Q1.1.1

Gaussian filters are used to smooth and blur the images.

The other three filters can pick up edges in the images.

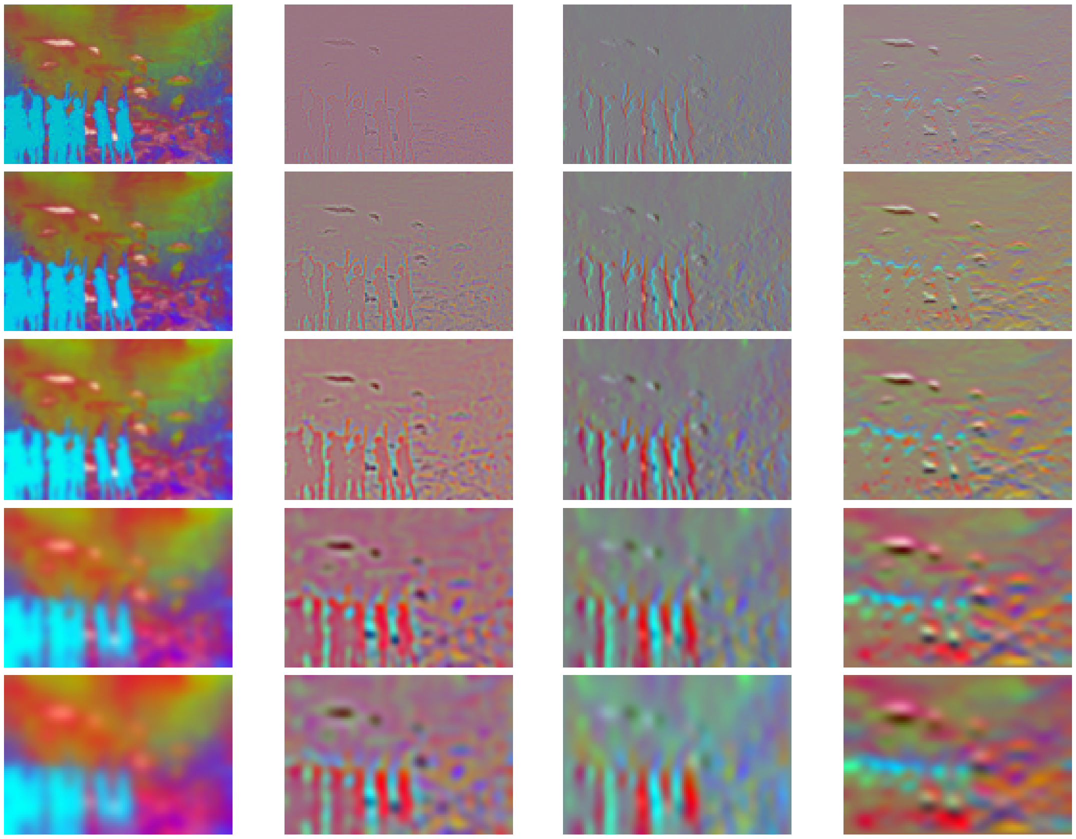
Laplacian filter can find edges in all directions, but it is sensitive to noises.
Gaussian of laplacian filters alleviate this problem by smoothing the images with gaussian filters.

Derivatives of gaussian in x direction pick up vertical edges.

Derivatives of gaussian in y direction pick up horizontal edges.

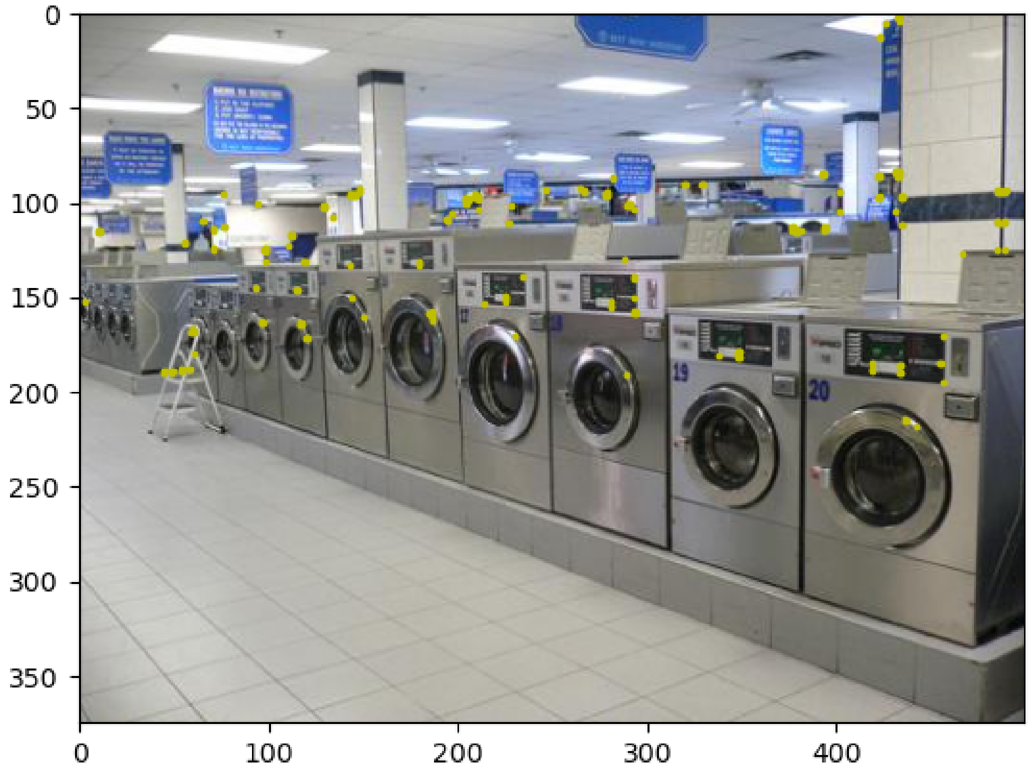
According to what we learned in class,
we can find highest filter response when the signal has the same characteristic scale as the filter.
Therefore, we need to search over different scales to find a scale that produces the peak filter response.

Q1.1.2

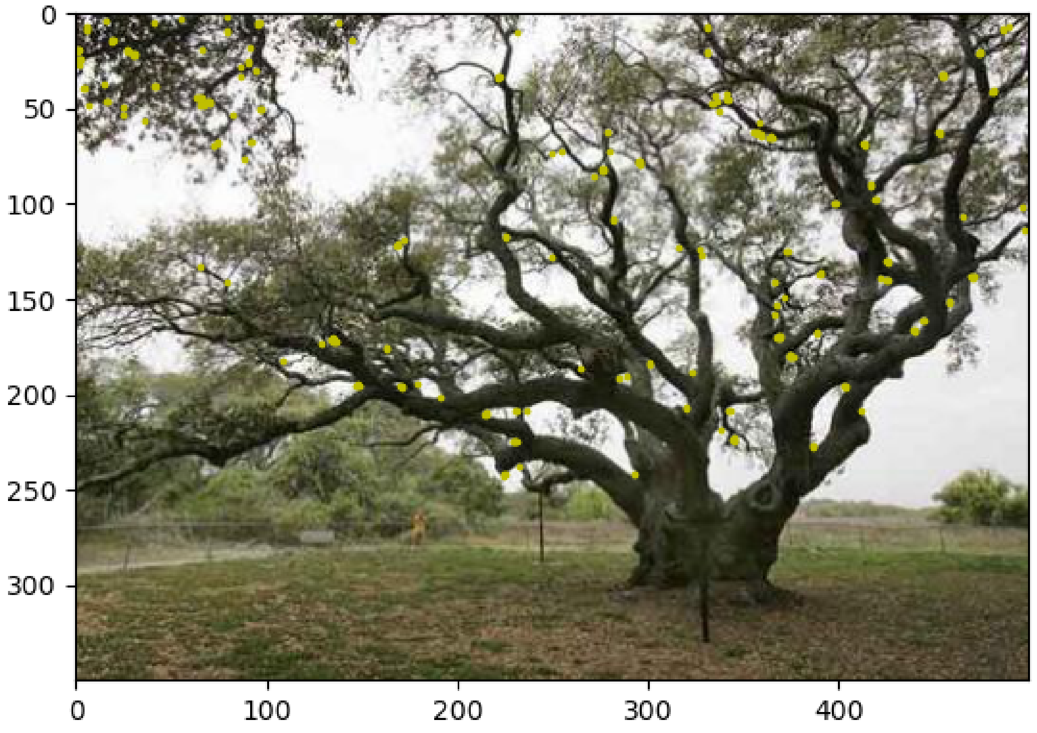


Q1.2.1

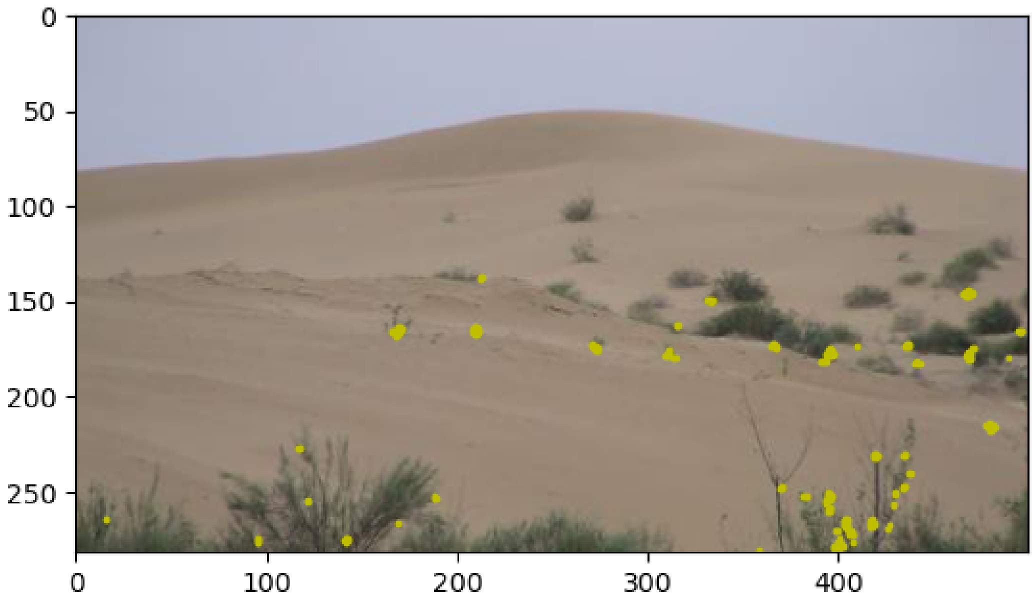
laundromat/sun_ahmgkuhrcsxibjkt.jpg



park/sun_aqeryumwccopqznm.jpg



desert/sun_bdvfsuhpokltwttm.jpg

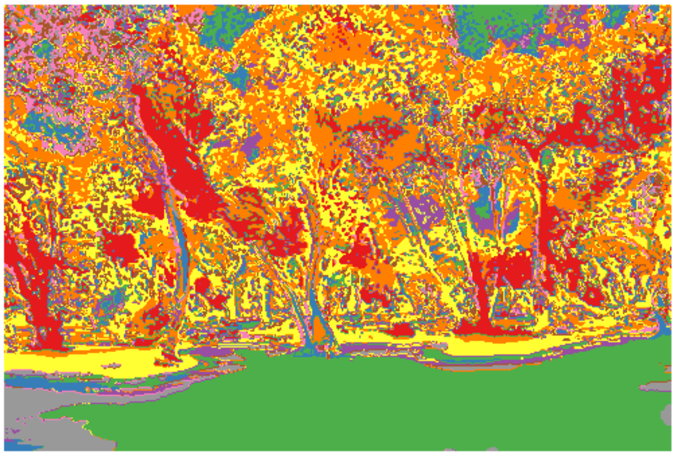


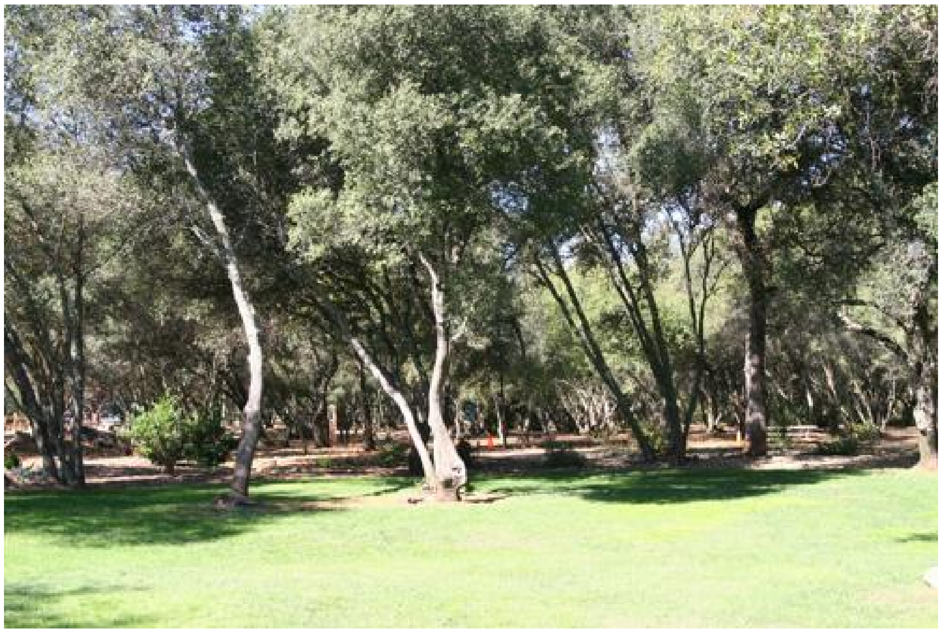
Q1.2.2

Please see details in the codes and dictionary.npy

Q1.3.1

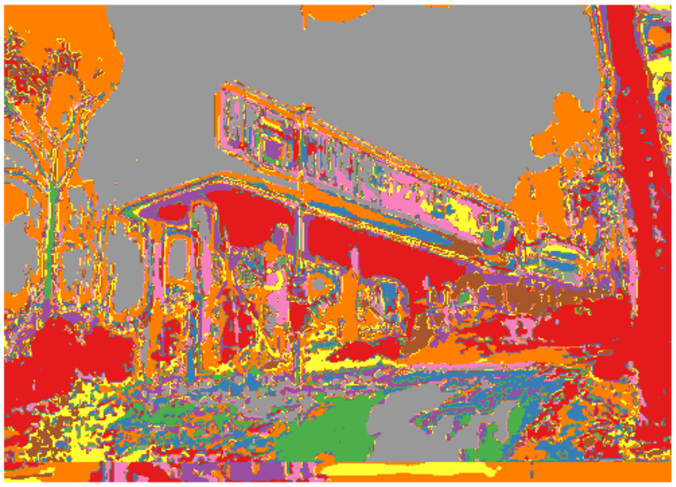
park/sun_bjhirlncvirefpno.jpg

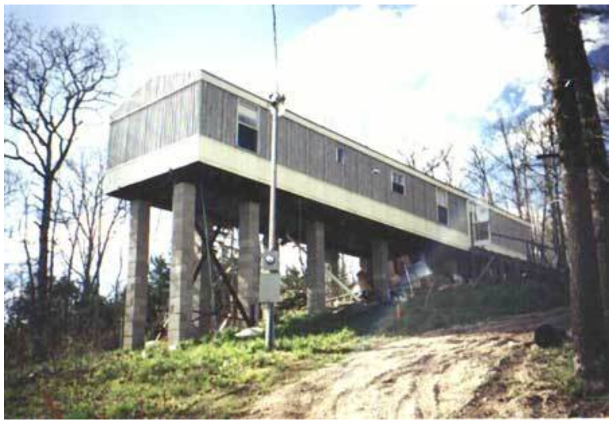




The boundaries make sense.
We can see that the bright parts of the grass are in one color (green) in the wordmap.
The areas of green leafs are either in orange or yellow, depending on the brightness of the leafs.
The dark areas, such as tree trunks, are red in the wordmap.

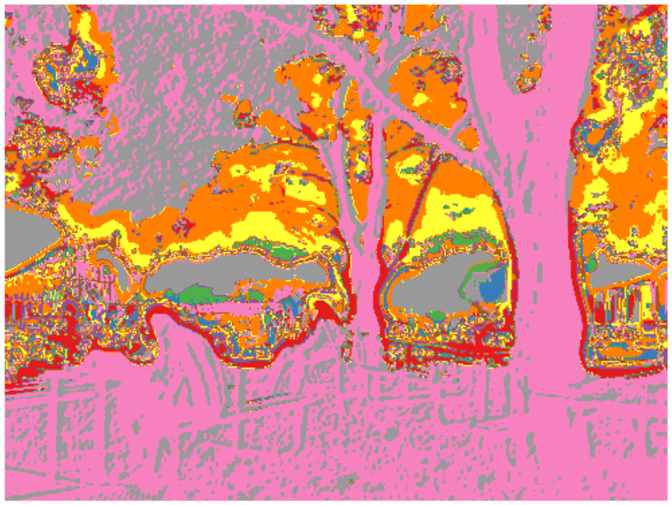
park/sun_aixwdvzmwvmrceyi.jpg

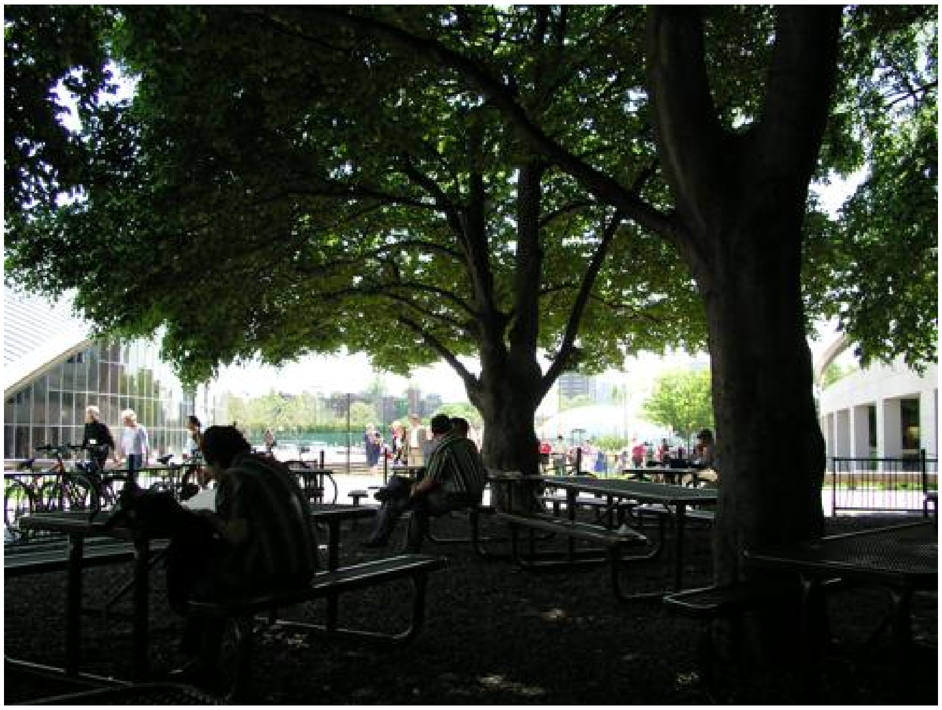




The boundaries make sense.
We can see that the bright areas in the sky and on the ground are in one color (gray).
The dark areas, such as tree trunks and the bottom side of the building, are in another color (red).

park/labelme_fjcqebsjrjqaxut.jpg

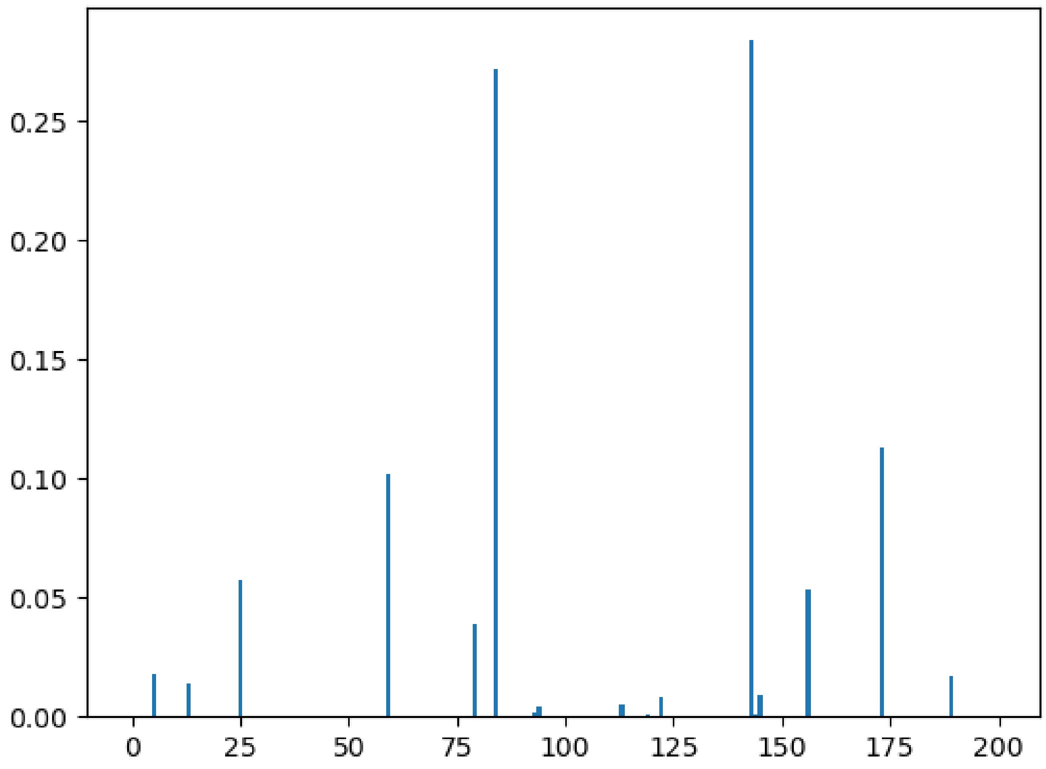




The boundaries make sense.
We can see that the bright background is in one color (gray).
The dark areas, such as tree trunks and the ground are in another color (pink).
The areas of green leafs are either in orange or yellow, depending on the brightness of the leafs.

Q2.1.1

histogram for aquarium/sun_aztvjgubyrgvirup.jpg



Q2.2.1

See details in code.

Q2.3.1

See details in code.

Q2.4.1

See details in code.

Q3.1.1

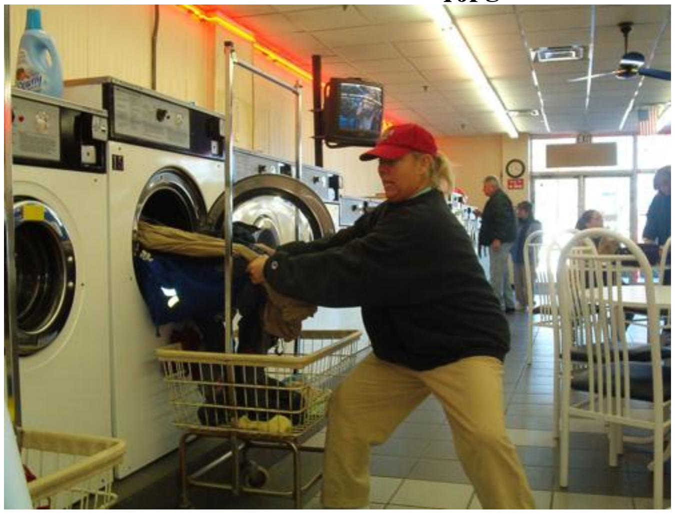
$$\begin{bmatrix} 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 15 & 0 & 0 & 0 & 2 & 1 & 0 \\ 0 & 0 & 18 & 3 & 1 & 0 & 0 & 3 \\ 2 & 2 & 1 & 14 & 0 & 2 & 0 & 5 \\ 2 & 0 & 0 & 0 & 10 & 1 & 0 & 0 \\ 0 & 2 & 1 & 0 & 9 & 11 & 1 & 0 \\ 0 & 3 & 2 & 1 & 2 & 1 & 12 & 0 \\ 0 & 2 & 3 & 2 & 3 & 0 & 1 & 8 \end{bmatrix}$$

The accuracy is 63.75%

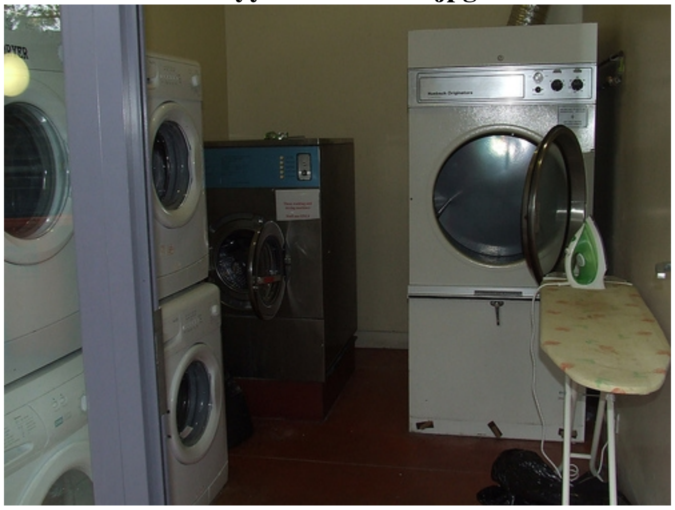
Q3.1.2

We can tell from the confusion matrix,
the model is not performing really well for class 5 (laundromat) and class 7 (windmill).
For class 5, here are some misclassified samples:

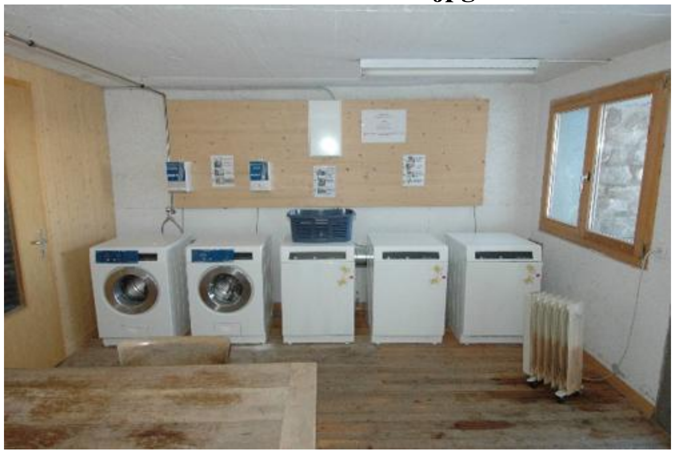
laundromat/sun_afrmdtnsnxzodzwq.jpg



laundromat/sun_aiyyxddvlfzfwdzc.jpg



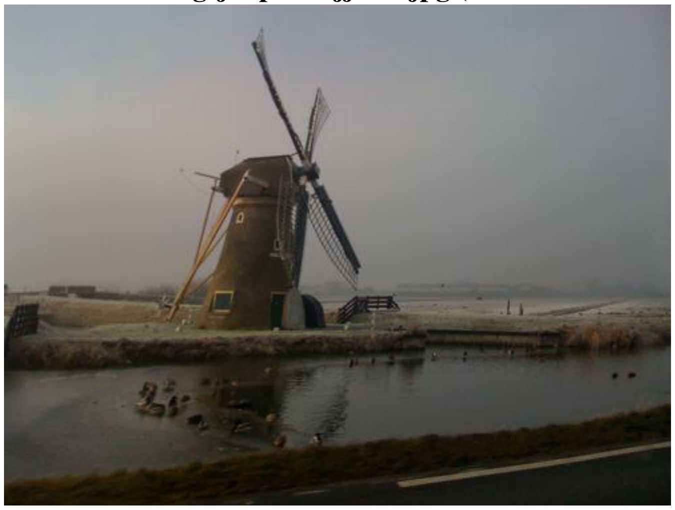
laundromat/sun_aiewbfesvvdzrhei.jpg



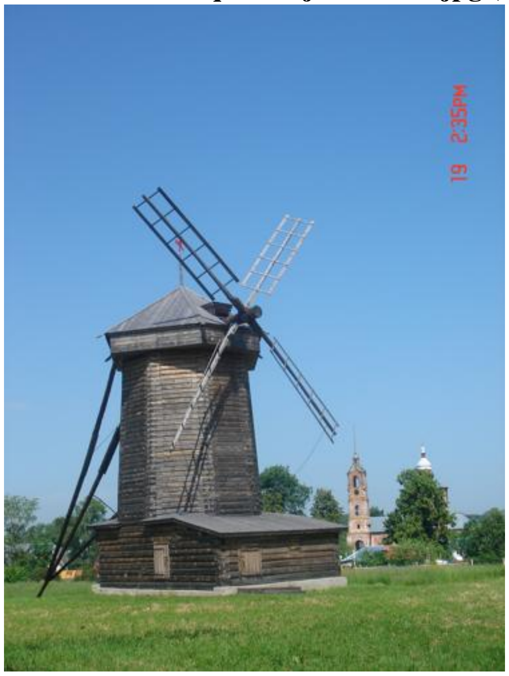
All three above images are misclassified as kitchen.
I think this is because laundry machines and stoves look quite similar, especially when they are all in white.

For class 7, here are some misclassified samples:

windmill/sun_bgsjdvplrxmjjnnd.jpg (misclassified as laundromat)



windmill/sun_beqradarsjammmwv.jpg (misclassified as park)



Unlike class 5, where the model usually misclassified the images to laundromat,
the mistakes for class 7 are more diverse.
I think this is because the colors change dramatically within this class.
We can see that the first image contains a lot of white and gray areas,
which can match to class laundromat.
For the second image,
the blue sky and green grass can make the model thinks that it is in the class of park.

Q3.1.3

I have reached an accuracy of 65.625% by setting $L = 4, k = 200, \alpha = 250$

To be honest, I didn't realize that we can have deterministic results from K-means method
by setting the random state parameter to an integer.

Therefore, I couldn't reproduce the exact dictionary.npy file
(due to random initialization from K-means),
which means if you don't use the same dictionary.npy file,
you might have higher or lower accuracy.

My thought process

I tried to tune the value of K and α first.

Initially, I let $k = 200, \alpha = 250, L = 3$
I received 63.75% of accuracy.

I tried multiple combinations of K $(150, 200, 250)$ and α $(200, 250, 300)$.
I expect the accuracy will increase with as the values of K and α increase.
However, in reality, the accuracy fluctuates between 60% to 64%.
I think the reason behind this is that K-means algorithm provides some randomness.

I have also tried horizontal random flip.
You can see the implementation in get_image_feature function in visual_words.py
(the code is commented out)
I expect the accuracy to increase since I did data augmentation.
However, the accuracy actually dropped to 60% with $K = 200, \alpha = 250, L = 3$
This is most likely due to K-means as well.

Finally, I tried to increase the value of L to 4 while keeping $K = 200, \alpha = 250$.
I expect the accuracy to increase,
because we would have more and smaller cells that can incorporate more specific spatial information.
I received 65.625% of accuracy.

Q3.1.4

I didn’t do this part.

Q4.1.1

On image aquarium/sun_aztvjgubyrgvirup.jpg,
the difference between results of the two feature extractors is: $2.5092011801675085e - 12$

Q4.2.1

The confusion matrix is:

$$\begin{bmatrix} 14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 17 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 22 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 26 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 23 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 21 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 19 \end{bmatrix}$$

The accuracy is 96.25%

Clearly, this is better than the classical bag of words.
This is because VGG is able to extract more complicated and non-linear features.
Also, the VGG has been pre-trained on ImageNet,
so the model might contain useful and transferable features for our task.