CS 6476 Project 1

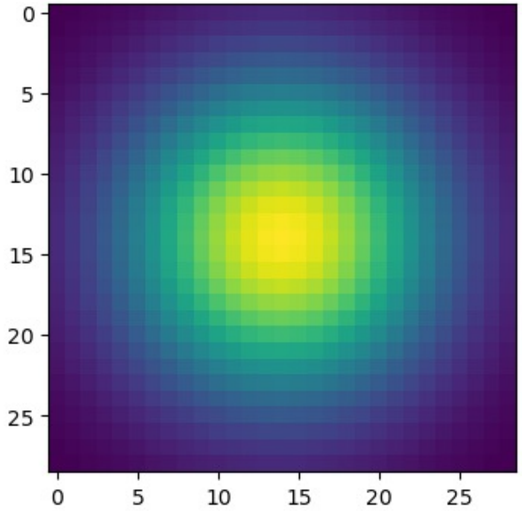
[Nakul Kuttua]
[nkuttua3@gatech.edu]
[nkuttua3]
[903520812]

Part 1: Image filtering

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

1D:

2D:

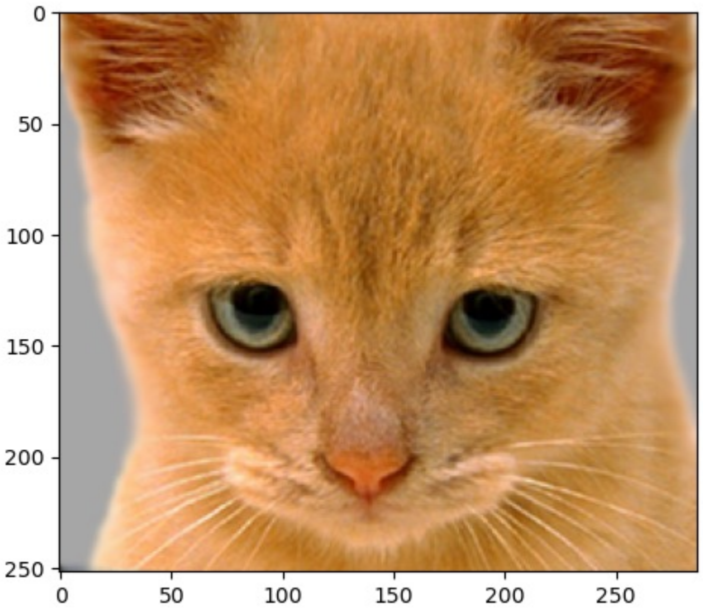


[Describe your implementation of my_conv2d_numpy() in words. Make sure to discuss padding, and the operations used between the filter and image.]

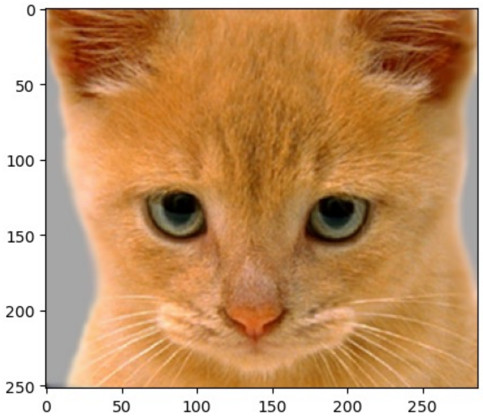
We used np.pad() and padded the image by half the filter's height and width (we did half specifically to account for extreme edge cases of the image's corners). For the filtered_image, we created an empty array of zeros using np.zeros_like with the image's array dimensions. Then using a triple nested for loop for the image's m, n, and c values, we shifted the image by the filter's shape values (k and j). The c value is left untouched. Then the filtered_image's value at each index is equal to np.sum(pad * filter). In conclusion, we have filtered an image with the given filter.

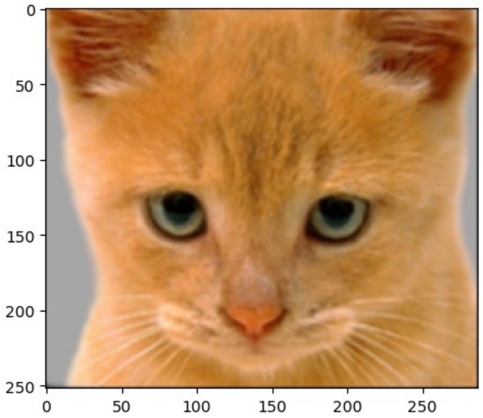
Part 1: Image filtering

Identity filter



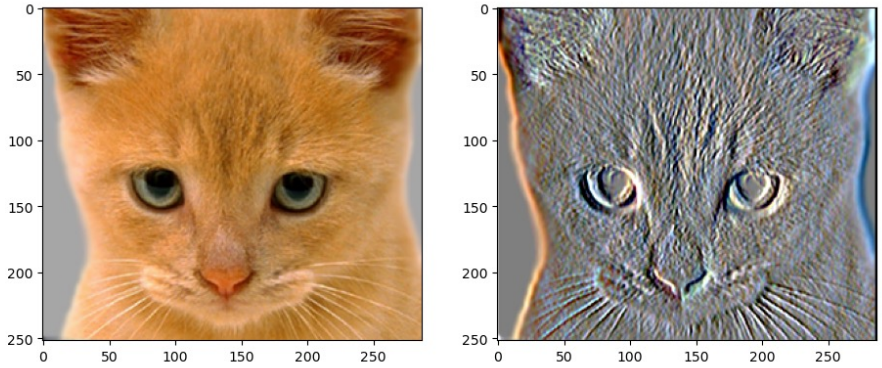
Small blur with a box filter



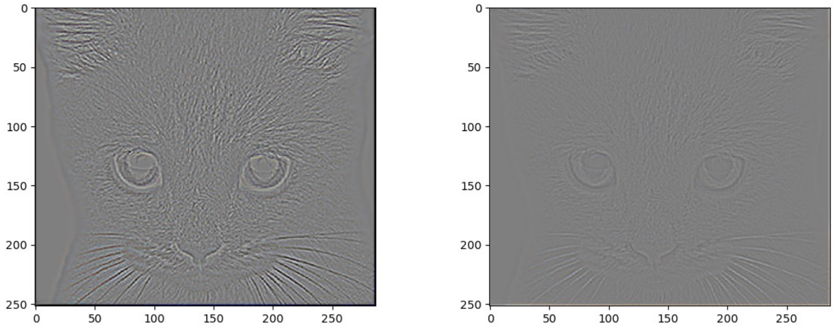


Part 1: Image filtering

Sobel filter



Discrete Laplacian filter



Part 1: Hybrid images

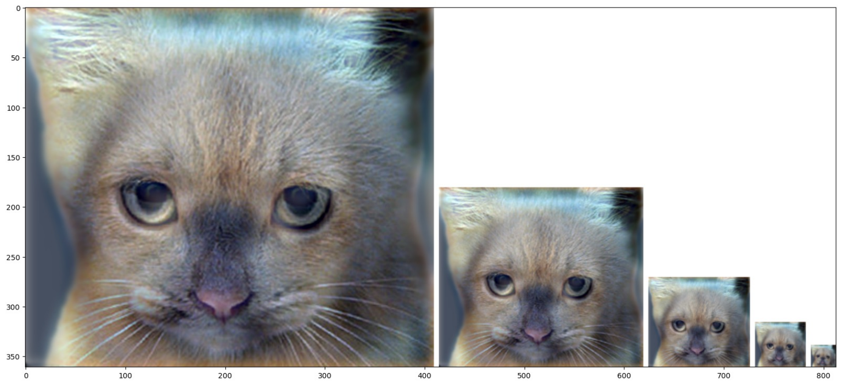
[Describe the three main steps of create_hybrid_image() here. Explain how to ensure the output values are within the appropriate range for matplotlib visualizations.]

Step 1: We get low_frequencies by applying the filter to image1 using my_conv_2d.

Step 2: We obtain high frequencies by taking image2 as it is and subtracting out its low_frequencies (using my_conv_2d(image2, filter).

Step 3: We use np.clip and give the low and high frequencies as input, and by using np.clip, we can ensure that we are restricting the pixel values to be between 0 and 1.

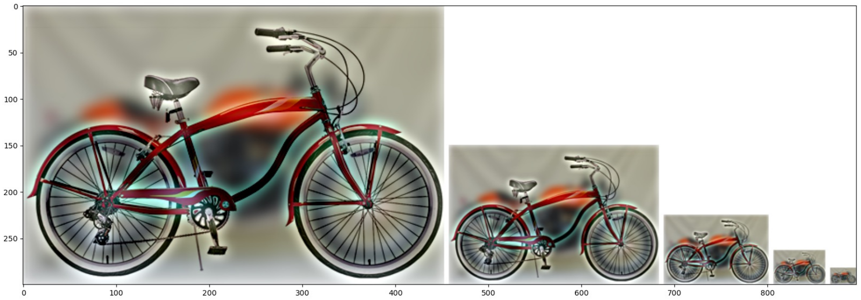
Cat + Dog



Cutoff frequency: 2

Part 1: Hybrid images

Motorcycle + Bicycle



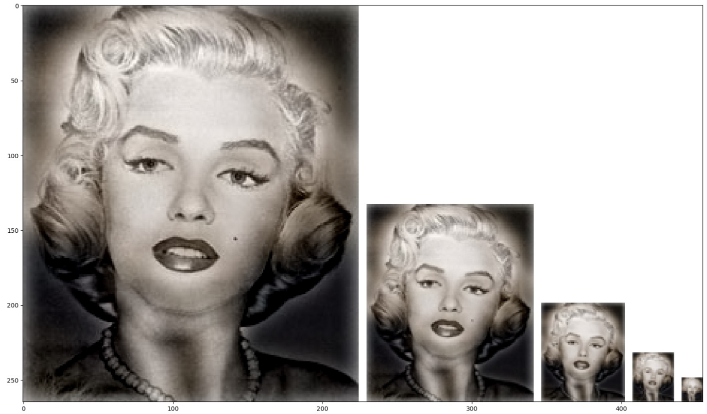
Cutoff frequency: 8

Plane + Bird

Cutoff frequency: 4

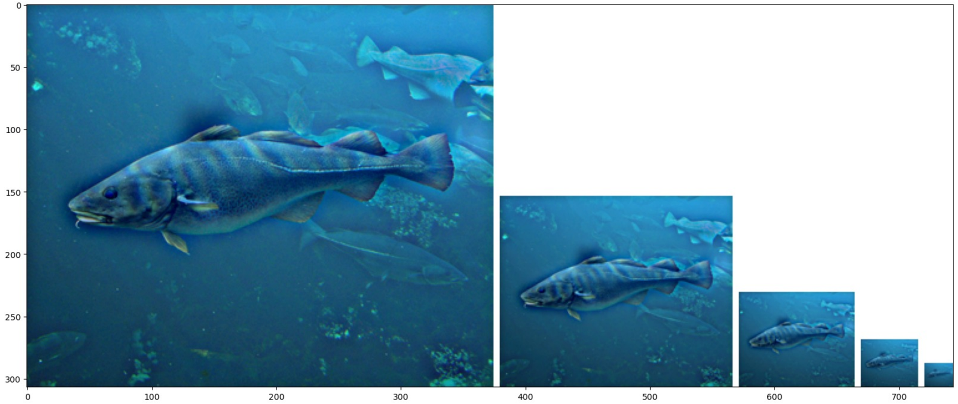
Part 1: Hybrid images

Einstein + Marilyn



Cutoff frequency: 9

Submarine + Fish



Cutoff frequency: 1

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

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

Part 1 Runtime: 2.937 seconds

Part 2 Runtime: 0.287 seconds

Part 2’s runtime is significantly 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? H = 3, W = 3
Stride = 2, padding = 0? H = 2, W = 2
Stride = 1, padding = 1? H = 5, W = 5
Stride = 2, padding = 1?] H = 3, W = 3

[What are the input & output dimensions of the convolutions of the dog image (410x361) and a 3x3 filter with the following parameters:
Stride = 1, padding = 0 W = 408, H = 359
Stride = 2, padding = 0 W = 204, H = 180
Stride = 1, padding = 1 W = 410, W = 361
Stride = 2, padding = 1?] W = 205, H = 181

Part 3: Understanding input/output shapes in PyTorch

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

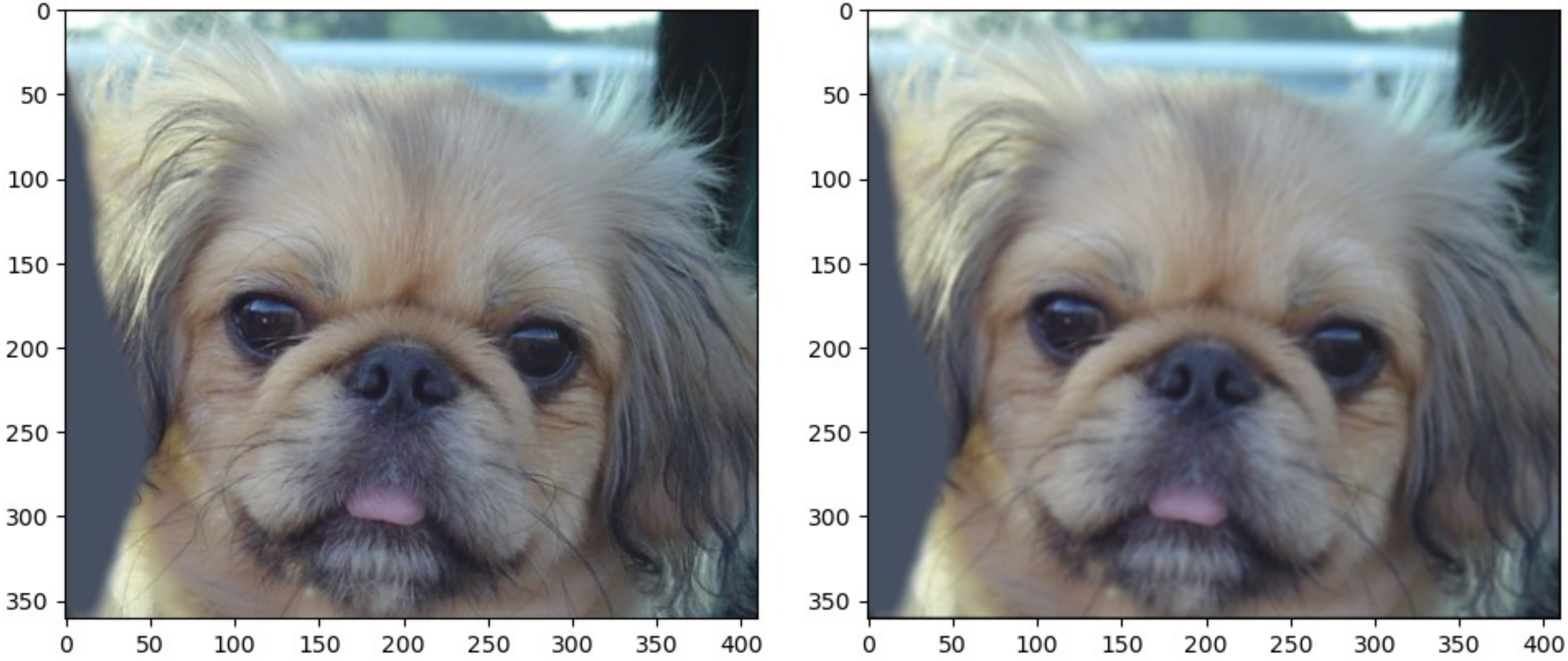
12 filtered were applied.

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

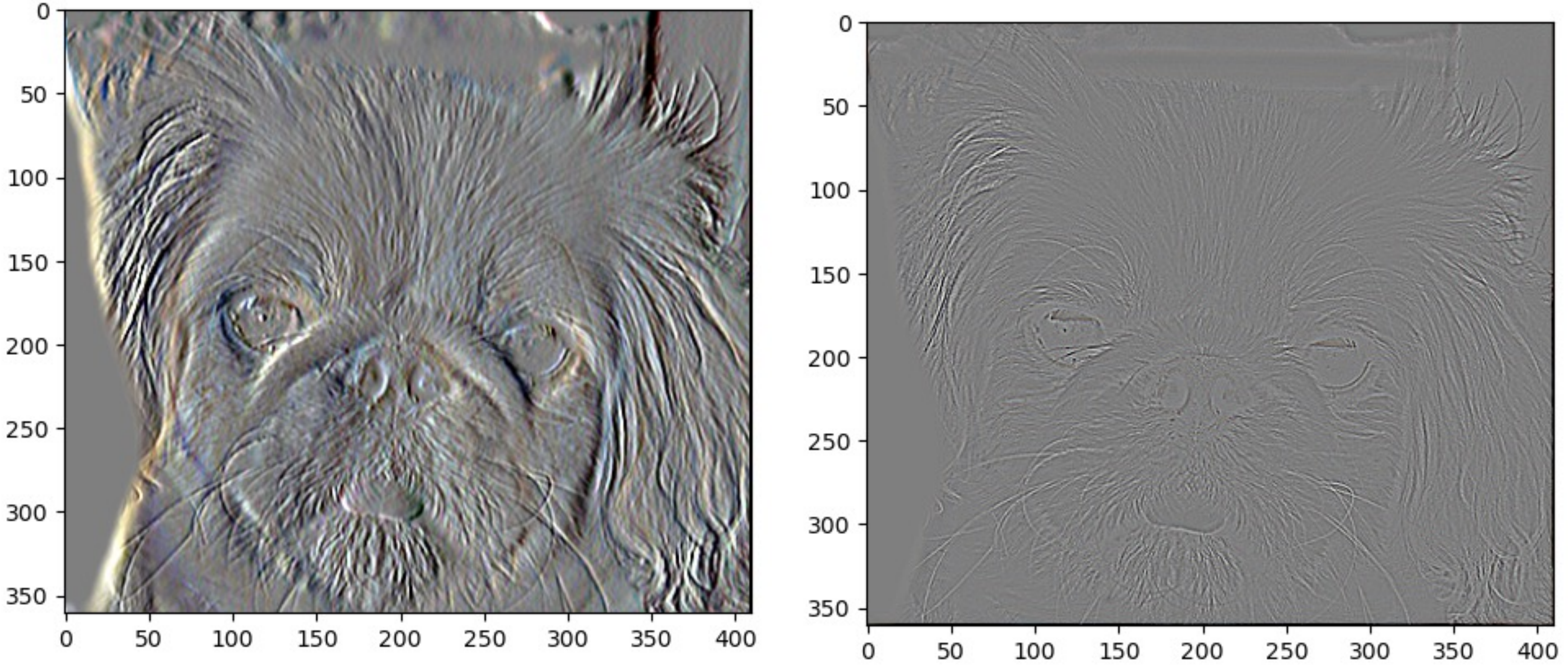
The equation for say h2 = (h1 – k+2*p)/(s) + 1 is this.

We start with the original image and substract the kernel+2*p. The 2*p accounts for padding on both sides, and we subtract the kernel from the image since the kernel will cover parts of the image. We divide by s to see how many times we can shift the kernel inside the image, and the +1 includes the starting position of the kernel since that is also a valid position.

Part 3: Understanding input/output shapes in PyTorch



Part 3: Understanding input/output shapes in PyTorch



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?]

The cutoff frequency value is proportional to how influential the low filter of the first image is in the hybrid image (higher cutoff = stronger low filter). Swapping Images within a pair puts more emphasis on the which image has a stronger low filter value.