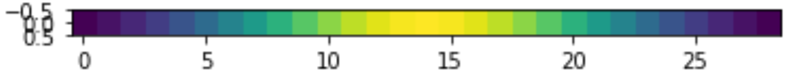
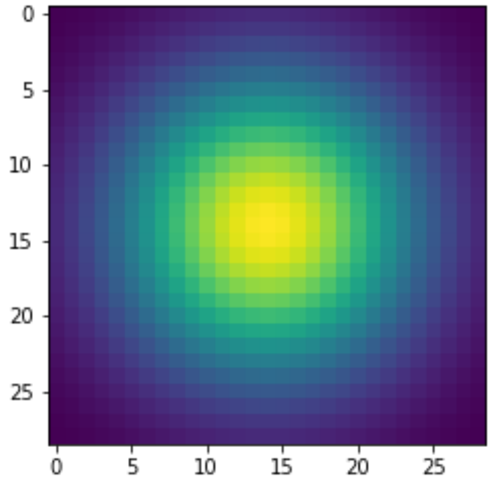
CS 6476 Project 1

[Zixin Yin]
[zyin81@gatech.edu]
[zyin81]
[903718320]

Part 1: Image filtering

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





]

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

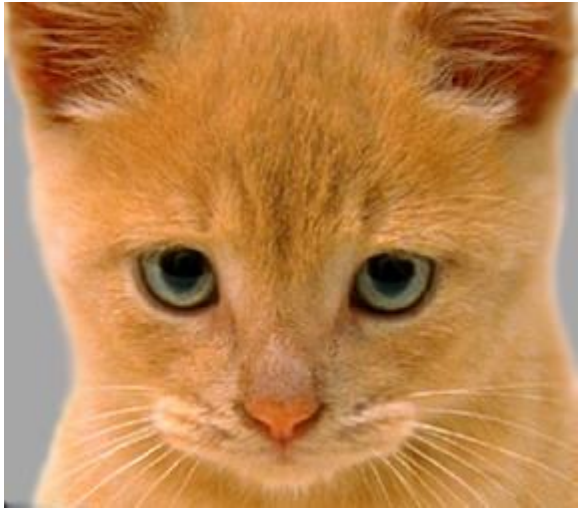
I added paddings of width of (filter.shape[0]-1)/2 to both the left and right of the image and paddings of height of (filter.shape[1]-1)/2 to both the top and bottom of the image.

For every channel in each pixel of the image, I sliced matrices of size of the filter, with the target pixels in the center, and multiplied the sliced matrices element-wise with filter. Then I summed every calculated element in the sliced matrices, and assigned the summary values to each channel of pixels in a matrix of the same size as image and return it as filtered_image.

Part 1: Image filtering

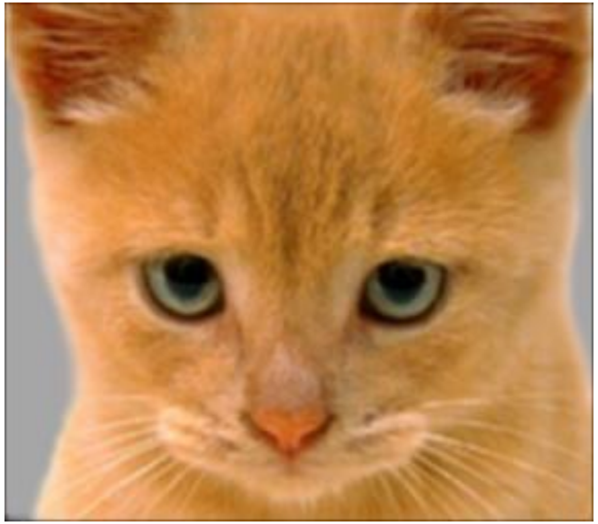
Identity filter

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



Small blur with a box filter

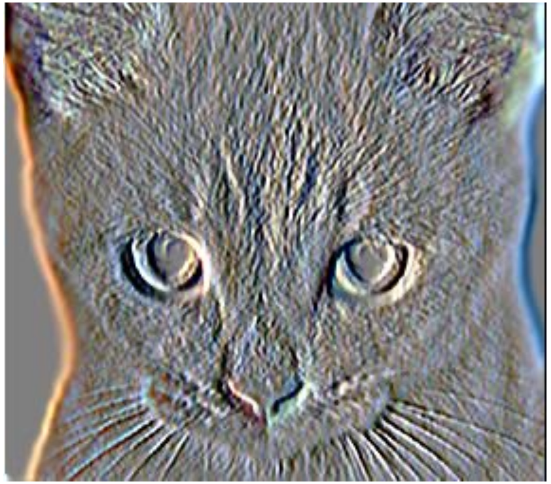
[insert the results from project-1.ipynb using 1b_cat.bmp with the box filter here]



Part 1: Image filtering

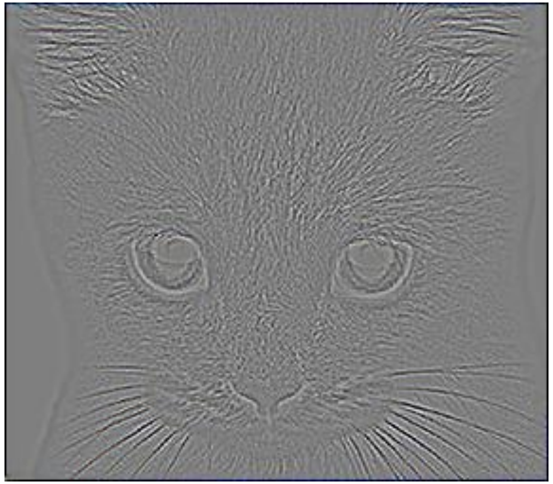
Sobel filter

[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.]

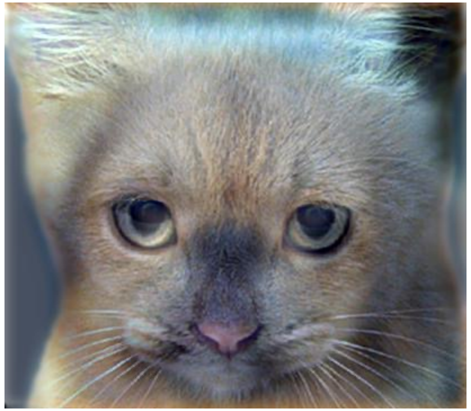
1. Create low frequency image with my_conv2d_numpy(image1, filter)

2. Create high frequency image with image element-wise subtracting the result of my_conv2d_numpy(image2, filter)

3. Sum low frequency and high frequency images element-wise, and clip the result using [0,1] range.

Cat + Dog

[insert your hybrid image here]

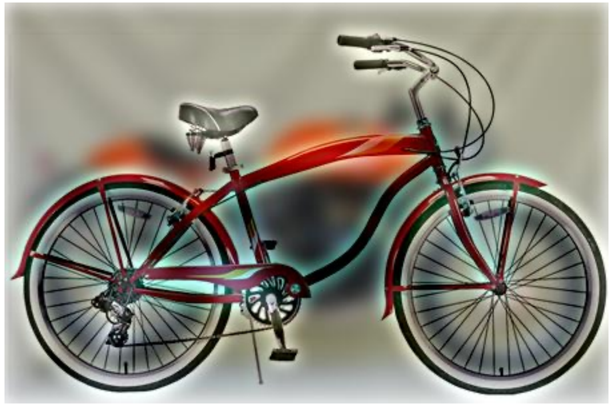


Cutoff frequency: [insert the value you used for this image pair] 7

Part 1: Hybrid images

Motorcycle + Bicycle

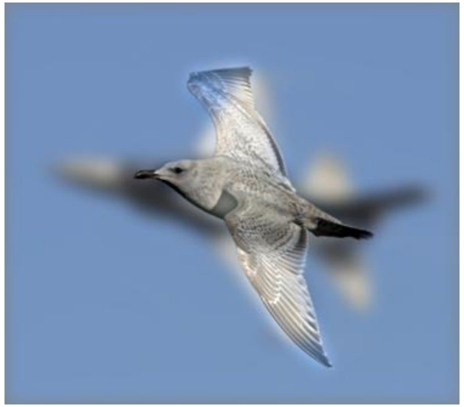
[insert your hybrid image here]



Cutoff frequency: [insert the value you used for this image pair]

Plane + Bird

[insert your hybrid image here]

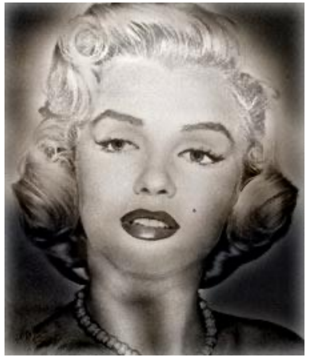


Cutoff frequency: [insert the value you used for this image pair]

Part 1: Hybrid images

Einstein + Marilyn

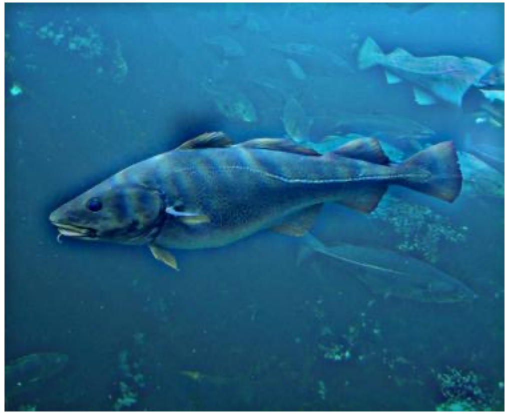
[insert your hybrid image here]



Cutoff frequency: [insert the value you used for this image pair]

Submarine + Fish

[insert your hybrid image here]

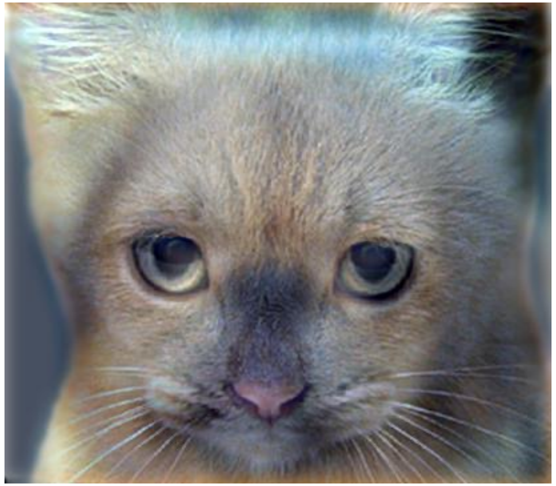


Cutoff frequency: [insert the value you used for this image pair]

Part 2: Hybrid images with PyTorch

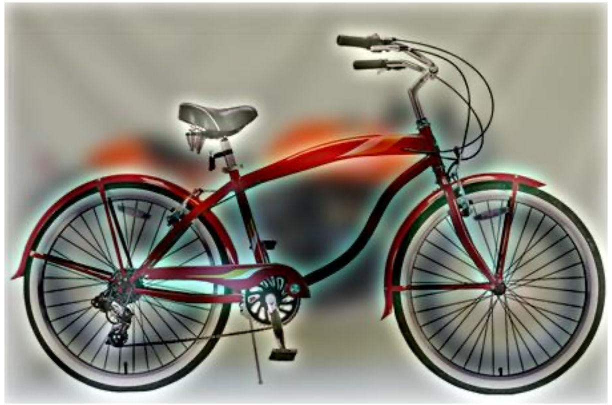
Cat + Dog

[insert your hybrid image here]



Motorcycle + Bicycle

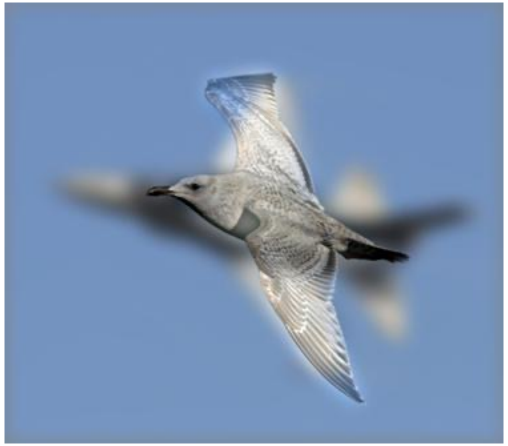
[insert your hybrid image here]



Part 2: Hybrid images with PyTorch

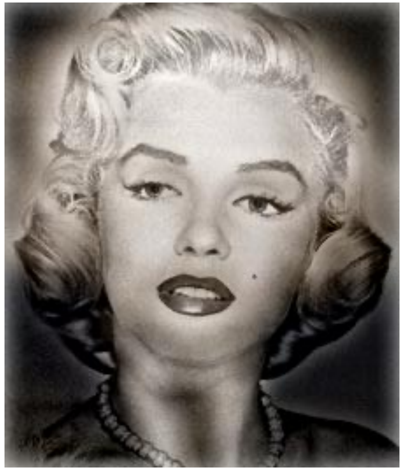
Plane + Bird

[insert your hybrid image here]



Einstein + Marilyn

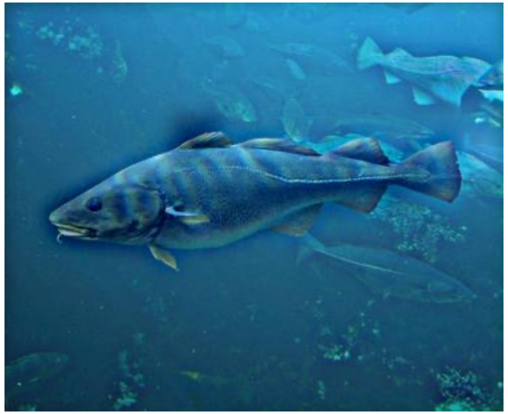
[insert your hybrid image here]



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?]

For part1, I have run time of 6.442 seconds.

For part2, I have run time of 1.956 seconds.

PyTorch runs 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 [1,1,359,408]
Stride = 2, padding = 0 [1,1,183,204]
Stride = 1, padding = 1 [1,1,361,410]
Stride = 2, padding = 1?] [1,1,181,205]

Part 3: Understanding input/output shapes in PyTorch

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

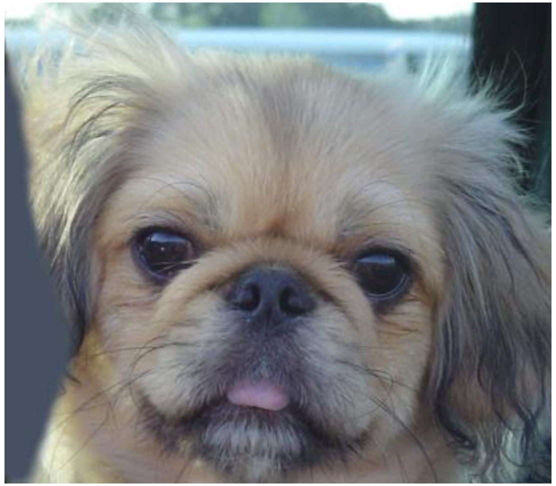
12

[Section 3 of the handout gives equations to calculate output dimensions given filter size, stride, and padding. What is the intuition behind this equation?]

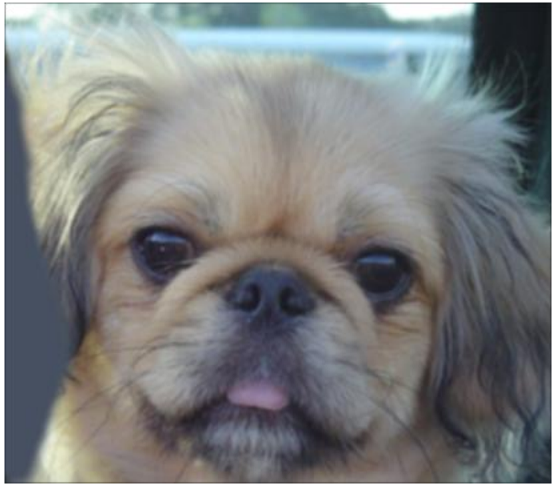
The intuition is calculating the number of times we can place the filter size in each row and column in padded image with certain stride value. Essentially the number of times we need to slice out a matrix with the size of filter and do the multiplication.

Part 3: Understanding input/output shapes in PyTorch

[insert visualization 0 here]



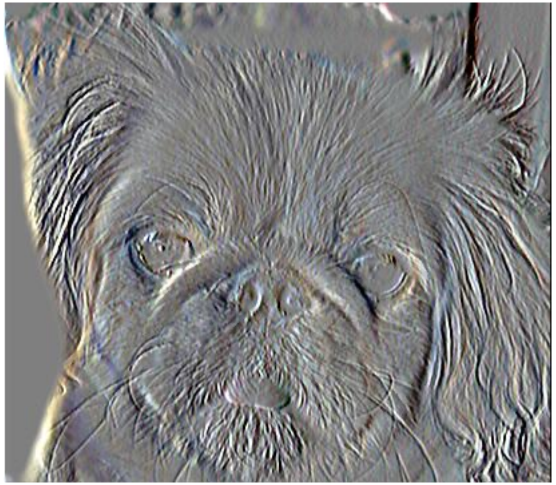
[insert visualization 1 here]

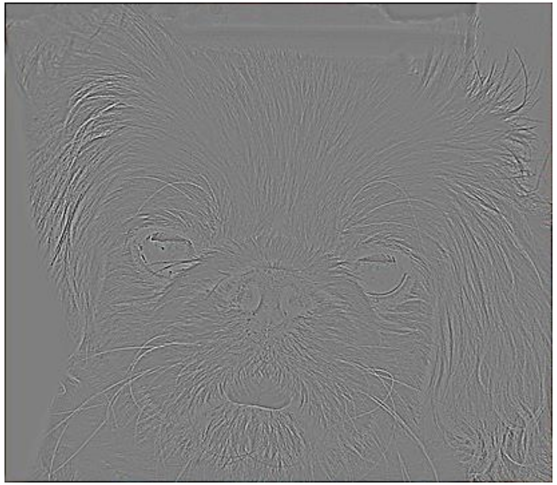


Part 3: Understanding input/output shapes in PyTorch

[insert visualization 2 here]

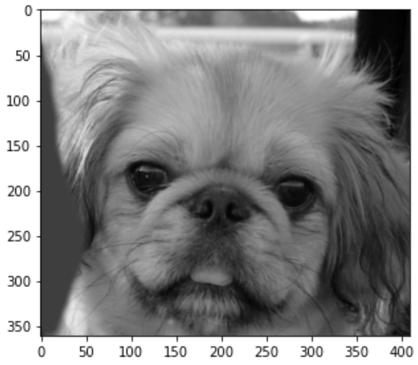
[insert visualization 3 here]

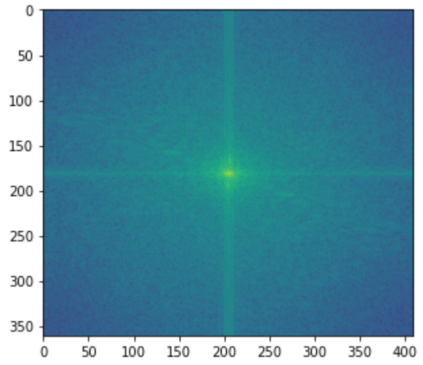




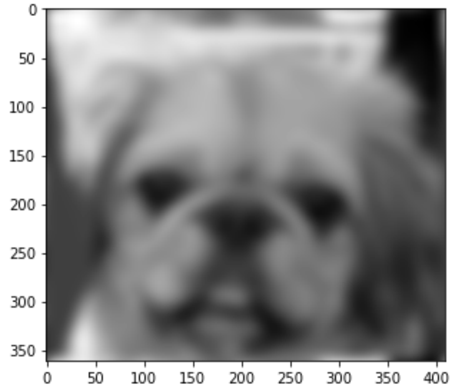
Part 4: Frequency Domain Convolutions

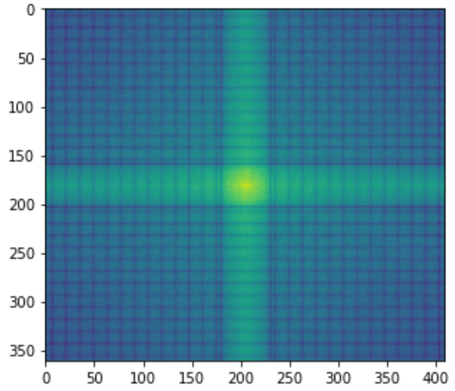
[Insert the visualizations of the dog image in the spatial and frequency domain]





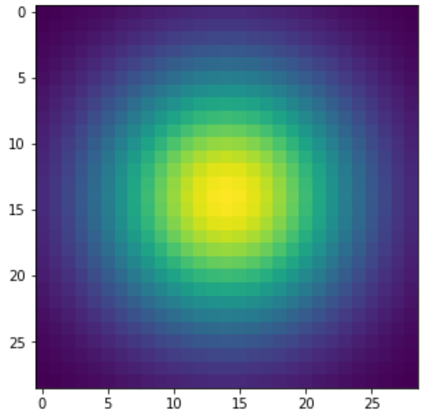
[Insert the visualizations of the blurred dog image in the spatial and frequency domain]

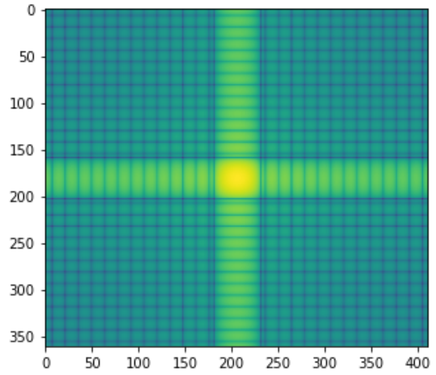




Part 4: Frequency Domain Convolutions

[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?]

If we the spatial domain image to be continuous not discrete and then convert it to frequency domain, it might look more similar.

Part 4: Frequency Domain Convolutions

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

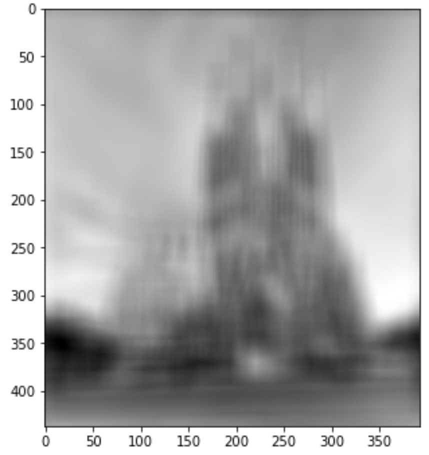
The Fourier transform of the convolution of two functions is the product of their Fourier transforms.

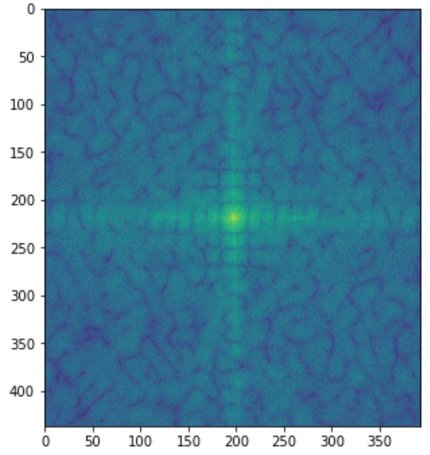
Convolution in spatial domain is equivalent to multiplication in frequency domain.

Deconvolution in spatial domain is division in frequency domain.

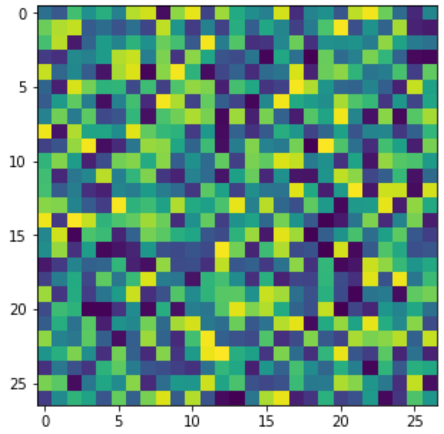
Part 4: Frequency Domain Convolutions

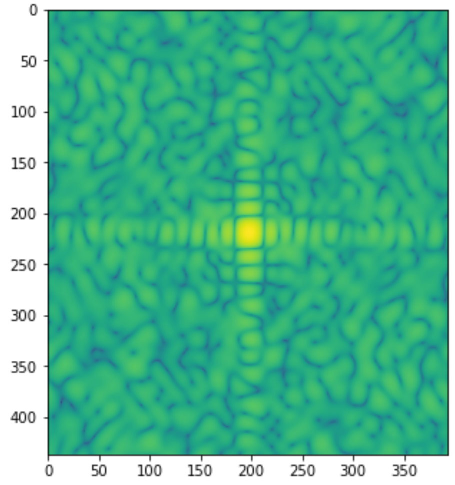
[Insert the visualizations of the mystery image in the spatial and frequency domain]





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

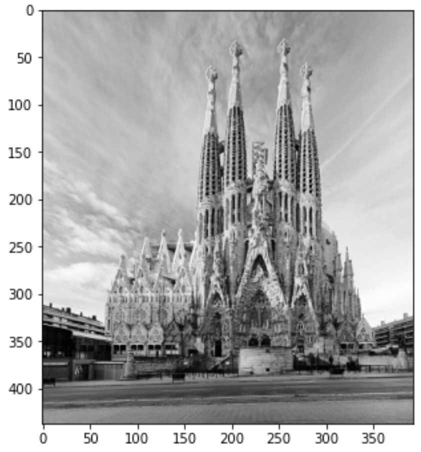


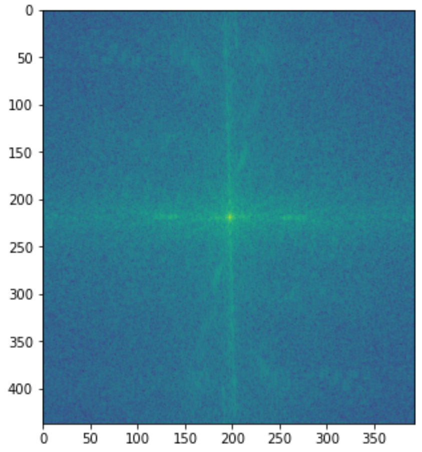


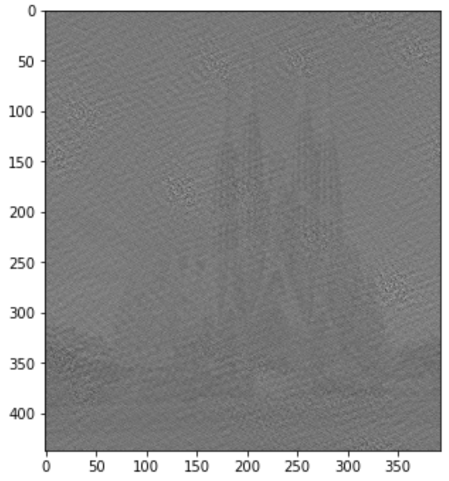
Part 4: Frequency Domain Convolutions

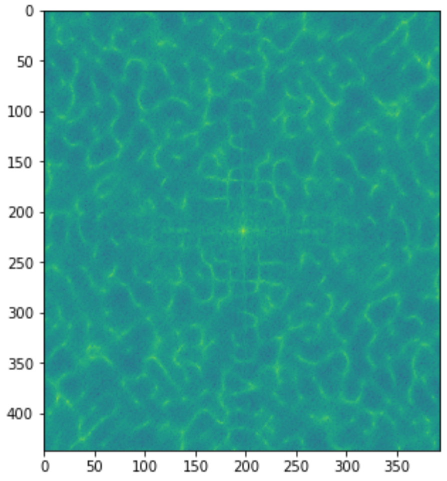
[Insert the de-blurred mystery image and its visualizations in the spatial and frequency domain]

[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]

Noise added to the convoluted image will change the result we get from deconvolution;
The convoluted image might be clipped, so we do not have access to all the details of the original image.

[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?]

In part 4(frequency domain convolution), the result looks a bit more blurry than the result in part 1(spatial domain convolution). It might be because the image in part1 has more channels to start with, so it gives more information for our brains to recognize. Or it could be because during the complicated changing from and to frequency domain, some numerical details are lost.

Conclusion

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

With a larger cutoff frequency, the low frequency image should be more prominent, and the high frequency image should be difficult to see.

Swapping images will result in change of order of images that we can see as we go further from the hybrid image.