CS 4476 Project 1

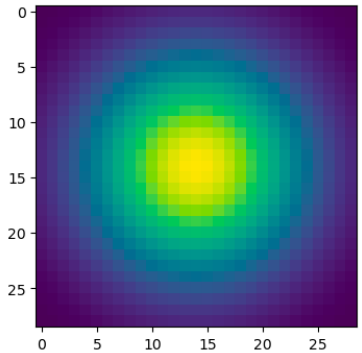
[Seohee Yoon]
[syoon333@gatech.edu]
[syoon333]
[903763900]

Part 1: Image filtering

[insert visualization of Gaussian kernel from project-1.ipynb here]

1D:

2D:



[Describe your implementation of my_conv2d_numpy() in words. Make sure to discuss padding, and the operations used between the filter and image.]

First of all, to prevent an image from shrinking during convolution, I calculated how much padding is required. We can get it by simply doing floor divide filter width and height by 2 so that we can locate the filter on the desired place of the image. And pad the input image, using np.pad. We need to pad with zeros, I used ‘constant’. And because the shape is image is 3 dimension (one is color channel), we need to reshape our filter as 3d to match the dimensions of the padded image. And declared filtered_image with value 1 with the same shape as the image in parameter. Lastly, Loop over the pixels in the image and apply the filter. It extracts a region of interest from the padded image and computes the element-wise multiplication with the filter. The result is summed along both axes to get the value for the filtered image.

Part 1: Image filtering

Identity filter

[insert the results from project-1.ipynb using 1b_cat.bmp with the identity filter here]

Small blur with a box filter

[insert the results from project-1.ipynb using 1b_cat.bmp with the box filter here]

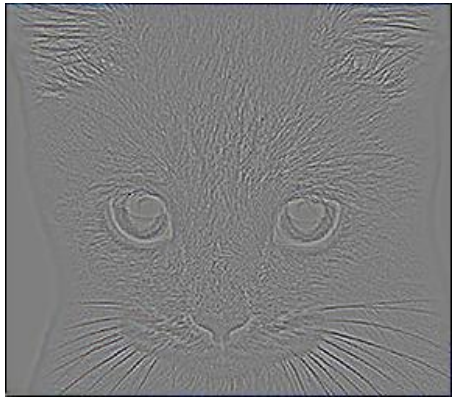
Part 1: Image filtering

Sobel filter

[insert the results from project-1.ipynb using 1b_cat.bmp with the Sobel filter here]

Discrete Laplacian filter

[insert the results from project-1.ipynb using 1b_cat.bmp with the discrete Laplacian filter here]



Part 1: Hybrid images

[Describe the three main steps of create_hybrid_image() here. Explain how to ensure the output values are within the appropriate range for matplotlib visualizations.]

1. Created low frequency image using my_conv2d_numpy .

2. Create high frequency image by subtracting the result of my_conv2d_numpy with image2 from image2.

3. Added two images. During adding, used np.clip() to make the pixel values of the hybrid image are between 0 and 1.

Cat + Dog

[insert your hybrid image here]

Cutoff frequency: 7

Part 1: Hybrid images

Motorcycle + Bicycle

[insert your hybrid image here]

Cutoff frequency: 3

Plane + Bird

[insert your hybrid image here]

Cutoff frequency: 5

Part 1: Hybrid images

Einstein + Marilyn

[insert your hybrid image here]

Cutoff frequency: 2

Submarine + Fish

[insert your hybrid image here]

Cutoff frequency: 3

Part 2: Hybrid images with PyTorch

Cat + Dog

[insert your hybrid image here]

Motorcycle + Bicycle

[insert your hybrid image here]

Part 2: Hybrid images with PyTorch

Plane + Bird

[insert your hybrid image here]

Einstein + Marilyn

[insert your hybrid image here]

Part 2: Hybrid images with PyTorch

Submarine + Fish

[insert your hybrid image here]

Part 1 vs. Part 2

[Compare the run-times of Parts 1 and 2 here, as calculated in project-1.ipynb. Which method is faster?]

```
Part 1: 7.298 seconds

Part 2: 0.184 seconds

Using PyTorch is faster
```

Part 3: Understanding input/output shapes in PyTorch

[Consider a 1-channel 5x5 image and a 3x3 filter. What are the output dimensions of a convolution with the following parameters?
Stride = 1, padding = 0? (1, 1, 3, 3)
Stride = 2, padding = 0? (1, 1, 2, 2)
Stride = 1, padding = 1? (1, 1, 5, 5)
Stride = 2, padding = 1?] (1, 1, 3, 3)

[What are the input & output dimensions of the convolutions of the dog image and a 3x3 filter with the following parameters:
Stride = 1, padding = 0
Input: (3, 361, 410)
Output: (1, 12, 359, 408)

Stride = 2, padding = 0
Input: (3, 361, 410)
Output: (1, 12, 180, 204)

Stride = 1, padding = 1
Input: (3, 361, 410)
Output: (1, 12, 361, 410)

Stride = 2, padding = 1
Input: (3, 361, 410)
Output: (1, 12, 181, 205)

Part 3: Understanding input/output shapes in PyTorch

[How many filters did we apply to the dog image?]

Identity filter, blur filter, sobel filter, Laplacian filter
4 filters

[Section 3 of the handout gives equations to calculate output dimensions given filter size, stride, and padding. What is the intuition behind this equation?]
Output format : (1, d2, h2, w2)

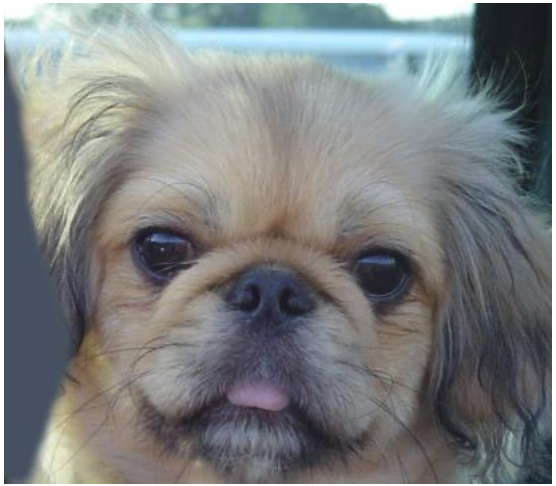
d2: the number of filters applied to the input

h2 = (input image height – filter height + 2 * padding) / stride + 1
w2 = (input image width – filter width + 2 * padding) / stride + 1

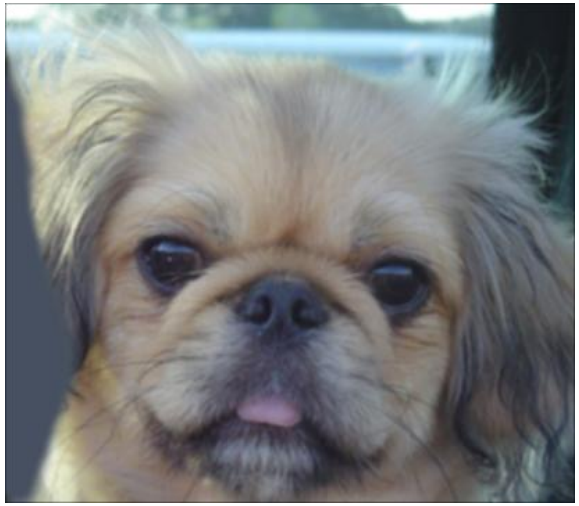
Subtract filter height from input image height/width : Image height and width after applying filter
Add 2 * padding : additional height/width by adding padding
Divide by stride : skipping positions
Add 1 : start position of the filter

Part 3: Understanding input/output shapes in PyTorch

[insert visualization 0 here]



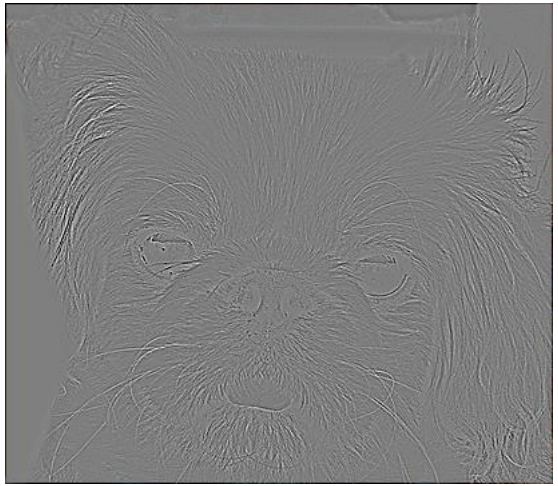
[insert visualization 1 here]



Part 3: Understanding input/output shapes in PyTorch

[insert visualization 2 here]

[insert visualization 3 here]



Part 4: Frequency Domain Convolutions

[Insert the visualizations of the dog image in the spatial and frequency domain]

[Insert the visualizations of the blurred dog image in the spatial and frequency domain]

Part 4: Frequency Domain Convolutions

[Insert the visualizations of the 2D Gaussian in the spatial and frequency domain]

[Why does our frequency domain representation of a Gaussian not look like a Gaussian itself? How could we adjust the kernel to make these look more similar?]

Part 4: Frequency Domain Convolutions

[Briefly explain the Convolution Theorem and why this is related to deconvolution]

Part 4: Frequency Domain Convolutions

[Insert the visualizations of the mystery image in the spatial and frequency domain]

[Insert the visualizations of the mystery kernel in the spatial and frequency domain]

Part 4: Frequency Domain Convolutions

[Insert the de-blurred mystery image and its visualizations in the spatial and frequency domain]

[Insert the de-blurred mystery image and its visualizations in the spatial and frequency domain after adding salt and pepper noise]

Part 4: Frequency Domain Convolutions

[What factors limit the potential uses of deconvolution in the real world? Give two possible factors]

[We performed two convolutions of the dog image with the same Gaussian (one in the spatial domain, one in the frequency domain). How do the two compare, and why might they be different?]

Conclusion

[How does varying the cutoff frequency value or swapping images within a pair influences the resulting hybrid image?]

When value of cutoff frequency was 7, the result image looked more like a high frequency image, and the characteristics of the low frequency image were difficult to find. For example, the hybrid image using motorcycle and bicycle, with 7 cutoff frequency, the image of bicycle is too clear to see motorcycle even though I saw the image far away However, the hybrid image using cat and dog, with 7 cutoff frequency, I can find both characteristics of dog and cat depending on the distance.

Motorcycle & bicycle at 7 cuttoff frequency

Cat & Dog at 7 cuttoff frequency