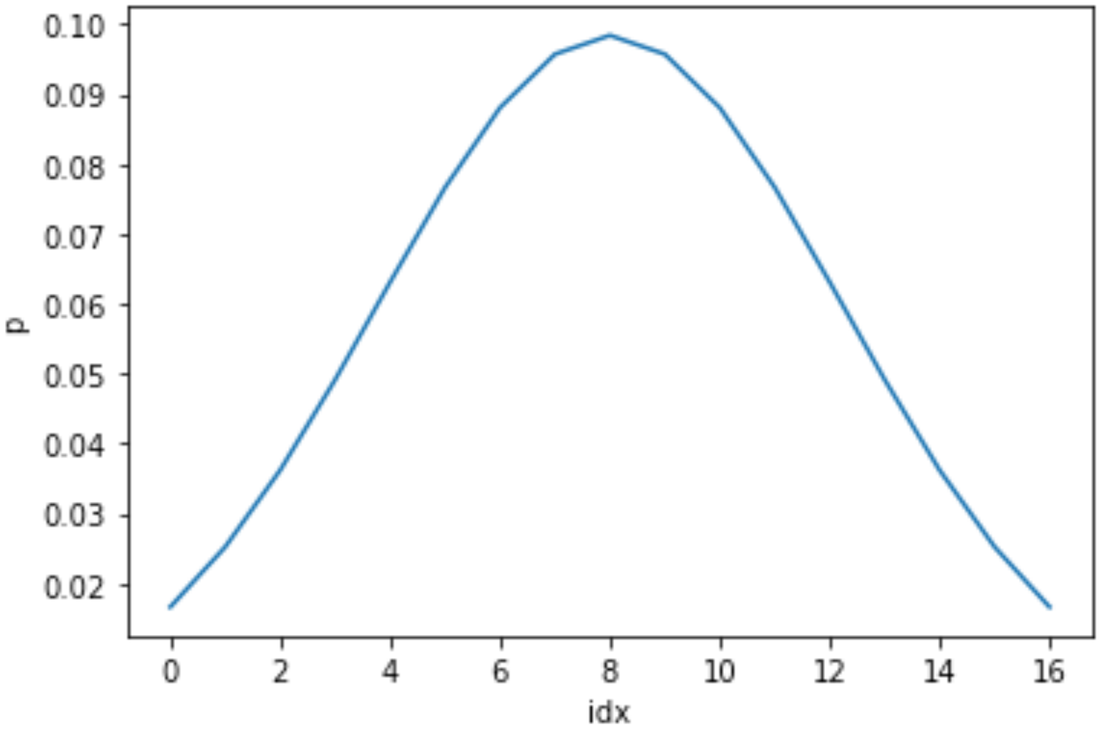
CS 6476 Project 1

Shen-Yi Cheng
scheng98@gatech.edu
scheng98
903514405

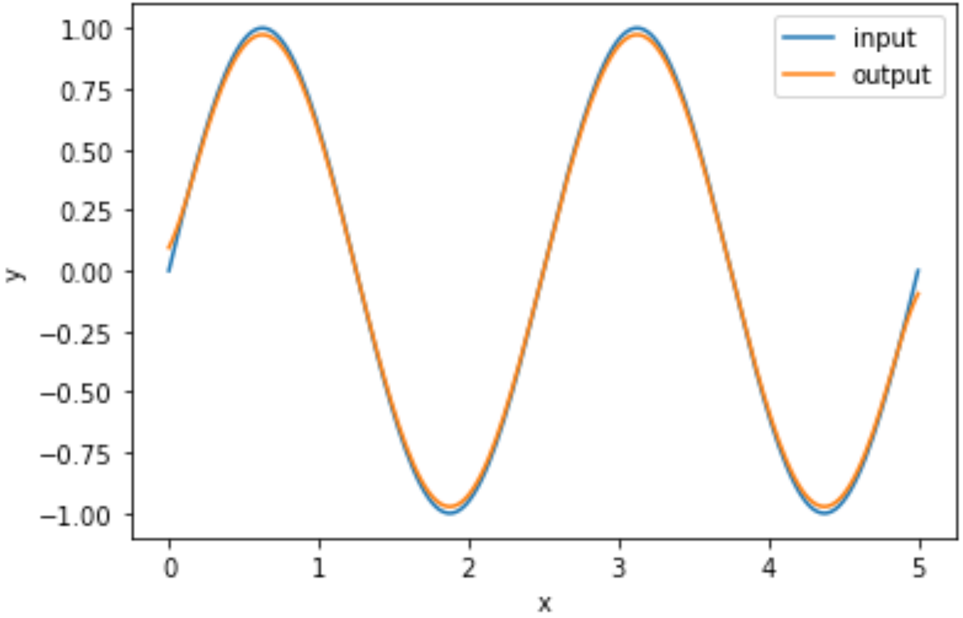
Part 1: 1D Filter

<insert visualization of the low-pass filter from proj1.ipynb here>



Part 1: 1D Filter

<insert visualization of filtered combined signal from proj1.ipynb here>



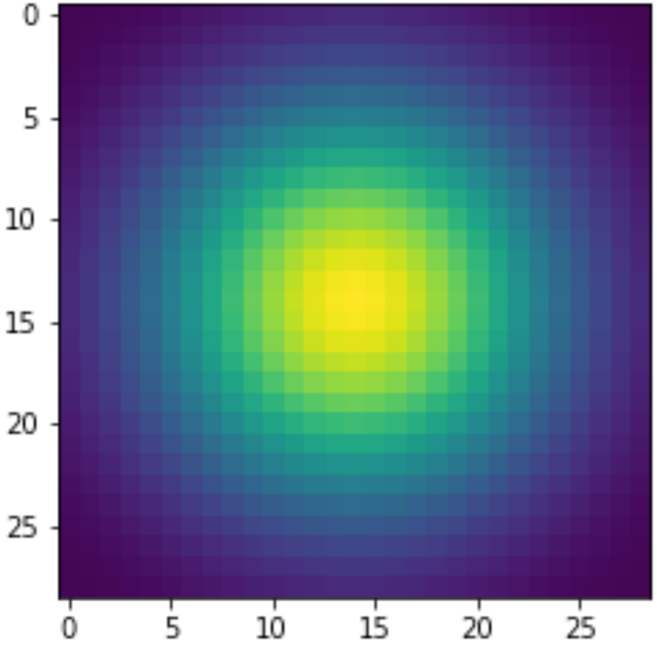
Describe your implementation in words and reflect on the checkpoint questions.

I create a 1D Gaussian low-pass filter to get the low frequency signal. And this filter also attenuate the high frequency while the low frequency is not affected.

Yes, the unit test passed.

Part 2: Image Filtering

<insert visualization of the 2D Gaussian kernel from proj1.ipynb here>



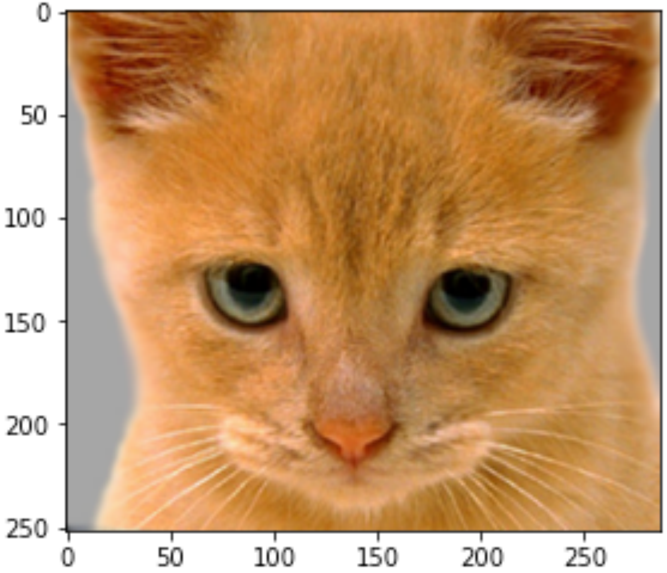
<Describe your implementation of my_imfilter() in words.>

I get the shape of image and filter first and use filter tensor to pad image tensor. Then use three for loops to create a new filtered image by calculating the dot product into tensor.

Part 2: Image filtering

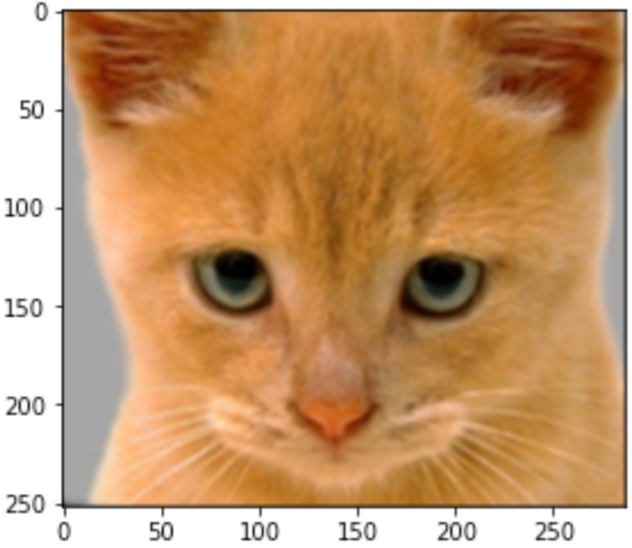
Identity filter

<insert the results from proj1_test_filtering.ipynb using 1b_cat.bmp with the identity filter here>



Small blur with a box filter

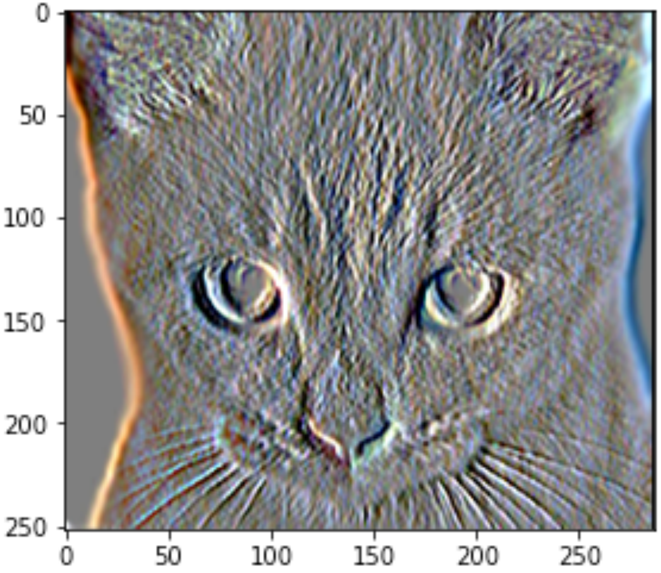
<insert the results from proj1_test_filtering.ipynb using 1b_cat.bmp with the box filter here>



Part 2: Image filtering

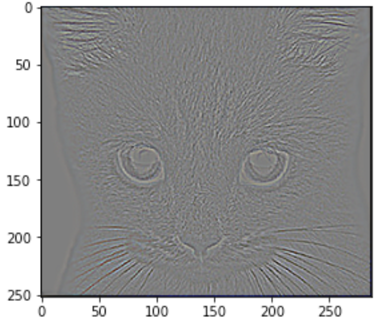
Sobel filter

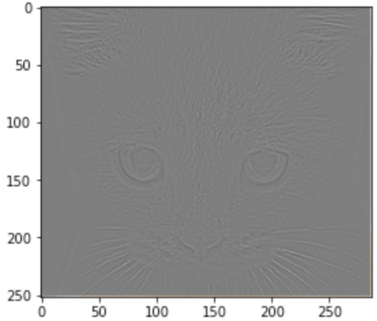
<insert the results from proj1_test_filtering.ipynb using 1b_cat.bmp with the Sobel filter here>



Discrete Laplacian filter

<insert the results from proj1_test_filtering.ipynb using 1b_cat.bmp with the discrete Laplacian filter here>





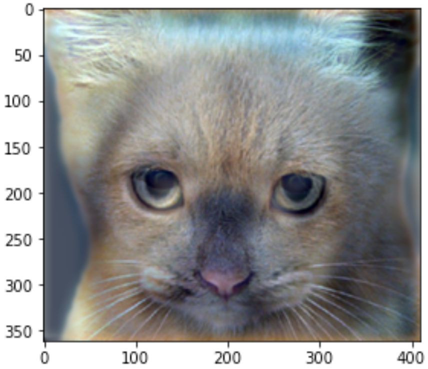
Part 2: Hybrid images manually using Pytorch

<Describe your implementation of create_hybrid_image() here.>

1. Dog x filter = low_freq_dog
2. Cat x filter = low_freq_cat
3. High_freq_cat = cat – low_freq_cat
4. (low_freq_dog + low_freq_dog) / 2 = tmp
5. hybrid = torch.clamp(tmp, min = 0, max = 1.0)

Cat + Dog

<insert your hybrid image here>

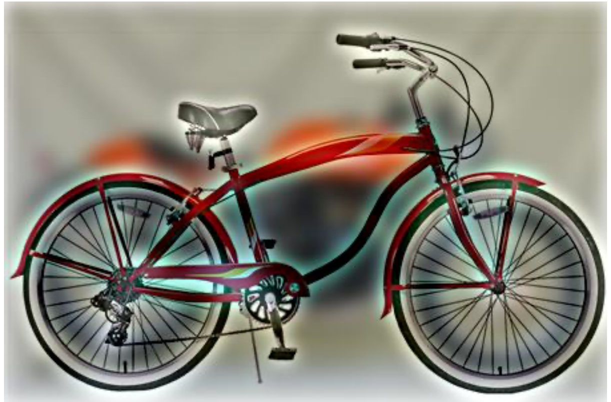


Cutoff frequency: <insert the value you used for this image pair> **7**

Part 2: Hybrid images manually using Pytorch

Motorcycle + Bicycle

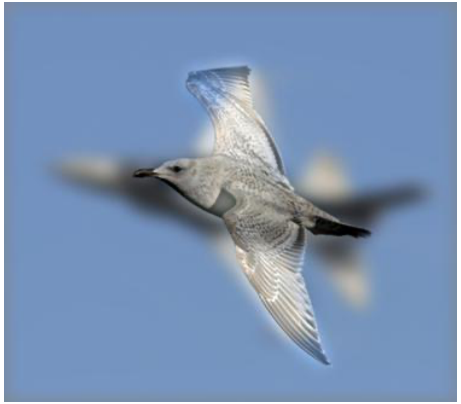
<insert your hybrid image here>



Cutoff frequency: <insert the value you used for this image pair> **7**

Plane + Bird

<insert your hybrid image here>

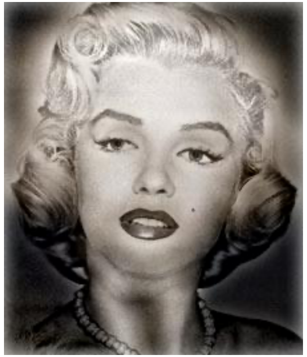


Cutoff frequency: <insert the value you used for this image pair> **7**

Part 2: Hybrid images manually using Pytorch

Einstein + Marilyn

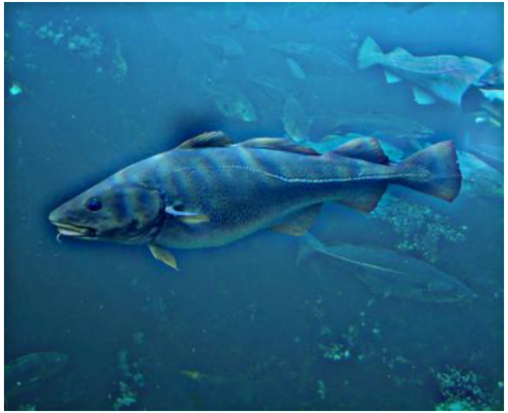
<insert your hybrid image here>



Cutoff frequency: <insert the value you used for this image pair> **7**

Submarine + Fish

<insert your hybrid image here>

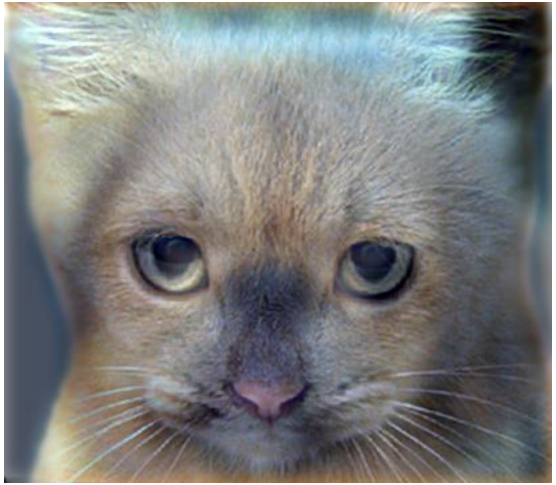


Cutoff frequency: <insert the value you used for this image pair> **7**

Part 3: Hybrid images with PyTorch operators

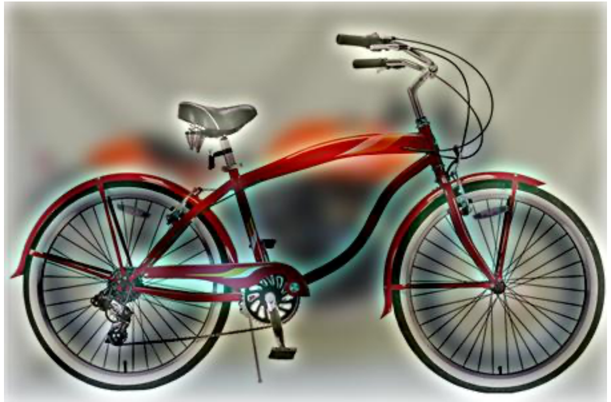
Cat + Dog

<insert your hybrid image here>



Motorcycle + Bicycle

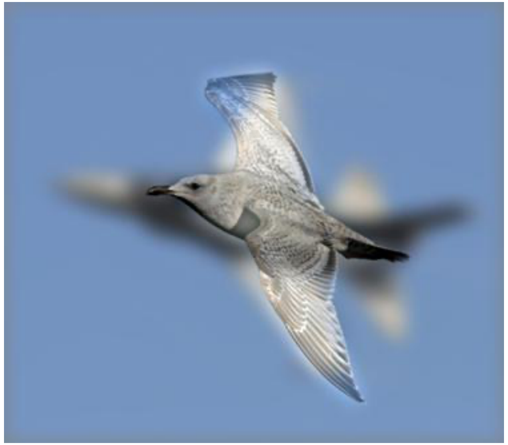
<insert your hybrid image here>



Part 3: Hybrid images with PyTorch operators

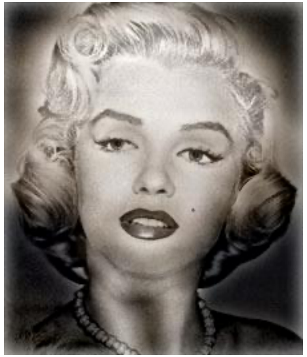
Plane + Bird

<insert your hybrid image here>



Einstein + Marilyn

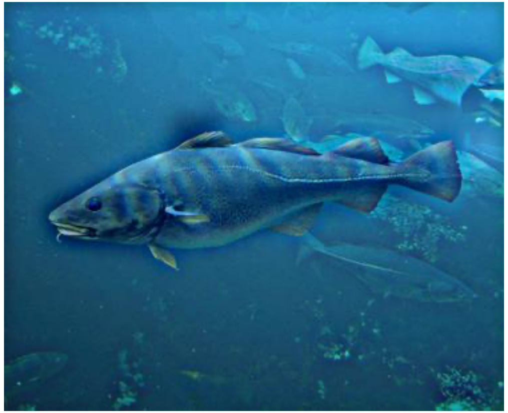
<insert your hybrid image here>



Part 3: Hybrid images with PyTorch operators

Submarine + Fish

<insert your hybrid image here>



Part 1 vs. Part 2

<Compare the run-times of Parts 2 and 3 here, as calculated in proj1.ipynb. What can you say about the two methods?>

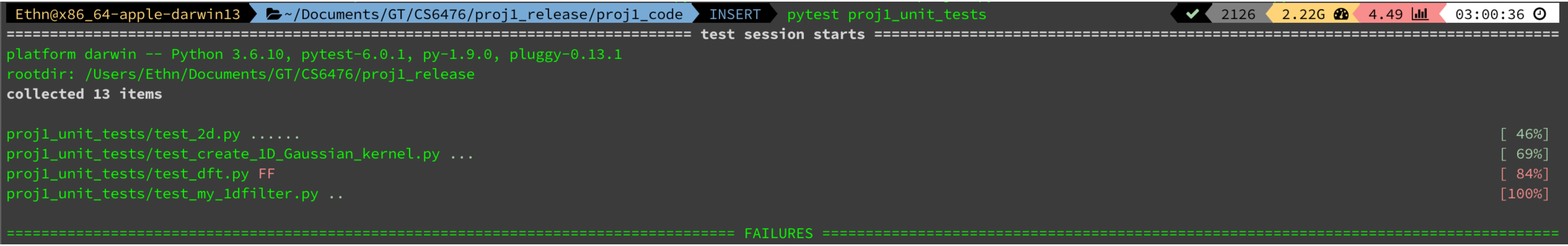
Part2: 45-55 sec

Part3: 0.18-0.25 sec

Torch.nn.functional.conv2d is much faster but when I implement myself, I can totally understand what inside the conv2d API

Tests

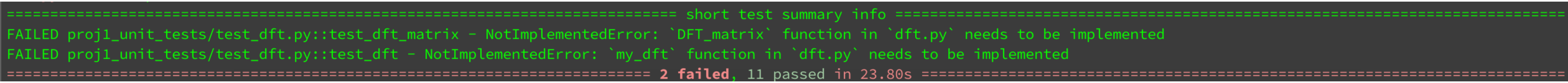
<Provide a screenshot of the results when you run `pytest tests` on your final code implementation (note: we will re-run these tests).>



```
Ethn@x86_64-apple-darwin13  ~/Documents/GT/CS6476/proj1_release/proj1_code  INSERT  pytest proj1_unit_tests          ✓ 2126  2.22G  4.49  03:00:36
================================================================ test session starts ================================================================
platform darwin -- Python 3.6.10, pytest-6.0.1, py-1.9.0, pluggy-0.13.1
rootdir: /Users/Ethn/Documents/GT/CS6476/proj1_release
collected 13 items

proj1_unit_tests/test_2d.py ......                                                                                                          [ 46%]
proj1_unit_tests/test_create_1D_Gaussian_kernel.py ...                                                                                      [ 69%]
proj1_unit_tests/test_dft.py FF                                                                                                             [ 84%]
proj1_unit_tests/test_my_1dfilter.py ..                                                                                                     [100%]

===================================================================== FAILURES ======================================================================
```



```
============================================================== short test summary info ==============================================================
FAILED proj1_unit_tests/test_dft.py::test_dft_matrix - NotImplementedError: `DFT_matrix` function in `dft.py` needs to be implemented
FAILED proj1_unit_tests/test_dft.py::test_dft - NotImplementedError: `my_dft` function in `dft.py` needs to be implemented
============================================================ 2 failed, 11 passed in 23.80s ============================================================
```

Conclusions

<Describe what you have learned in this project. Consider questions like how varying the cutoff standard deviation value or swapping images within a pair influences the resulting hybrid image. Feel free to include any challenges you ran into.>

Extra Credit

Image Filtering using DFT

<insert visualization of the DFT filtered 6a_dog.bmp from proj1.ipynb here>

Describe your implementation in words.

Add some cool hybrid images!