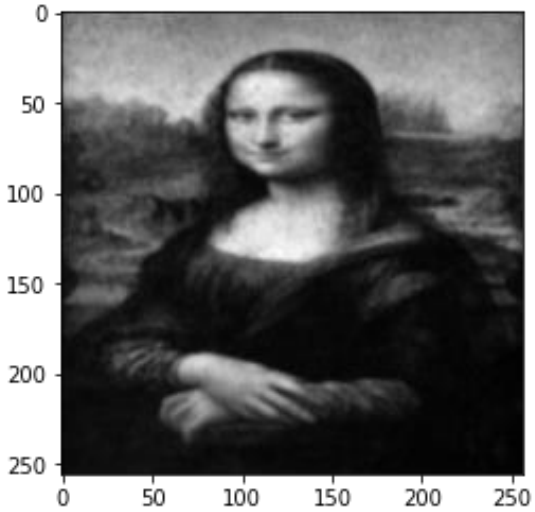
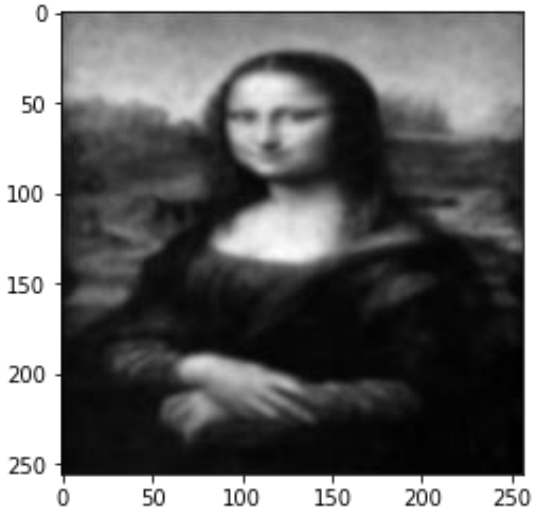
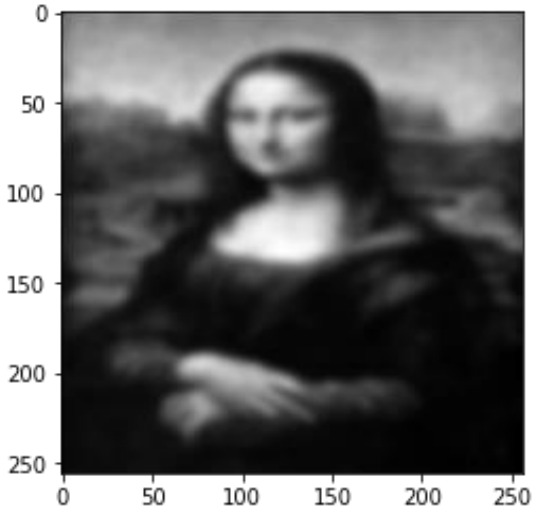






2-3

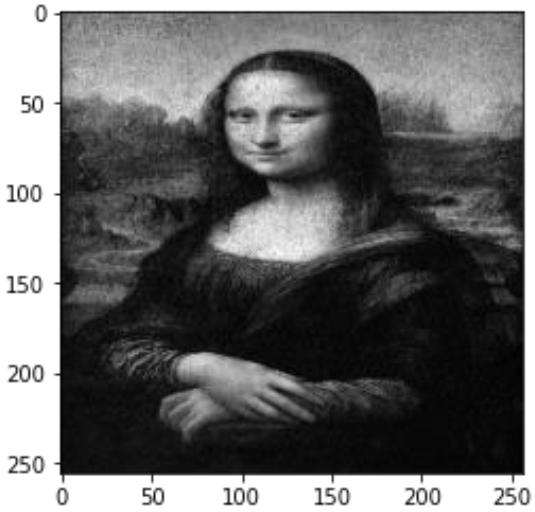
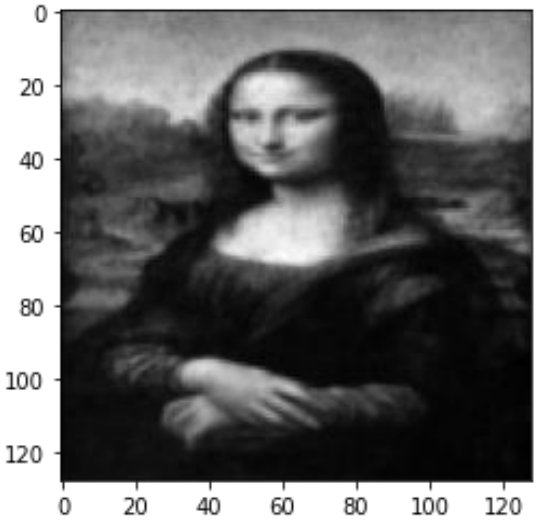
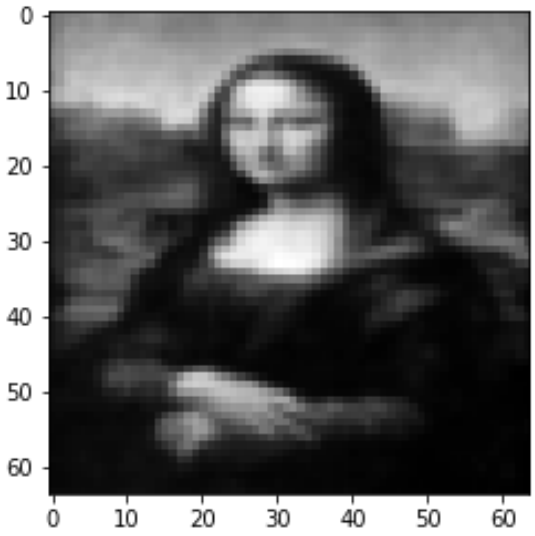
طبق الگوریتم شرح داده شده در بالا، تصاویر فیلتر شده با σ، √2σ، 2σ به ترتیب نمایش داده میشوند:

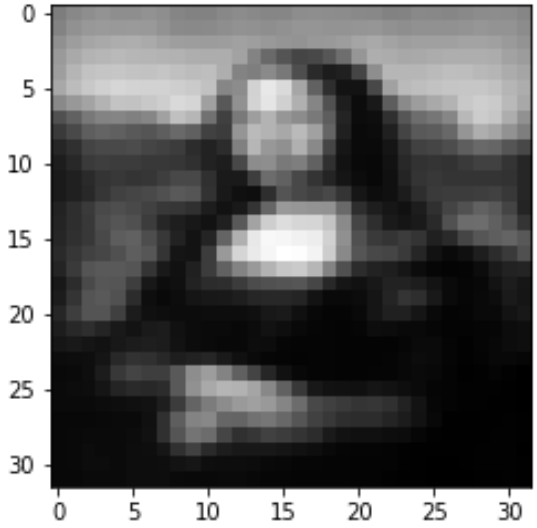




3-3

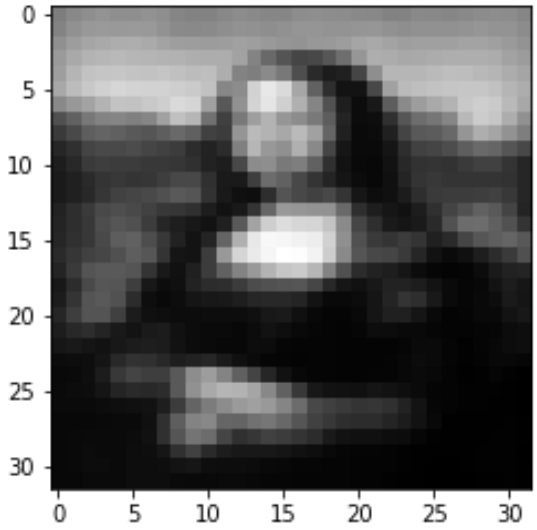
تصاویر زیر هر یک حاصل از هرم گوسی (با سیگما های مختلف) در سطح سوم هستند. همانطور که مشاهده می شود مقدار سیگما انحراف معیار توزیع گوسی را تعیین می کند و هر چه بیشتر باشد، مقدار smoothing بیشتر است.

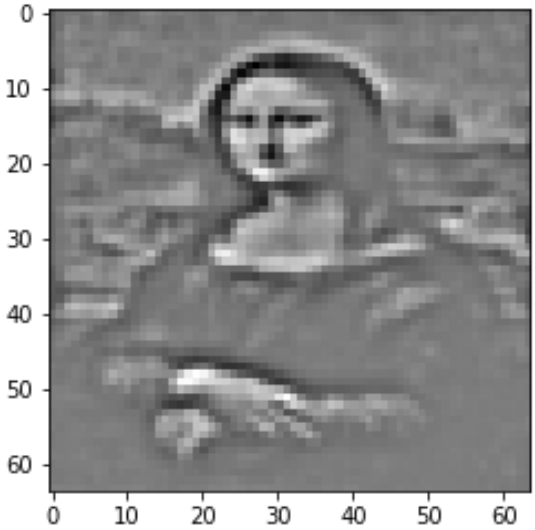
تصاویر زیر هر یک حاصل از هرم گوسی سه سطحی می باشند:

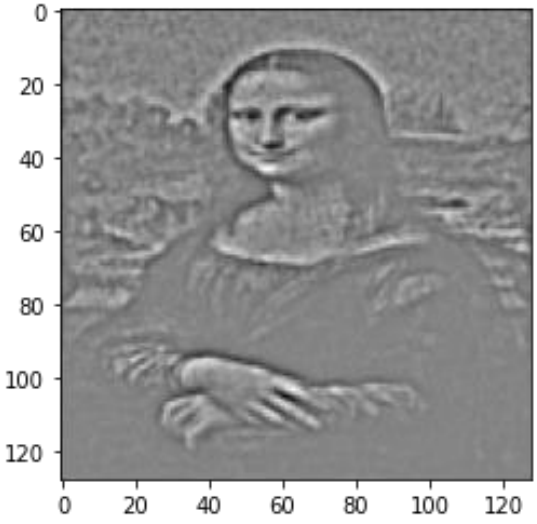




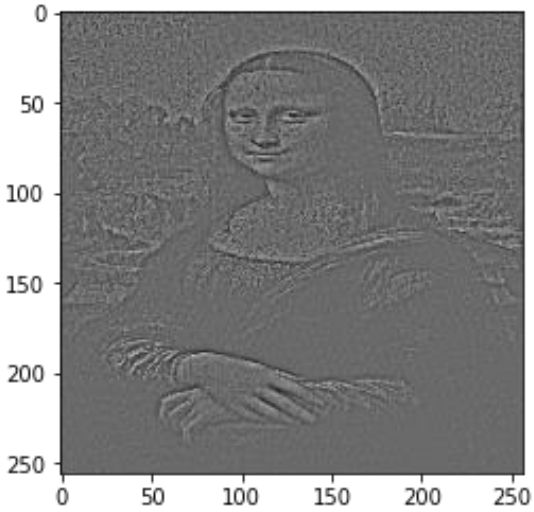


تصاویر زیر هر یک حاصل از هرم لاپلاسی سه سطحی می باشند:









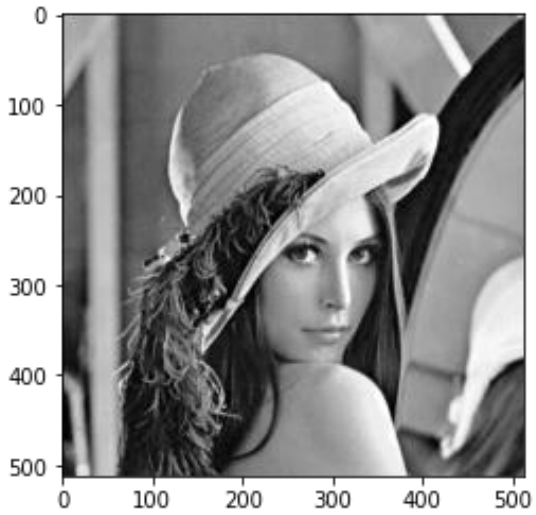
اگر دقت کنیم لبه های تصویر در این تصاویر وجود دارند که میتوان از آن برای بازسازی تصویر استفاده کرد.

4-3

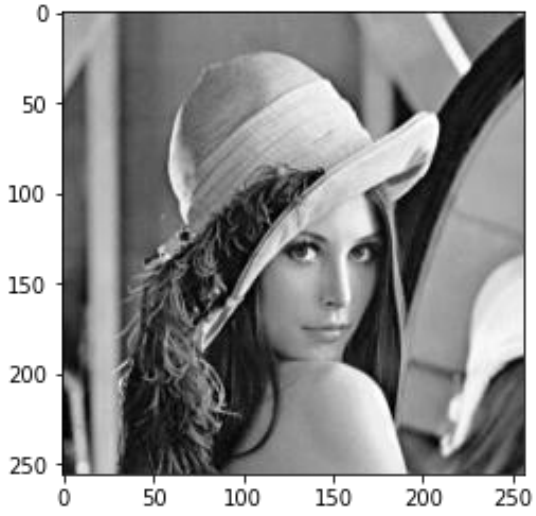
با در این تمرین از ما خواسته شده برای تصویر لنا یک هرم approximation3سطحی و هرم prediction residual مربوطه را به صورت دستی محاسبه کنیم. همچنین قید شده که برای approximation یا down-sampling از فیلتر میانگین گیر 2×2 و برای prediction یا upsample کردن از pixel replication برای فیلترهای interpolation استفاده کنیم.

ابتدا Approximation Pyramid را بررسی میکنیم؛

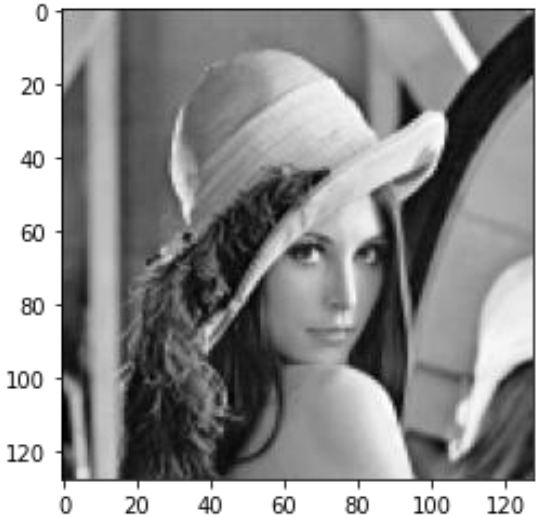
سطح صفر :



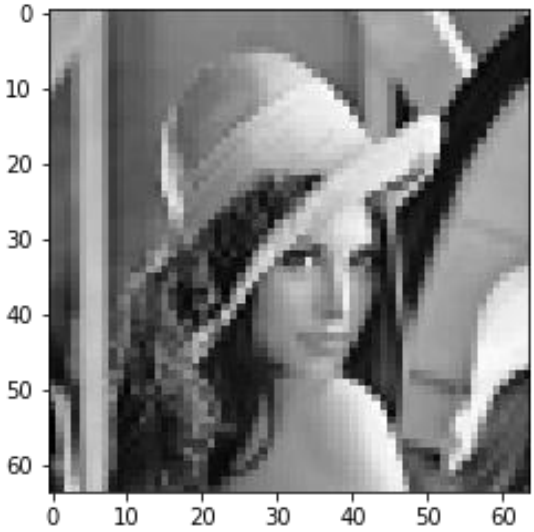
سطح 1:



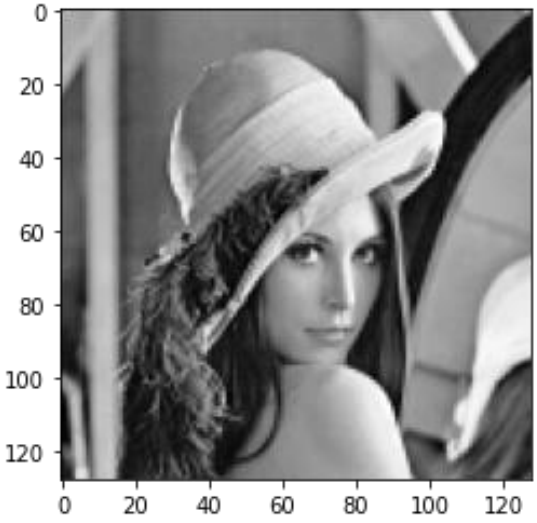
سطح2:

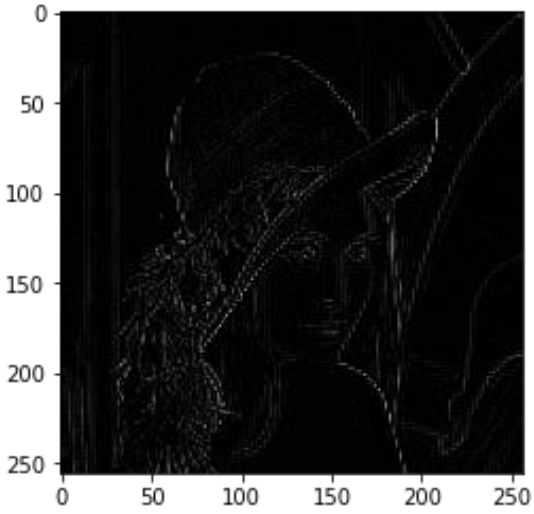


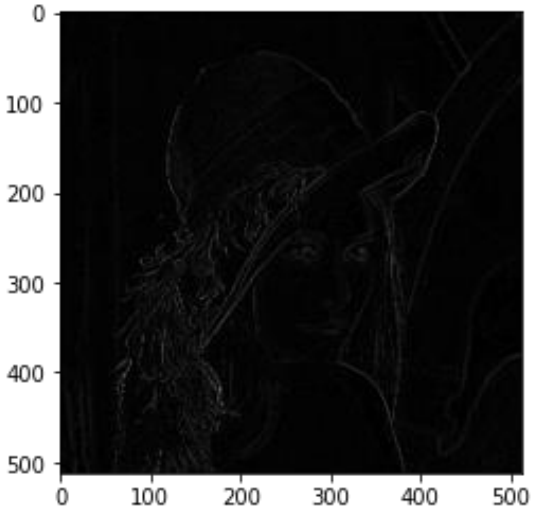
سطح 3:



سپس با داشتن تصویر بالا که سطح 3 هرم approximation است هرم prediction residual را میسازیم:







5-3

در این تمرین از ما خواسته شده برای تصویر لنا در مقیاس خاکستری، تبدیل موجک (با 3 سطح) را با استفاده از فیلترهای تجزیه و تحلیل Haar محاسبه کنیم.

تصاویر زیر به ترتیب cA, cH, cV, cD سطح اول هرم موجک می باشد:

cA/LL

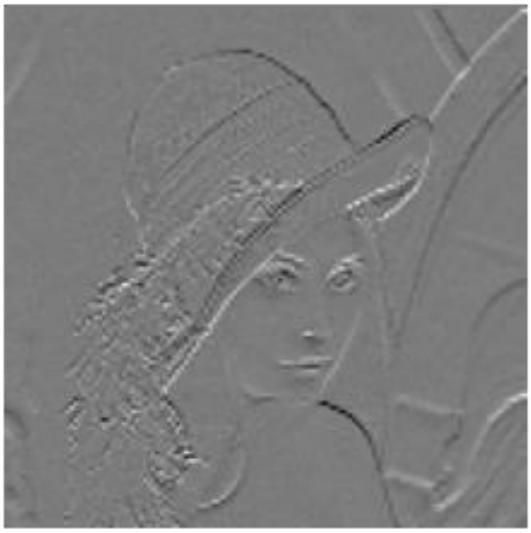
cH/LH

cV/HL

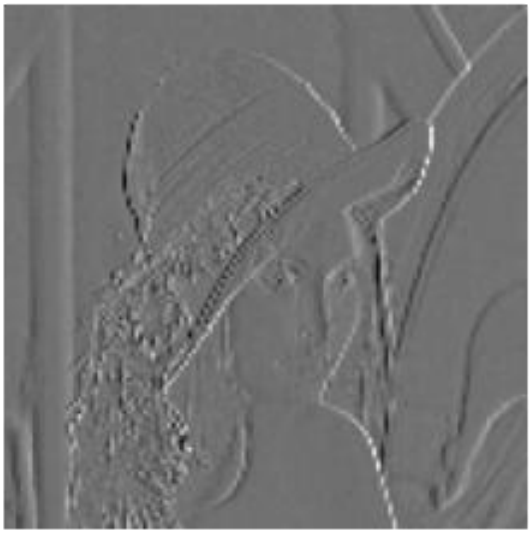
cD/HH

تصاویر زیر به ترتیب cA, cH, cV, cD سطح دوم هرم موجک می باشد:

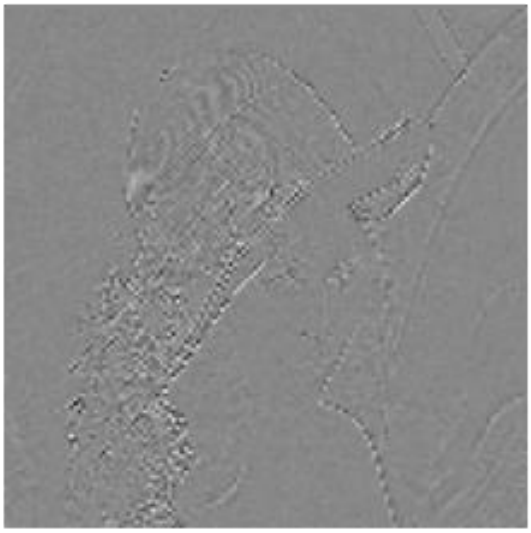
cA/LL



cH/LH



cV/HL



cD/HH

تصاویر زیر به ترتیب cA, cH, cV, cD سطح سوم هرم موجک می باشد:

cA/LL

cH/

cV/HL

cD/HH

6-3

در این تمرین از ما خواسته شده بر روی تمام ضرایب موجک (کل زیر باند) ایجاد شده در تمرین قبلی را کوانتیزه کنیم. همچنین این کار را طبق فرمول داده شده و با اندازه گام $\gamma=2$ انجام دهیم.

ابتدا هرم موجک تصویر اصلی را مشاهده میکنیم:

این هرم موجک تصویر بازسازی شده می باشد:

در مرحله آخر نیز باید pnsr را برای تصویر بازسازی شده نسبت به تصویر اولیه را محاسبه کرده و گزارش کنیم.

مقداری که PNSR به ما میدهد برابر است با:

42.74083149321203

4- Code

https://colab.research.google.com/drive/16HxHsyXUSUhH97oPdzvWClm8FpGvmZG8?usp=sharing

5.1.1

```
def gaussian_pyramid(img, num_levels):
  lower = img.copy()
  gaussian_pyr = [lower]
  for i in range(num_levels):

    lower = cv2.pyrDown(lower)
    gaussian_pyr.append(np.float32(lower))
  return gaussian_pyr


def laplacian_pyramid(gaussian_pyr):
    laplacian_top = gaussian_pyr[-1]
    num_levels = len(gaussian_pyr) - 1

    laplacian_pyr = [laplacian_top]
    for i in range(num_levels,0,-1):
        size = (gaussian_pyr[i - 1].shape[1], gaussian_pyr[i - 1].shape[
0])
        gaussian_expanded = cv2.pyrUp(gaussian_pyr[i], dstsize=size)
        laplacian = np.subtract(gaussian_pyr[i-1], gaussian_expanded)
        laplacian_pyr.append(laplacian)
    return laplacian_pyr
```

5.1.2

```
def get_kernel(sigma):
    size = int(4 * sigma + 1)
    kernel = cv2.getGaussianKernel(size, sigma)
    kernel = np.dot(kernel, kernel.T)
    return kernel
    return
```

```
k1 = get_kernel(1)
r1 = cv2.filter2D(monalisa, -1, k1)
plt.imshow(r1,'gray')
k2 = get_kernel(np.sqrt(2))
r2 = cv2.filter2D(r1, -1, k2)
plt.imshow(r2,'gray')
k3 = get_kernel(2)
r3 = cv2.filter2D(r2, -1, k3)
plt.imshow(r3,'gray')
```

5.1.3

```
#maximum number of levels
J = int(round(math.log(N, 2)))

#no. total pixels in pyramid
(N*N)*(4/3)
```

5.1.4

```
def down_sample_avg(im):
    original_width = im.shape[1]
    original_height = im.shape[0]
    width = original_width // 2
    height = original_height // 2
    resized_image = np.zeros(shape=(height, width), dtype=np.uint8)
    scale = 2
    for i in range(height):
        for j in range(width):
            temp = 0
            for x in range(scale):
                for y in range(scale):
                    temp += im[i*scale + x, j*scale + y]
            resized_image[i, j] = temp/(scale*scale)

    return resized_image
```

```
def nearest(input):
  sx,sy = input.shape
  output = np.zeros((sx*2, sy*2), input.dtype)
  for y in range(len(output)):
    for x in range(len(output[y])):
      proj_x = x // 2
      proj_y = y // 2
      output[y][x] = input[proj_y][proj_x]

  return output
```

```
# approximation pyramid 2*2 averaging
G = lena.copy()
gpA = [G]
for i in range(3):
    G = down_sample_avg(G)
    gpA.append(G)
```

```
# prediction residual pyramid pixel replication for the interpolation fi
lter
lpA = [gpA[2]]
for i in range(2,0,-1):
    GE = nearest(gpA[i])
    L = cv2.subtract(gpA[i-1],GE)
    lpA.append(L)
```

5.1.5

```
cA, (cH, cV, cD) = pywt.dwt2(lena, 'haar', mode='periodization')

plt.imshow(cA, cmap='gray')
plt.axis('off')
plt.show()
plt.imshow(cH, cmap='gray')
plt.axis('off')
plt.show()
plt.imshow(cV, cmap='gray')
plt.axis('off')
plt.show()
plt.imshow(cD, cmap='gray')
plt.axis('off')
plt.show()
```

```
cA2, (cH2, cV2, cD2) = pywt.dwt2(cA, 'haar', mode='periodization')

plt.imshow(cA2, cmap='gray')
plt.axis('off')
plt.show()
plt.imshow(cH2, cmap='gray')
plt.axis('off')
plt.show()
plt.imshow(cV2, cmap='gray')
plt.axis('off')
plt.show()
plt.imshow(cD2, cmap='gray')
plt.axis('off')
plt.show()
```

```
cA3, (cH3, cV3, cD3) = pywt.dwt2(cA2, 'haar', mode='periodization')

plt.imshow(cA3, cmap='gray')
```

```
plt.axis('off')
plt.show()
plt.imshow(cH3, cmap='gray')
plt.axis('off')
plt.show()
plt.imshow(cV3, cmap='gray')
plt.axis('off')
plt.show()
plt.imshow(cD3, cmap='gray')
plt.axis('off')
plt.show()
```

5.1.6

```
coeffs = pywt.wavedec2(lena, 'haar', mode='periodization', level=3)
q_coeffs = coeffs

c_matrix, c_slices = pywt.coeffs_to_array(coeffs)
plt.figure(figsize=(12, 12))
plt.imshow(c_matrix, cmap='gray')
plt.axis('off')
plt.show()
```

```
cH1, cV1, cD1 = coeffs[-1]
for i in range(256):
  for j in range(256):
    cH = cH1[i][j]
    cH1[i][j] = np.round(abs(cH) / 2) * np.sign(cH) * 2

    cV = cV1[i][j]
    cV1[i][j] = np.round(abs(cV) / 2) * np.sign(cV) * 2

    cD = cD1[i][j]
    cD1[i][j] = np.round(abs(cD) / 2) * np.sign(cD) * 2
```

```
cH2, cV2, cD2 = coeffs[-2]
for i in range(128):
  for j in range(128):
    cH = cH2[i][j]
    cH2[i][j] = np.round(abs(cH) / 2) * np.sign(cH) * 2

    cV = cV2[i][j]
    cV2[i][j] = np.round(abs(cV) / 2) * np.sign(cV) * 2

    cD = cD2[i][j]
    cD2[i][j] = np.round(abs(cD) / 2) * np.sign(cD) * 2
```

```
cH3, cV3, cD3 = coeffs[-3]
for i in range(64):
  for j in range(64):
    cH = cH3[i][j]
    cH3[i][j] = np.round(abs(cH) / 2) * np.sign(cH) * 2

    cV = cV3[i][j]
    cV3[i][j] = np.round(abs(cV) / 2) * np.sign(cV) * 2

    cD = cD3[i][j]
    cD3[i][j] = np.round(abs(cD) / 2) * np.sign(cD) * 2

    cA = cA3[i][j]
    cA3[i][j] = np.round(abs(cA) / 2) * np.sign(cA) * 2
```

```
q_coeffs[-1] = cH1, cV1, cD1

q_coeffs[-2] = cH2, cV2, cD2

q_coeffs[-3] = cH3, cV3, cD3

q_coeffs[0] = cA3
```

```
a,_ = pywt.coeffs_to_array(q_coeffs)
plt.figure(figsize=(12, 12))
plt.imshow(a, cmap='gray')
plt.axis('off')
plt.show()
```

```
recon = pywt.waverec2(q_coeffs,'haar')
```

```
def PSNR(original, compressed):
    mse = np.mean((original - compressed) ** 2)
    if(mse == 0):  # MSE is zero means no noise is present in the signal
 .
                # Therefore PSNR have no importance.
        return 100
    max_pixel = 255.0
    psnr = 20 * np.log10(max_pixel / sqrt(mse))
    return psnr
```

```
psnr = PSNR(lena, recon)
```