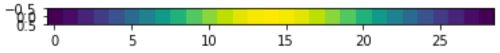
Computer Vision Project 1

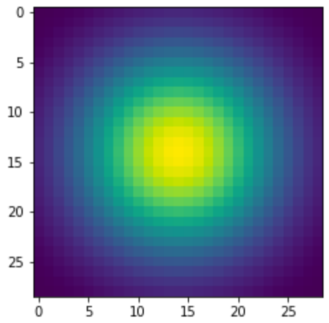
[Jinbin Bai]
[171860607]
[jinbin5bai@gmail.com]

Part 1: Image filtering

[insert visualization of Gaussian kernel from proj1.ipynb here]



Gaussian_kernel_1D



Gaussian_kernel_2D

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

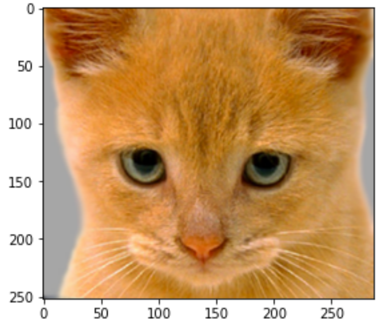
First, zero padding the image in four directions with half length or width of the filter to make sure that the size of filtered image is the same as the original image. Second, using a three-layer loop for every channel and convolutional block to convolve with the filter (using np.sum and np.multiply function). The code is as follows.

```
def my_conv2d_numpy(image: np.ndarray, filter: np.ndarray) -> np.ndarray:
    m, n, c = image.shape
    k, j = filter.shape
    padding_image = np.pad(image, ((k // 2, k // 2), (j // 2, j // 2)), 'constant')
    filtered_image = np.zeros((m, n, c))
    for C in range(c):
        for x in range(m):
            for y in range(n):
                conv_block = padding_image[x:x + k, y:y + j, C]
                filtered_image[x, y, C] = np.sum(np.multiply(conv_block, filter))
return filtered_image
```

Part 1: Image filtering

Identity filter

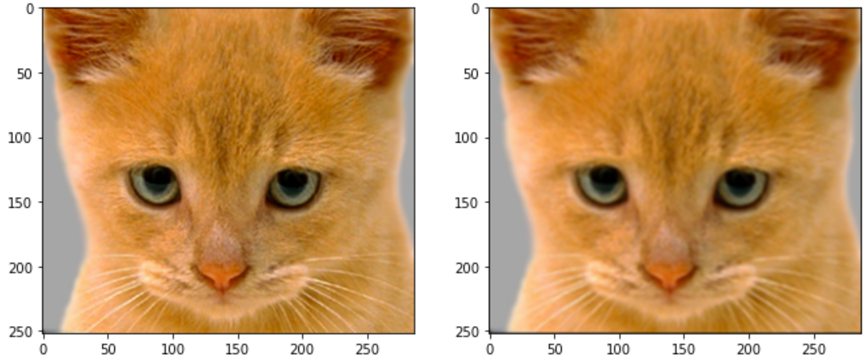
[insert the results from proj1.ipynb using 1b_cat.bmp with the identity filter here]



Identity filter

Small blur with a box filter

[insert the results from proj1.ipynb using 1b_cat.bmp with the box filter here]

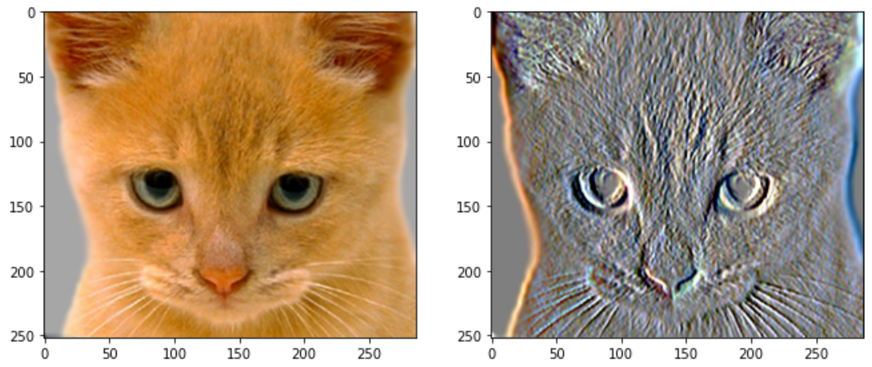


box filter

Part 1: Image filtering

Sobel filter

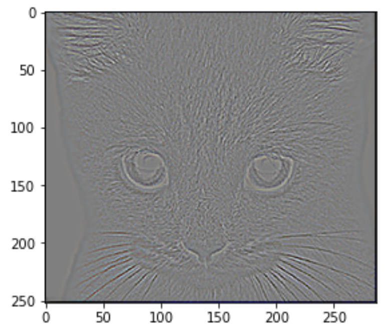
[insert the results from proj1.ipynb using 1b_cat.bmp with the Sobel filter here]



sobel filter

Discrete Laplacian filter

[insert the results from proj1.ipynb using 1b_cat.bmp with the discrete Laplacian filter here]



Discrete laplacian filter

Part 1: Hybrid images

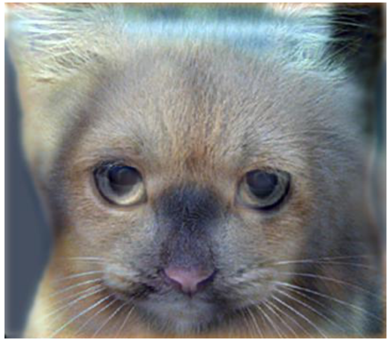
[Describe the three main steps of create_hybrid_image() here. Explain how to ensure the output values are within the appropriate range for matplotlib visualizations.]

First, convolve the first image with the low-pass filter to obtain its low frequency content. Second, subtract the second image with the convolution of itself and the low-pass filter to obtain its high frequency content. Third, add the first image's low frequency and the second image's high frequency content to obtain the hybrid image. Finally, use np.clip function to make sure the pixel values of the hybrid image are between 0 and 1. The code is as follows.

```
low_frequencies = my_conv2d_numpy(image1, filter)
high_frequencies = image2 - my_conv2d_numpy(image2, filter)
hybrid_image = low_frequencies + high_frequencies
hybrid_image = np.clip(hybrid_image, 0.0, 1.0)
```

Cat + Dog

[insert your hybrid image here]

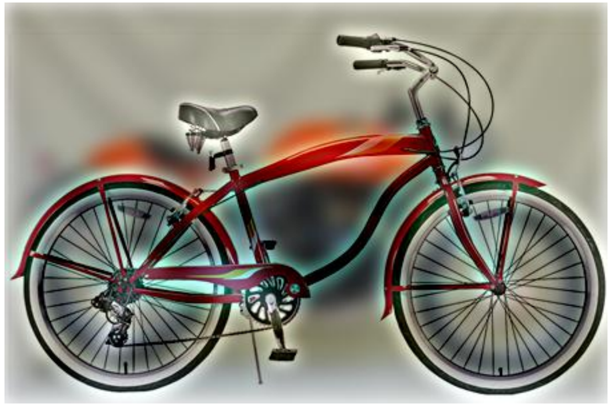


Cutoff frequency: [5]

Part 1: Hybrid images

Motorcycle + Bicycle

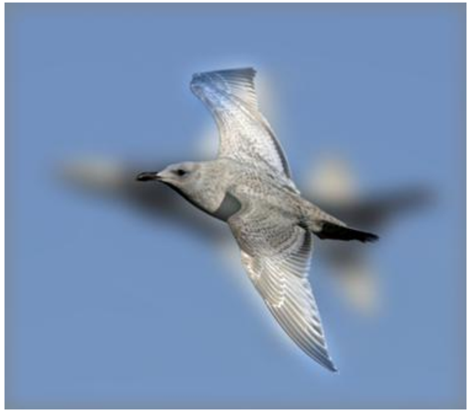
[insert your hybrid image here]



Cutoff frequency: [7]

Plane + Bird

[insert your hybrid image here]

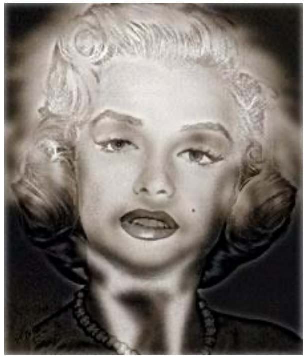


Cutoff frequency: [9]

Part 1: Hybrid images

Einstein + Marilyn

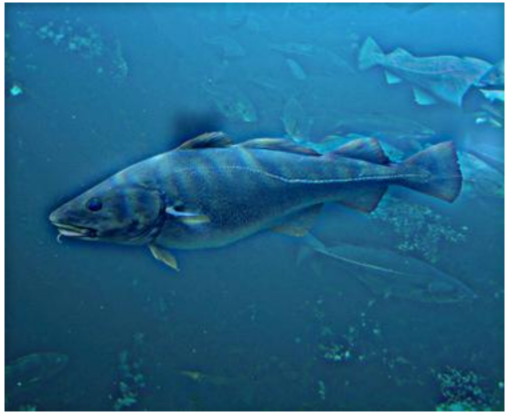
[insert your hybrid image here]



Cutoff frequency: [3]

Submarine + Fish

[insert your hybrid image here]

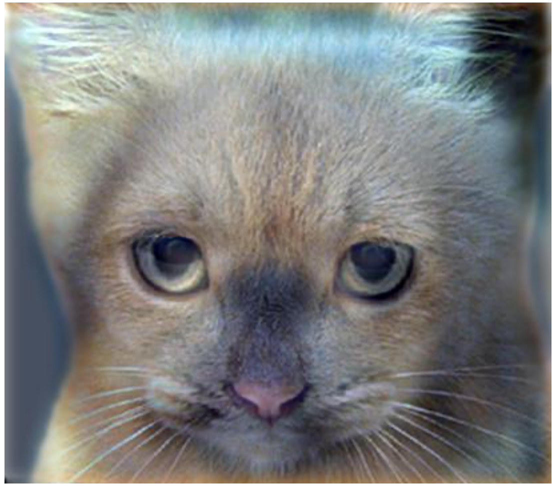


Cutoff frequency: [5]

Part 2: Hybrid images with PyTorch

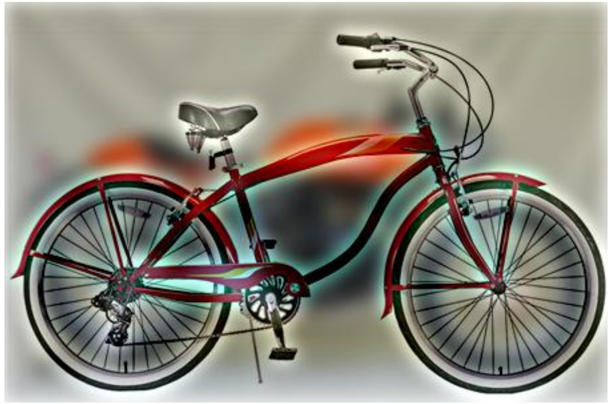
Cat + Dog

[insert your hybrid image here]



Motorcycle + Bicycle

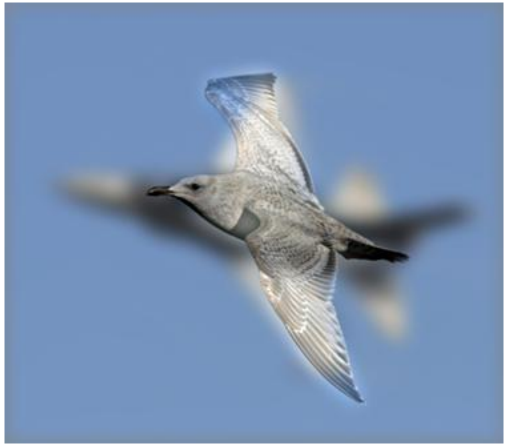
[insert your hybrid image here]



Part 2: Hybrid images with PyTorch

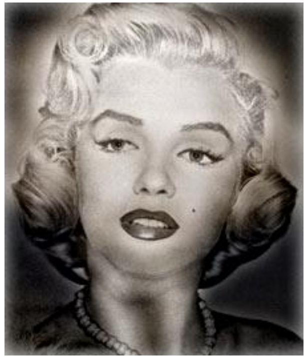
Plane + Bird

[insert your hybrid image here]



Einstein + Marilyn

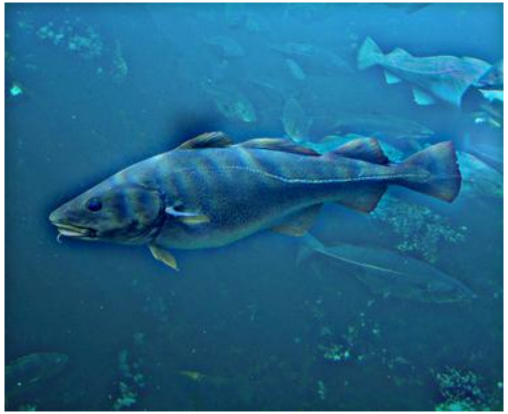
[insert your hybrid image here]



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 proj1.ipynb. Which method is faster?]

Part 1: 9.346 seconds
Part 2: 0.110 seconds

So part 2 with PyTorch is faster.

Part 3

[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?
Stride = 2, padding = 0?
Stride = 1, padding = 1?
Stride = 2, padding = 1?]

1-channel 3x3
1-channel 2x2
1-channel 5x5
1-channel 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
Stride = 2, padding = 0
Stride = 1, padding = 1
Stride = 2, padding = 1?]

Input: (3,361,410) Output: (1,359,408)
Input: (3,361,410) Output: (1,180,204)
Input: (3,361,410) Output: (1,361,410)
Input: (3,361,410) Output: (1,181,205)

Part 3

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

12

[Why do the output dimensions adhere to the equations given in the instructions handout?]

Because it can be regarded as a kxk square move on a hxw grid with stride s, and padding parameter p can be regarded part of the grid. So the new grid will be (h+2p)*(w+2p). And in every direction, the num of steps that kxk square can move is (h+2p-(k-s))/s and (w-k+2p)/s +1, which is equal to (h-k+2p)/s +1 and (w-k+2p)/s +1. And this is the output dimension.

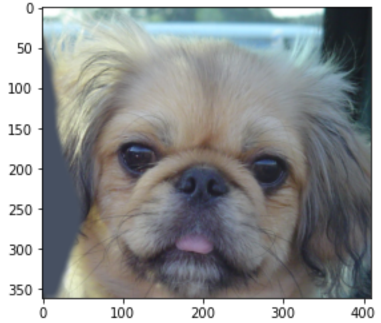
Part 3

[What is the intuition behind this equation?]

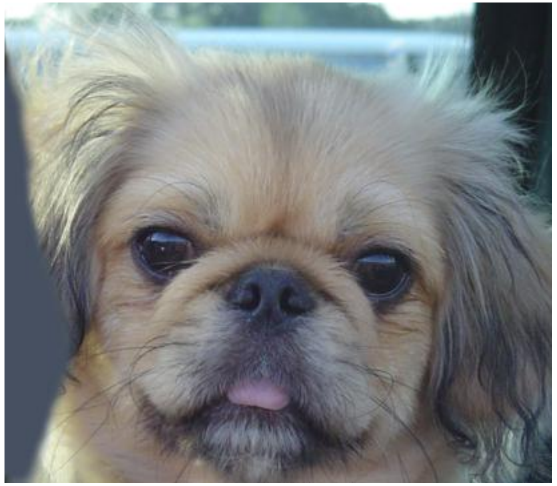
Because it can be regarded as a kxk square move on a hxw grid with stride s, and padding parameter p can be regarded part of the grid. So the new grid will be (h+2p)*(w+2p). And in every direction, the num of steps that kxk square can move is (h+2p-(k-s))/s and (w-k+2p)/s +1, which is equal to (h-k+2p)/s +1 and (w-k+2p)/s +1. And this is the output dimension.

Part 3

[insert visualization 0 here]

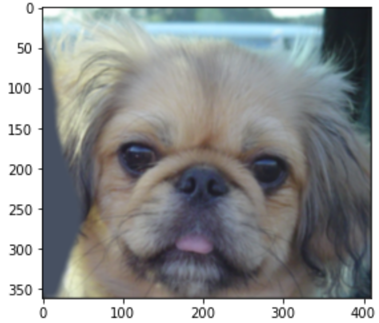


np.clip(feature_map+0,0,1)

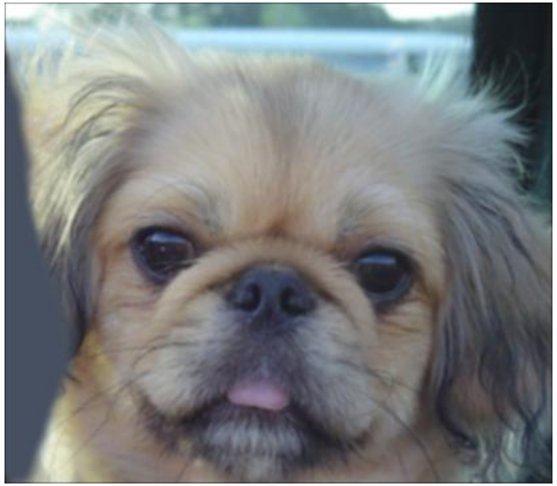


Feature_map

[insert visualization 1 here]



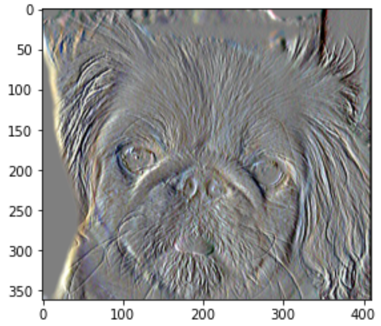
np.clip(feature_map+0,0,1)



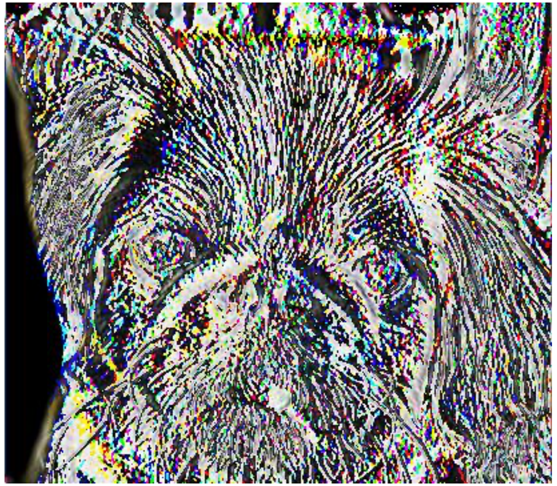
Feature_map

Part 3

[insert visualization 2 here]

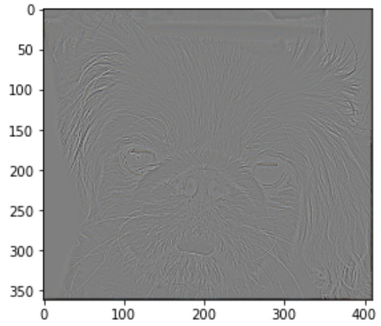


np.clip(feature_map+0.5,0,1)

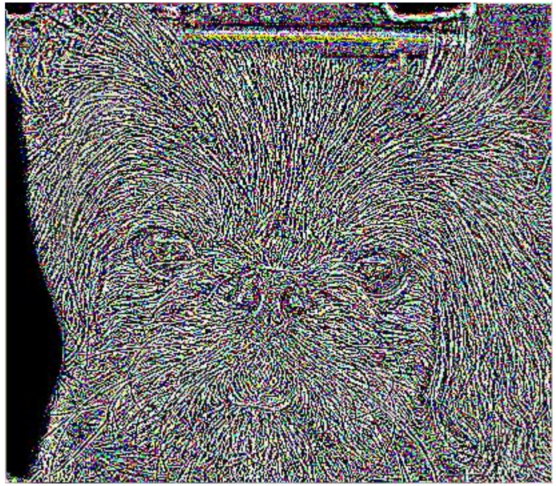


Feature_map

[insert visualization 3 here]



np.clip(feature_map+0.5,0,1)



Feature_map

Conclusion

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

Increase the cutoff frequency value will keep less high frequency information of the first image to make it more blur and more high frequency information of the second image. As a result, the hybrid image will look more like the second image which keeps its high frequency information.

Decrease the cutoff frequency value will result to the contrast effect.

Swap two images will change the real objects that low frequency information and high frequency information show, and that will make the hybrid image more like the other object.