Homework 1 Written Questions

Instructions

- 6 questions.
- Include code, images, and equations where appropriate.
- Please make this document anonymous.
- When you are finished, compile this document to a PDF and submit it directly to Gradescope. On upload, **Gradescope will ask you to assign question numbers to your pages**. Making each question end with a page break after your answer is a good way to ease this process.
- This assignment is **fixed length**, and the pages have been assigned for you in Gradescope. As a result, **please do NOT add any new pages**. We will provide ample room for you to answer the questions. If you *really* wish for more space, please add a page *at the end of the document*.
- **We do NOT expect you to fill up each page with your answer.** Some answers will only be a few sentences long, and that is okay.

Questions

Q1: Image convolution, a type of image filtering, is a fundamental image processing tool that you will use repeatedly throughout the course.

(a) *Explicitly describe* the 3 main components of image convolution: (5-10 sentences)

(i) input

(ii) transformation (how it happens)

(iii) output

(b) Why is image convolution important in Computer Vision? Which applications does it allow? (5-10 sentences)

A1: Your answer here.

(a) (i) The input should be an image and a filter. The image is 2D array contain pixel values and filter is a 2D array contain integers.

(ii) We compute each pixel with its neighborhood according to the filter. Basically, we multiply each value in the filter with the corresponding value in the image and sum together the products to get the new value for the pixel in the center. The function is $h[m,n] = \sum f[k,l] * I[m-k, n-l]$

(iii) The output is a calculated 2D array containing pixel values, with the same array size as the original image.

(b) The image convolution is the basic operation in convolutional neural networks. Learning convolutional kernels allow us to learn which 'features' provide useful information in the image. Using filters, we can smooth the image, get rid of noise in the image, perform tilt-shift photography digitally.

Q2: Correlation and convolution are both filtering operations we can perform to extract information from images.

(a) What is the difference between convolution and correlation? (3-5 sentences)

(b) Construct a scenario which produces a different output between both operations. Include the kernel you used and your image results. Use your understanding of convolution and correlation to explain the outputs. (5-10 sentences)

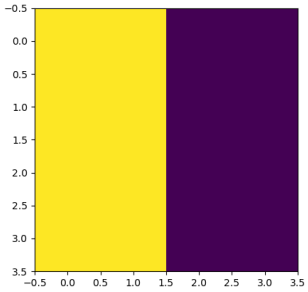
Please use scipy.ndimage.convolve and scipy.ndimage.correlate to experiment!

A2: Your answer here.

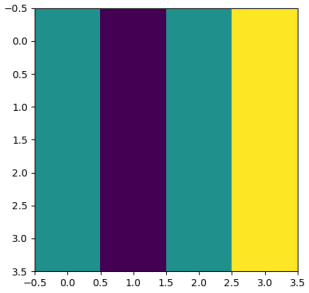
(a) The compute function is different. Correlation: $h[m,n] = \sum f[k,l] * I[m+k, n+l]$ and convolution: $h[m,n] = \sum f[k,l] * I[m-k, n-l]$. Correlation of two signals is the convolution between one signal with the functional inverse version of the other signal. Correlation is measurement of the similarity between two signals/sequences. Convolution is measurement of effect of one signal on the other signal. The convolution is linear.

(b) The filter I used is ([[1, 1, -1],[1, 1, -1],[1, 1, -1]], which is not a symmetric matrix.

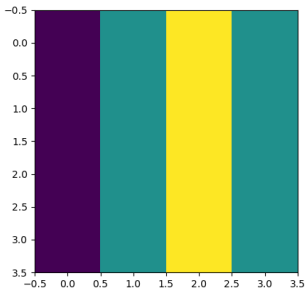
The original Image that I constructed is here:



The convolution operation image is here:



The correlation operation image is here:



The original image contains yellow on the left while purple on the right. The filter try to accentuate left neighbors over right neighbors. In the convolution, we see that filter accentuate left neighborhoods over right neighborhoods. The convolution operation and correlation operation in this case, perform a symmetric operation, which proves what we discussed in (a).

Q3: Please review the 'Thinking in Frequency Part 1' lecture slides on aliasing, and think about how we might anti-alias with a low-pass filter. Related high-pass filters also exist. For (a–c), which kind of filter does the kernel represent? For (d–e), which filter produced the output images?

(a) $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$

- ☐ High pass
- ☐ Low pass
- [✓] Neither

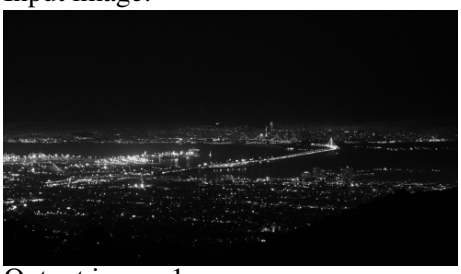
(b) $\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$

- ☐ High pass
- [✓] Low pass
- ☐ Neither

(c) $\begin{bmatrix} -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \\ -\frac{1}{9} & \frac{8}{9} & -\frac{1}{9} \\ -\frac{1}{9} & -\frac{1}{9} & -\frac{1}{9} \end{bmatrix}$

- [✓] High pass
- ☐ Low pass
- ☐ Neither

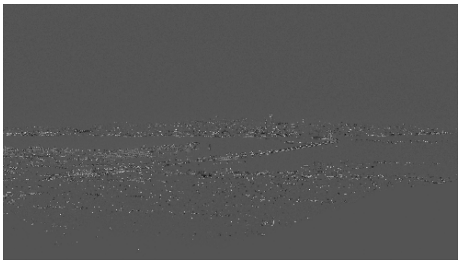
(d) Input image:



Output image 1:

- ☐ High pass
- [✓] Low pass

(e) Output image 2:



- [✓] High pass
- ☐ Low pass

(f) Which of the following statements are true? (Check all that apply).

- ☐ High pass filter kernels will always contain at least one negative number
- ☐ A Gaussian filter is an example of a low pass filter
- ☐ A high pass filter is the basis for most smoothing methods
- [✓] In a high pass filter, the center of the kernel must have the highest value

Q4:

(a) How does computation time vary with filter sizes from 3×3 to 15×15 (test odd and square sizes, i.e. 3×3, 5×5, 7×7, etc.), and with image sizes from approximately 0.25 to 8 megapixels? To find out fill out the below stencil to graph, for each of the above filter sizes, the correlation between an image size (x-axis) and time to convolve/correlate (y-axis) that image– each filter size should be its own line in the multi-line plot.

The stencil code imports the libraries you will need, but to understand how to use them you must look at the documentation.

- convolve/correlate - *scipy.ndimage.convolve* or *scipy.ndimage.correlate*
- rescale - *skimage.transform.rescale*
- resize - *skimage.transform.resize*
- rescale vs resize – an example here

Add your graph as well as a brief description of what your graph demonstrates, below.

Note A megapixel is 1,048,576 (2^{20}) pixels (1024×1024), or sometimes also 1,000,000 pixels (especially if you manufacture cameras). Megapixels is often shortened to MP or MPix.

Image: RISDance.jpg (in the .tex directory).

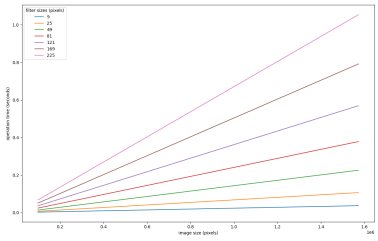
```
import time
import matplotlib.pyplot as plt
from skimage import io, img_as_float32
#use to rescale+resize image
from skimage.transform import rescale, resize
#use to convolve/correlate image
from scipy.ndimage import correlate
import numpy as np

#This reads in image and converts to a floating point format
# 1) TODO - replace PATH with the actual path to the
#    downloaded RISDance.jpg image linked above
image = img_as_float32(io.imread('./questions/RISDance.jpg'))

# 2) TODO - change the image size so it starts at 8MPix
#    use one of the imported libraries
original_image = resize(image, (4096, 2048))

# 3) TODO - iterate through odd numbers from 3 to 15
#   (inclusive!!) these will represent your filter sizes
#   (3x3,5x5,7x7, etc.), for each filter size you will...
for kernel_size in range(3, 16, 2):

    #because for each loop you are resizing your image, you
    #want to start each loop w/the original image size
    shrinking_image = original_image

    #these lists will hold the values you plot
```

```
    image_sizes = [] #x axis
    times = [] #y axis

    #while image size is bigger than .25MPx
    while(shrinking_image.size > 250000):

      # 4) TODO - create your kernel. Your kernel can hold
      #    any values, as the kernel values shouldn't
      #    affect computation time. The size of the kernel
      #    must be kernel_size x kernel_size
        kernel = np.array([[1 for _ in range(kernel_size)] for _
                                          in range(kernel_size)]
                                          )

        #5) TODO - reduce your image size. You can choose by
        # what increments to reduce your image.
        shrinking_image = rescale(shrinking_image, (0.25, 0.25, 1
                                          ))

        #gets the current time (in seconds)
        start = time.time()

        # 6) TODO - use one of the imported libraries to do
        # your correlation/convolution on the image. You can
        # choose which operation to perform.
        conv = correlate(shrinking_image, np.array([kernel,
                                          kernel, kernel]))


        #gets the current time (in seconds)
        end = time.time()

        #7) TODO - figure out what values to append, and
        #   append them here
        image_sizes.append(shrinking_image.size)
        times.append(end - start)

    #each filter size will be plotted as a separate line, in
    #a multi-line 2-dimensional graph
    plt.plot(image_sizes, times, label=str(kernel.size))

#plot
plt.xlabel('image size (pixels)')
plt.ylabel('operation time (seconds)')
plt.legend(title="filter sizes (pixels)")
plt.show()
```

(b) Do the results match your expectation given the number of multiply and add operations in convolution? (3-5 sentences)

A4: Your graph and answer here.

(a) Add your graph as well as a brief description of what your graph demonstrates.



The graph's x-axis is the size of image and y axis is the time of correlate operation, while the legend shows the size of the kernel. In this graph, we can see that as the kernel size increases, the operation time increases. Also, as the image size increases, the operation time increases.

(b) Do the results match your expectation given the number of multiply and add operations in convolution?

Yes. In the correlation and convolution operation, the more pixels the image have, the more convolution operations needed to be calculated. The larger the kernel is, the more add and multiply operation needed to be calculated in each product between kernel and each pixel.

Q5: The coding portion of this homework requires the use of the library *numpy*, which provides fast computation with large multi-dimensional arrays and matrices. Here are some small exercises on basic numpy operations to get you started! Write *one* numpy function to complete each of the following tasks.

Note that numpy is usually imported as

```
import numpy as np
```

at the top of the code file. You can then call numpy functions with

```
np.function_name()
```

You are encouraged to test out your answers by creating your own python program, importing numpy and calling and printing the results of your solutions! Some numpy functions you might find useful are np.squeeze, np.expand_dims, np.clip, np.pad, and np.zeros.

a. Create an array of shape (320,640) filled with zeros.

b. Given a variable called *img* of shape (1, 1, 320, 640), create a new 2D array variable of shape (320, 640) from *img*, i.e. remove all the 1-sized dimensions.

c. Conversely, say you have an array variable *img* of shape (320, 640), create a new 2D array variable of shape (1, 320, 640) from *img*, i.e. add a dimension.

d. Clip the image array, *img*, so all its values lie within the range [-0.5, 0.5].

e. With an RGB-image array, *img*, of shape (320, 640, 3), retrieve the blue channel of the image while preserving all of *img*'s dimensions and values.

f. With an RGB-image array, *img*, of shape (320, 640, 3), retrieve the red and blue channels of the image.

g. Pad an **RGB** (remember, this means the image has 3 channels) image array, *img*, with two zeros on either side of each row and 3 zeros values on either side of each column. Don't add zeros to the front or back (the color channel dimension).

A5: Your answer here. Remember, write *one* numpy function to complete each of the tasks—this includes operators like [] and :.

(a) np.zeros((320, 640))

(b) np.squeeze(img)

(c) np.expand_dims(img, axis=0)

(d) np.clip(img, 0.5, 0.5, out=img)

(e) img[:,:,0]

(f) img[:,:,[0,2]]

(g) np.pad(img, ((0,0), (2,2), (3,3), mode='constant')

Q6: Goal: Understand consequences of image manipulation and ways to mitigate them

In 1990, *New York Times* photography critic Andy Grundberg stated that:
"In the future, readers of newspapers and magazines will probably view news pictures more as illustrations than as reportage, since they can no longer distinguish between a genuine image and one that has been manipulated."

(a) When is Grundberg's 'future'? Why? (2–4 sentences)

(b) When is a news picture no longer genuine? Are any manipulations permissible, and if so, which ones? (2–4 sentences)

(c) Who benefits from image manipulation in the news? Who is negatively affected? How? (2–3 sentences)

(d) If you worked for the *New York Times* and were tasked with maintaining readers' trust in news pictures in Grundberg's future, what would you do? Consider everything that happens in the publication of a news picture. Describe your approach, and why it would work. (3–5 sentences)

(e) Include an cited example of a manipulated news picture, and identify the manipulation. What was the intent and impact of the manipulation? (2–4 sentences)a

Note: These are open questions. We will grade for thoughtfulness and justification.

A6: Your answer here

(a) I think now is the Grundberg's 'future', because nowadays, people can easily manipulate their pictures using PS, TikTok and many other apps. People can even create fake pictures using 'GAN'.

(b) In terms of a picture itself, when a news picture's event is manipulated to show a different story from its original one, the news picture is no longer genuine. I think when people an anonymously post messages online through social media and use apps to manipulate the pictures, a news picture is no longer genuine. Some manipulations are permissible. For example, sometimes media needs to make people's faces mosaic to protect privacy.

(c) People who controls media benefits such as bloggers, reporters, editors and etc, because they can modify the content of the pictures. People who are characters in the news are negatively affected, because they are impacted by rumors.

(d) First, we need to set up rules to punish those pictures that modify the truth of the event. I need to encode the original pictures that were taken by reporters or from any original source in an protected way. Before the newspaper is published, we need to check the consistency between the original ones and the ones on the newspaper. It will work, since the original picture is encoded when it is taken, so if the picture is modified or manipulated, the encoding will be different.

A6: Extra page.

(e) In 1997, there was terrorism happening in Egypt and the water in front of the Olympics Stadium was changed to red like blood. This manipulation add horror into the picture and negatively impacted people's feelings of the stadium and frightened the local residents there. link

Feedback? (Optional)

Please help us make the course better. If you have any feedback for this assignment, we'd love to hear it!

In terms of homework, sometimes the way of ticking the multi-choice question change, I hope the proper ticking instruction is added right before the question.