Warning: Do not delete slides.
This includes extra credit slides and any problems you do not complete.
All problems, including extra credit, must be assigned to a slide on Gradescope. Failure to follow this will result in a penalty

CS 6476 Project 1

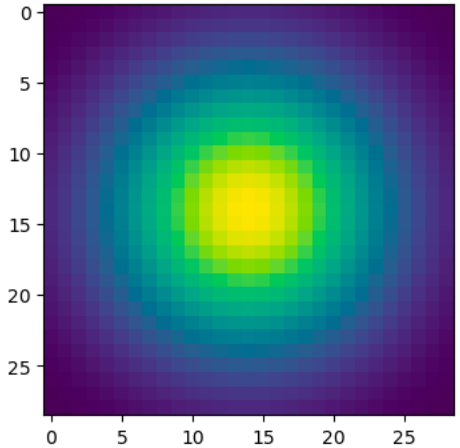
Venkata Aashutosh Aripirala
av84@gatech.edu
av84
904173725

Part 1: Image filtering

Gaussian Kernels(1D and 2D)

1D:

2D:



My implementation of the my_conv2d_numpy() function first involves padding the input image with zeroes. The amount of padding is calculated as half the filter dimension, which ensures the output image has same spatial resolution as the input. The algorithm then iterates through all the pixels and color channels of the image. At each location, the algorithm does an element-wise multiplication between the corresponding image patch and the filter, followed by a summation to generate the new pixel value.

Part 1: Image filtering

Identity filter

During the convolution operation with the identity filter, element wise multiplication multiplies the center pixel with 1, and the surrounding ones with 0. This sum is simply equal to the original center pixel's value. This reconstructs the original image perfectly identically.

Small blur with a box filter

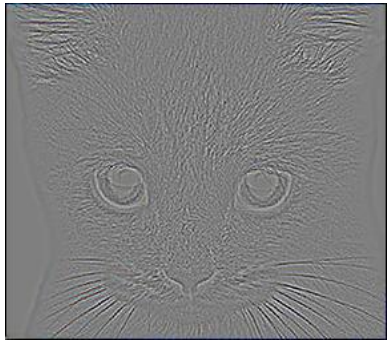
The blur box filter is basically a filter with all values equal(here, 1/9). The sum of this filter is 1, and therefore it preserves the original brightness of the image. The value for each pixel is taken as the average of its own value and its neighboring pixels, and therefore this filter smooths out the sharp intensity changes like edges and noise.

Part 1: Image filtering

Sobel filter

The Sobel filter(an edge detector) produced an image that highlights the vertical edges of the cat. This particular kernel is designed to measure the change in pixel intensity along the horizontal axis. In the output image, the bright areas correspond to light-to-dark translation, and vice-versa. Neutral gray indicates no significant vertical edge.

Discrete Laplacian filter



(a)Discrete Laplacian (b)Highpass Filter

In the Laplacian filter, it is calculating the difference between the center pixel and its 4 immediate neighbors. This operation results in a pixel value of 0 near smooth regions, but a strong response where pixel values change abruptly, leading to a large positive or large negative value, thus generating a dark or white output pixel.

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.]

First, we create a low-pass version of the first image. This is achieved by convoluting the image with the 2D Gaussian Filter. As a step 2, we create the high-pass version of the second image, by first creating its low-pass, and then subtracting it from the original image. The final hybrid image is simply created by summing the low-frequencies of img1 and high-frequencies of img2. To ensure they're visualizable, we clamp the pixel values of this hybrid image to the [0,1] range using np.clip.

Cat + Dog

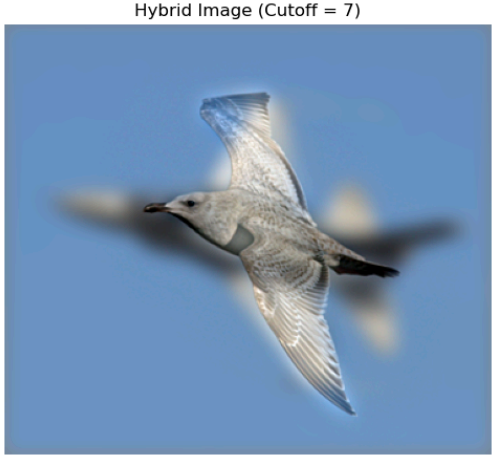
Cutoff frequency for this image: 6

Part 1: Hybrid images

Motorcycle + Bicycle

Cutoff frequency: 8

Plane + Bird



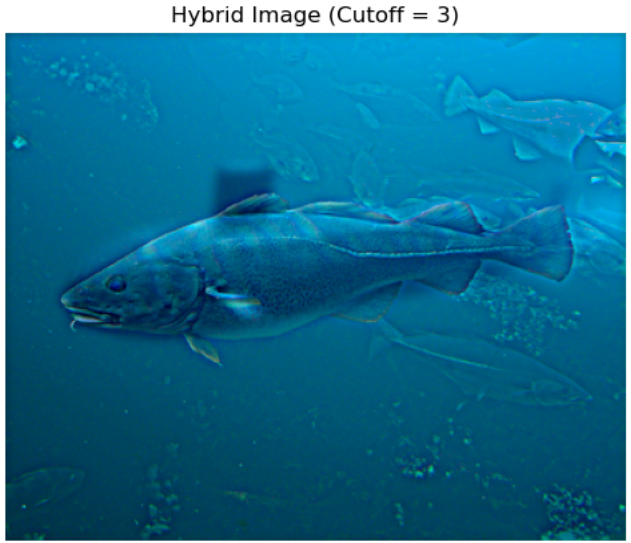
Cutoff frequency: 7

Part 1: Hybrid images

Einstein + Marilyn

Cutoff frequency: 4

Submarine + Fish



Cutoff frequency: 3

Part 2: Hybrid images with PyTorch

Cat + Dog

Motorcycle + Bicycle

Part 2: Hybrid images with PyTorch

Plane + Bird

Einstein + Marilyn

Part 2: Hybrid images with PyTorch

Submarine + Fish

Part 1 vs. Part 2

Part 1: 6.382 Seconds

Part 2: 0.192 Seconds

The PyTorch implementation is much faster(33x).

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?
The formula used for a square image(h=w):
output_dim=(h + 2*p – 3)/s + 1
Here, ”k” is the dimemsion of the filter, and s is the stride
Stride = 1, padding = 0? Ans: 3 x 3
Stride = 2, padding = 0? Ans: 2 x 2
Stride = 1, padding = 1? Ans: 5 x 5
Stride = 2, padding = 1?] Ans: 3 x 3

[What are the input & output dimensions of the convolutions of the dog image and a 3x3 filter with the following parameters:
The original dimensions of the dog image are 361x410
Stride = 1, padding = 0? Ans: 359 x 408
Stride = 2, padding = 0? Ans: 180 x 204
Stride = 1, padding = 1?Ans: 361x 410
Stride = 2, padding = 1? Ans: 181 x 205

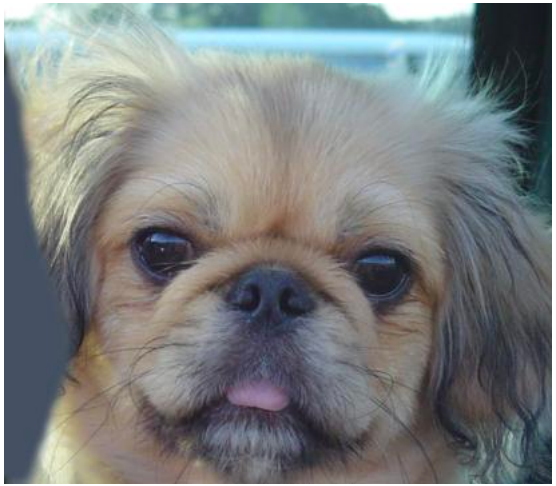
Part 3: Understanding input/output shapes in PyTorch

[How many filters did we apply to the dog image?]
Ans: 3 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?]
Ans: The intuition starts at the effective size of the image. Since we are padding the image with a dimension p, we're effectively adding p layers of border on each side(left, right, top, bottom), thus adding 2*p vertically and 2*p horizontally.

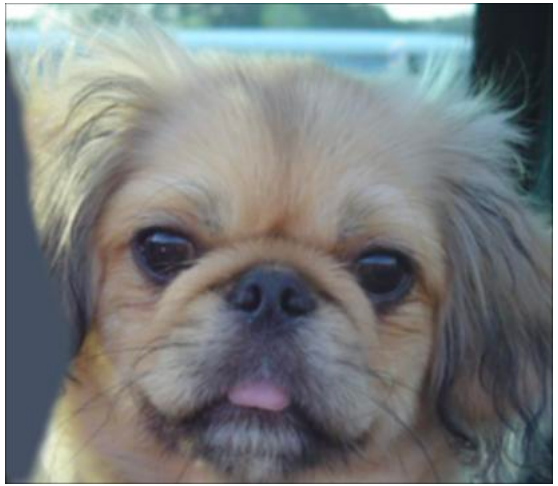
Assuming H and W to be height and width, we know we need to account for the filter height(k) and width(k) as well, being able to travel to the edges. So the travel distance becomes H + 2*p – k and W + 2*p – k along the height and width of the image. Now if we take a longer step, say of length s, we need to account for a factor of 1/s less steps as well. This would lead to our size = travel distance / stride. Now say we have a 5x5 image and a 5x5 filter. Here the travel distance is 0, but we need to account for this one position, therefore +1 should be added to (travel distance/stride).
Hence the final formula becomes: (H + 2*p – k)/s + 1 and (W + 2*p – k)/s + 1

Part 3: Understanding input/output shapes in PyTorch



What filter was applied here?

Ans: Identity Filter



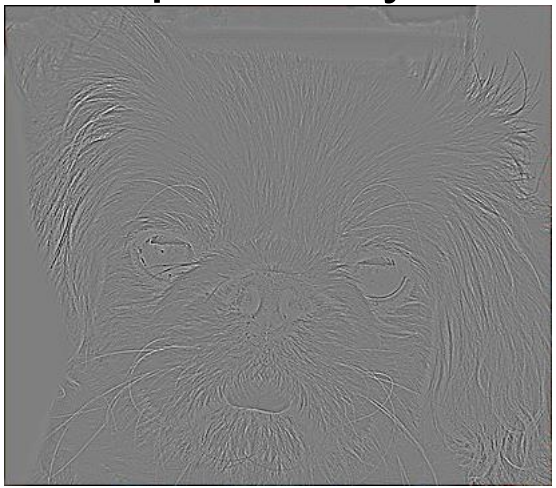
What filter was applied here?

Ans: Box Blur Filter

Part 3: Understanding input/output shapes in PyTorch

What filter was applied here?

Ans: Sobel Filter

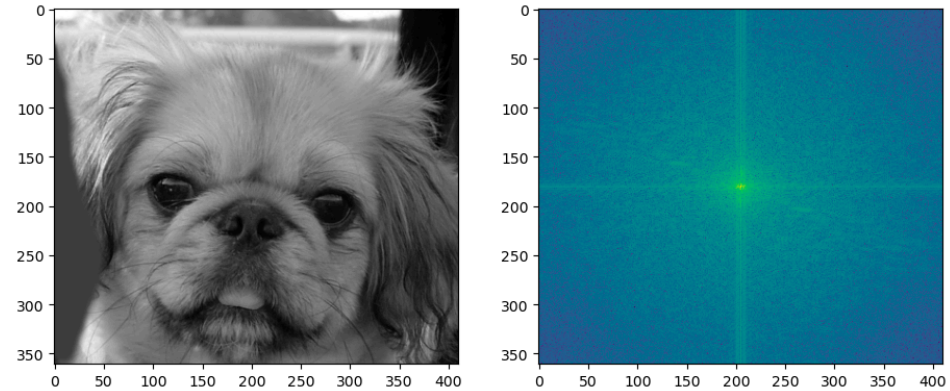


What filter was applied here?

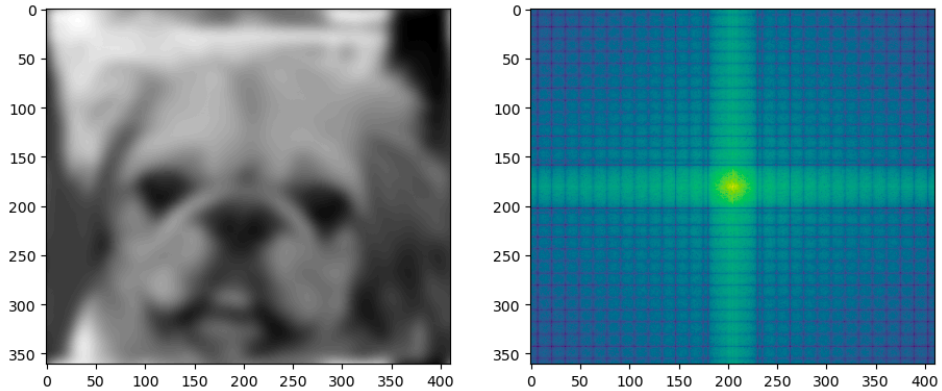
Ans: Laplacian Filter

Part 4: Frequency Domain Convolutions (extra credit)

[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]

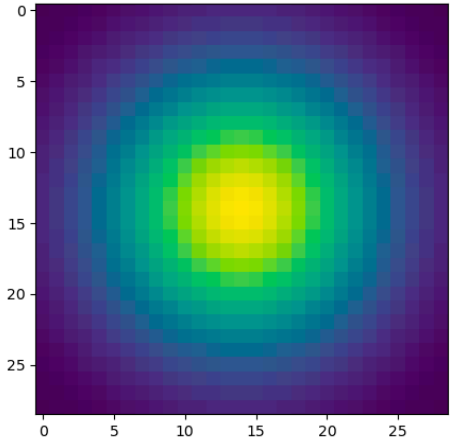


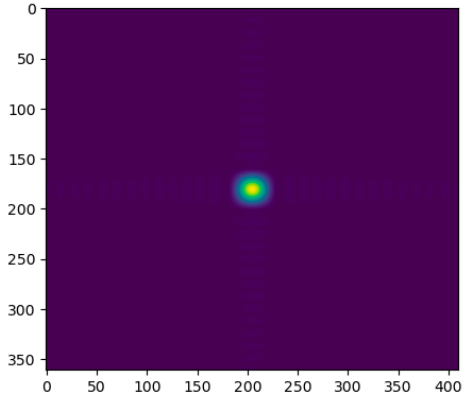
Explain the difference in the frequency domain visualizations for the original and blurred dog images:

The original image frequency shows a bright center representing the image's basic structure and the surrounding haze representing sharp details. Now the haze has almost disappeared while the center remains similar. This confirms the blur acted like a low-pass filter which preserves the image's foundational structure, but fades out the fine, high-frequency details.

Part 4: Frequency Domain Convolutions (extra credit)

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





[Try out some different cutoff values for the 2D Gaussian. What relationship do you notice between the cutoff value and the frequency domain representation? Why is that?]

There is an inverse relationship between the kernel's blurriness in the spatial domain and the size in frequency domain. A larger cutoff value creates a blurry kernel in the spatial domain, which is more spread out, averaging pixels over a wider neighborhood. A larger cutoff value in the frequency domain creates a sharp, narrow and concentrated kernel in the frequency domain, since a smooth kernel in the spatial domain can be made almost entirely of low frequencies, therefore not needing a variation in them.

Part 4: Frequency Domain Convolutions (extra credit)

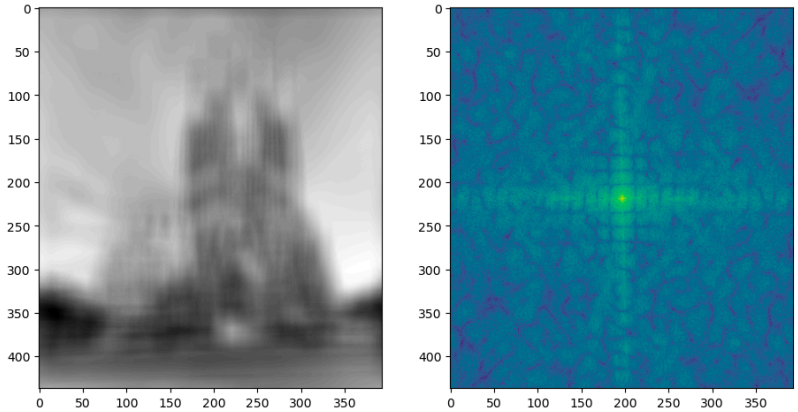
[Briefly explain the Convolution Theorem and why this is related to deconvolution]

The convolution theorem states that the expensive operation of convolution in the spatial domain is equivalent to element-wise multiplication in the frequency domain. This means that blurring an image with a filter is the same as multiplying their Fourier transforms together.

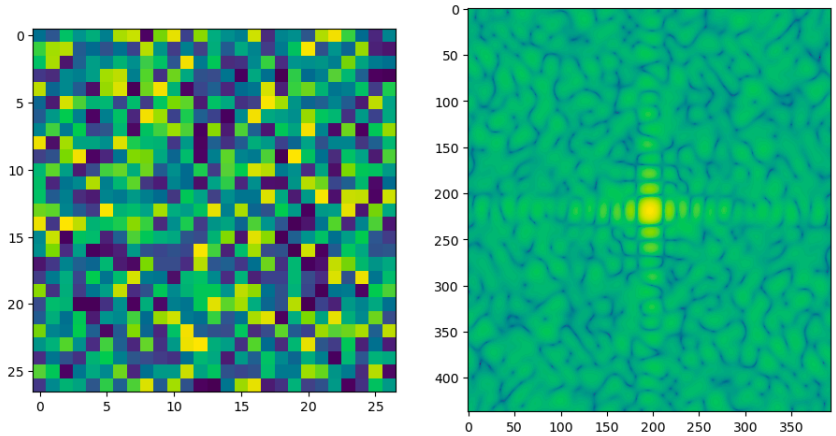
This relationship is linked to deconvolutions because of inverses. If convolution is element-wise multiplication, deconvolution is the inverse operation of division. To deblur an image, we can transform both the blurry image and the blur kernel into frequency domain, divide the image frequencies by the kernel frequencies and then transform the result back into spatial domain.

Part 4: Frequency Domain Convolutions (extra credit)

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

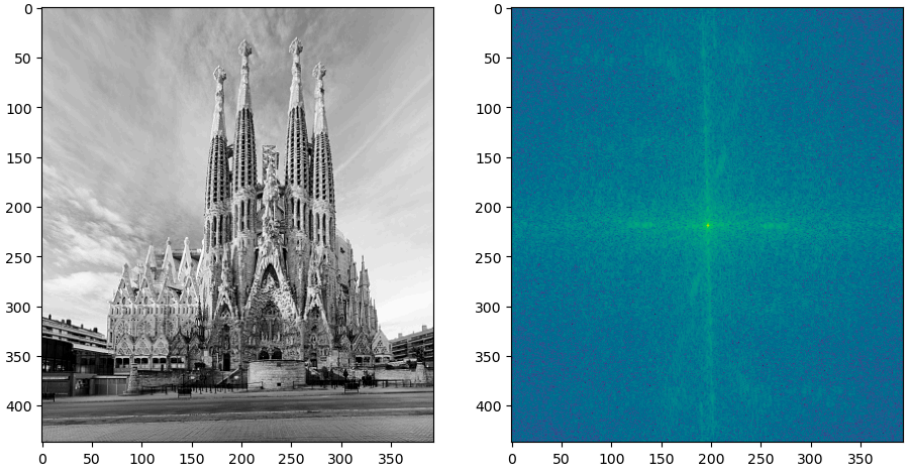


[Insert the visualizations of the mystery kernel in the spatial and frequency domain]

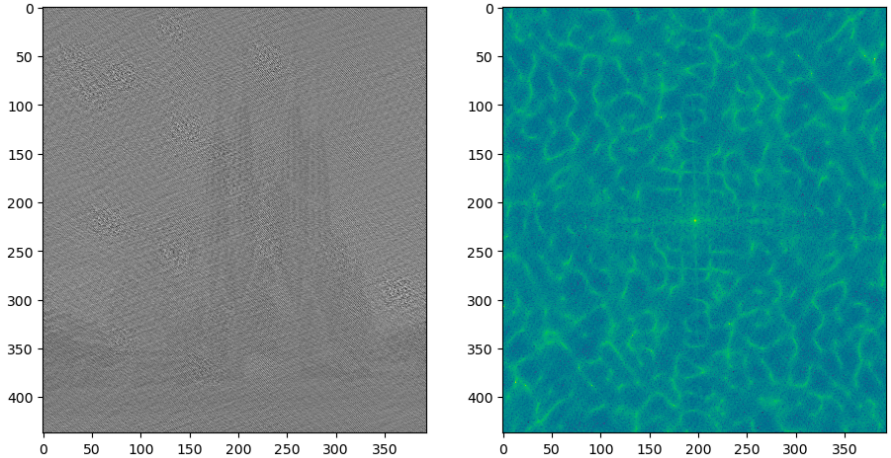


Part 4: Frequency Domain Convolutions (extra credit)

[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 (extra credit)

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

1. Unknown / Inaccurate Kernel: Even a small error in the original kernel can lead to a very poor deconvolution / de-blurring result.
2. As demonstrated in the final experiment, deconvolution is extremely sensitive to noise. Since blur kernels are low-pass filters, any high-frequency noise gets divided by these near-zero filter values, leading to massive amplification of the noise.

[Describe any structures found in the frequency domain of the mystery image and explain what it's caused by.]

The frequency domain of the blurred image, dominantly has a bright cross shape, with strong vertical and horizontal lines extending from the center. This signifies the presence of vertical and horizontal edges of the monument and the ground plane respectively. Web-like patterns radiate from the cross corresponding to mid-range frequencies of the monument's façade. While the spatial blur has suppressed the high frequencies, their signature remains visible in the frequency plot.

Conclusion

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

The cutoff frequency is the primary parameter in tuning the blurring effect. Observing the results, it looks like low-cutoff values allow the low-frequency images to retain too many of their sharp details, which interfere with the high-frequency image. High cutoff values create a strong blur, causing the high-frequency image to retain too much of its structure, which doesn't allow it to disappear properly when viewed from a distance / with a squint, thus weakening the illusion.

Swapping images within a pair results in the inverse effect. The image seen from up-close can be seen from far away, and vice versa. However, the quality of the illusion isn't seemingly symmetrical. The best outputs come when low frequency images have strong, simple and recognizable large-scale shapes, and high-frequency images are rich in fine textures and intricate details.