CS 4476 Project 1

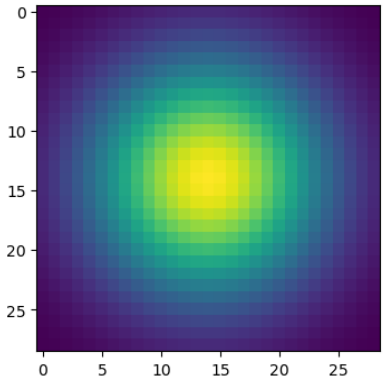
[Charles Wyner]
[cwyner3@gatech.edu]
[cwyner3]
[903687005]

Part 1: Image filtering

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

1D:

2D:



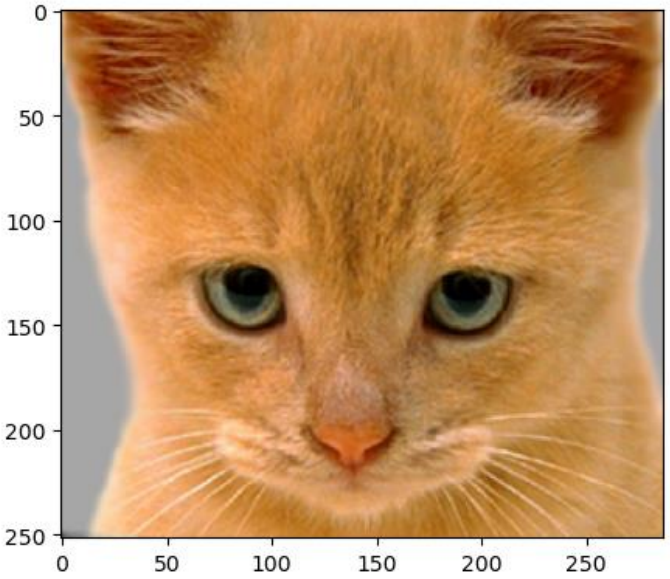
[Describe your implementation of my_conv2d_numpy() in words. Make sure to discuss padding, and the operations used between the filter and image.]

In my implementation of my_conv2d_numpy(), I basically just applied what we learned in class. For each pixel on the image, I find the region of interest and multiply the values for each pixel in the region by the corresponding value in the filter. Then I sum them. I also make sure to pad the image by the filter width and height so I can make proper calculations without going off the edge of the image.

Part 1: Image filtering

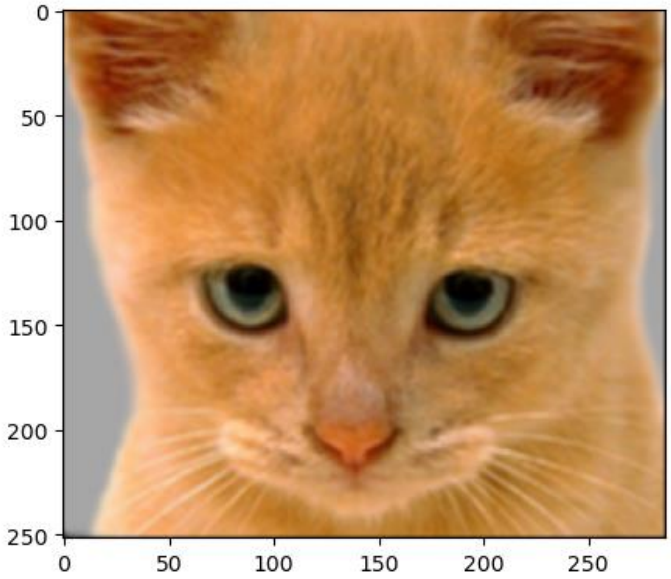
Identity filter

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



Small blur with a box filter

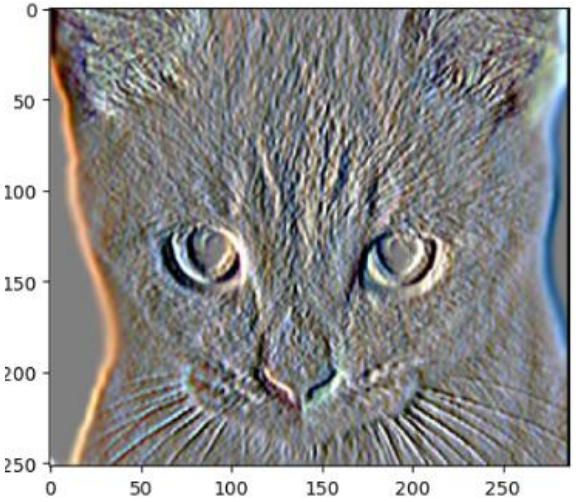
[insert the results from project-1.ipynb using 1b_cat.bmp with the box filter here]



Part 1: Image filtering

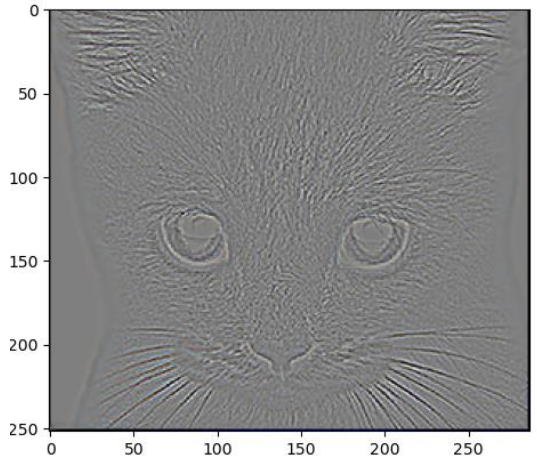
Sobel filter

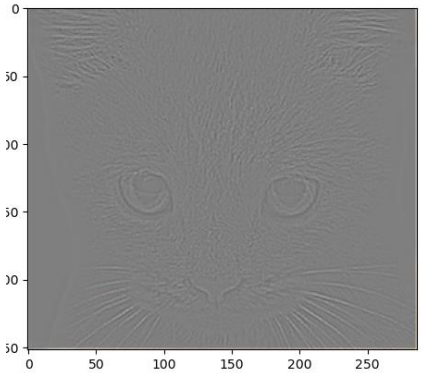
[insert the results from project-1.ipynb using 1b_cat.bmp with the Sobel filter here]



Discrete Laplacian filter

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





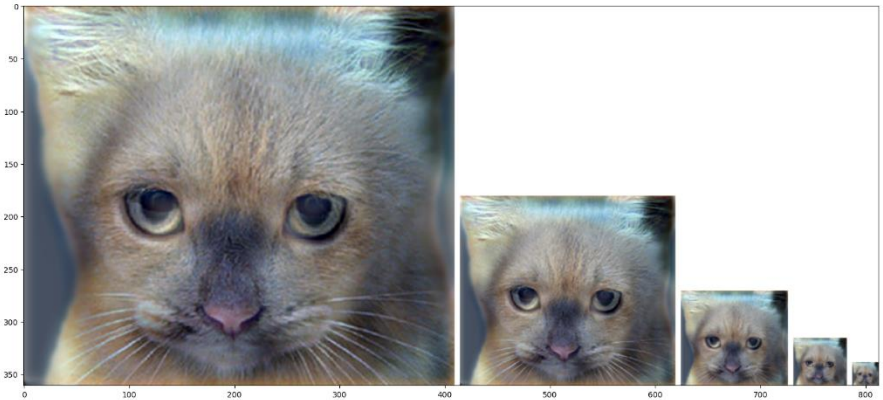
Part 1: Hybrid images

[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.]

First, I obtained the low_frequency images by applying the convolution method I made previously. Next, to obtain the high_freq images, I simply subtracted the values of the low_freq from the actual images themselves. Finally, I combined high_freq with low_freq to get a cat/dog mix.

Cat + Dog

[insert your hybrid image here]



Cutoff frequency: 7

Part 1: Hybrid images

Motorcycle + Bicycle

[insert your hybrid image here]

Cutoff frequency: 7

Plane + Bird

[insert your hybrid image here]

Cutoff frequency: 7

Part 1: Hybrid images

Einstein + Marilyn

[insert your hybrid image here]

Cutoff frequency: 7

Submarine + Fish

[insert your hybrid image here]

Cutoff frequency: 7

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?]

The PyTorch implementation was much faster (0.224s compared to 2.748s)

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? **3x3**
Stride = 2, padding = 0? **2x2**
Stride = 1, padding = 1? **5x5**
Stride = 2, padding = 1? **3x3**

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 **408x359**
Stride = 2, padding = 0 **204x180**
Stride = 1, padding = 1 **410x361**
Stride = 2, padding = 1 **205x181**

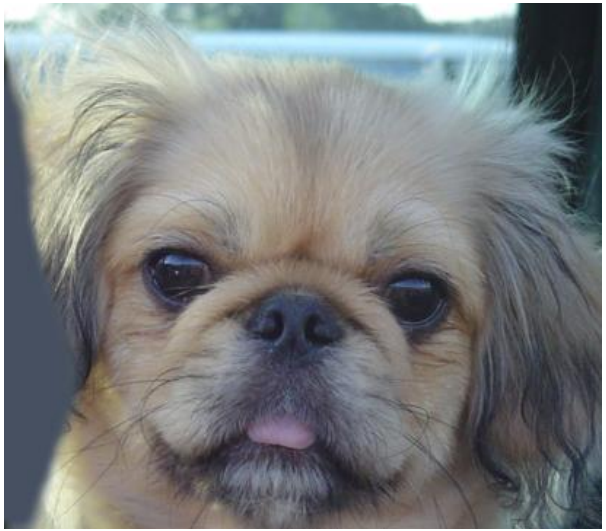
Part 3: Understanding input/output shapes in PyTorch

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

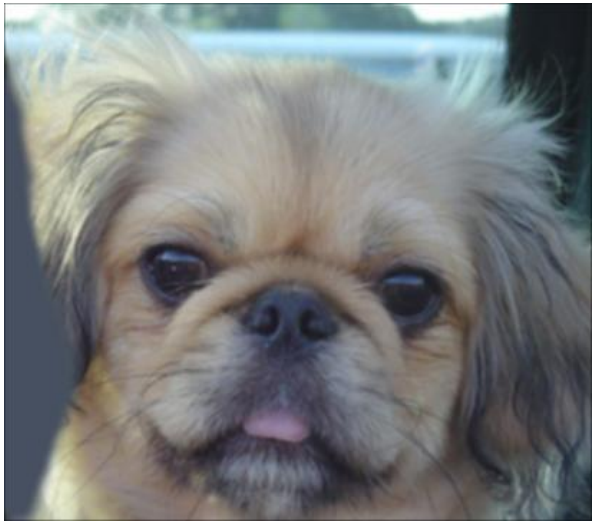
The filter size determines how big the region of interest is going to be when applying convolution. We must make sure to pad the image accordingly so that this region of interest doesn't stretch past the bounds of the image. Then, the stride determines how far the filter moves after each operation. Knowing all of these things allows us to exactly calculate the dimensions of the output image.

Part 3: Understanding input/output shapes in PyTorch

[insert visualization 0 here]



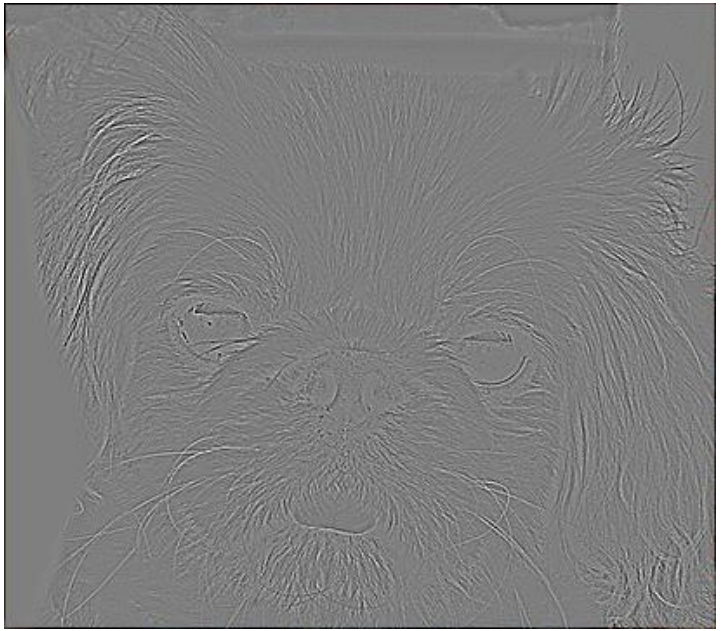
[insert visualization 1 here]



Part 3: Understanding input/output shapes in PyTorch

[insert visualization 2 here]

[insert visualization 3 here]



Part 4: Frequency Domain Convolutions (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]

Part 4: Frequency Domain Convolutions (Extra Credit)

[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 (Extra Credit)

[Briefly explain what is the Convolution Theorem]

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]

[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 and swapping images within a pair influences the resulting hybrid image?]

A low cutoff frequency means the image is blurrier. The blurrier image dominates at a far distance (or squinting eyes). The high frequency image is the more dominant when just normally looking at it, as it retains more features. Swapping these images can change which one is more dominant up close and which one is more dominant further away.