Q1.1.1 What properties do each of the filter functions pick up? Try to group the filters into broad categories (e.g. all the Gaussians). Why do we need multiple scales of filter response?

A Gaussian filter acts as a low-pass filter. It blurs the image and in doing so, eliminates high-frequency noise in the image.

The Laplace of Gaussian filter accentuates edges by producing a positive response on one side of the edge, and a negative response on the other side of the edge. All other values are zero.

The Gaussian filter with the derivative in the X direction accentuates vertical edges (discovered by sweeping horizontally over the image) in a similar fashion as the LoG.

The Gaussian filter with the derivative in the Y direction accentuates horizontal edges (discovered by sweeping vertically over the image) in a similar fashion as the LoG.

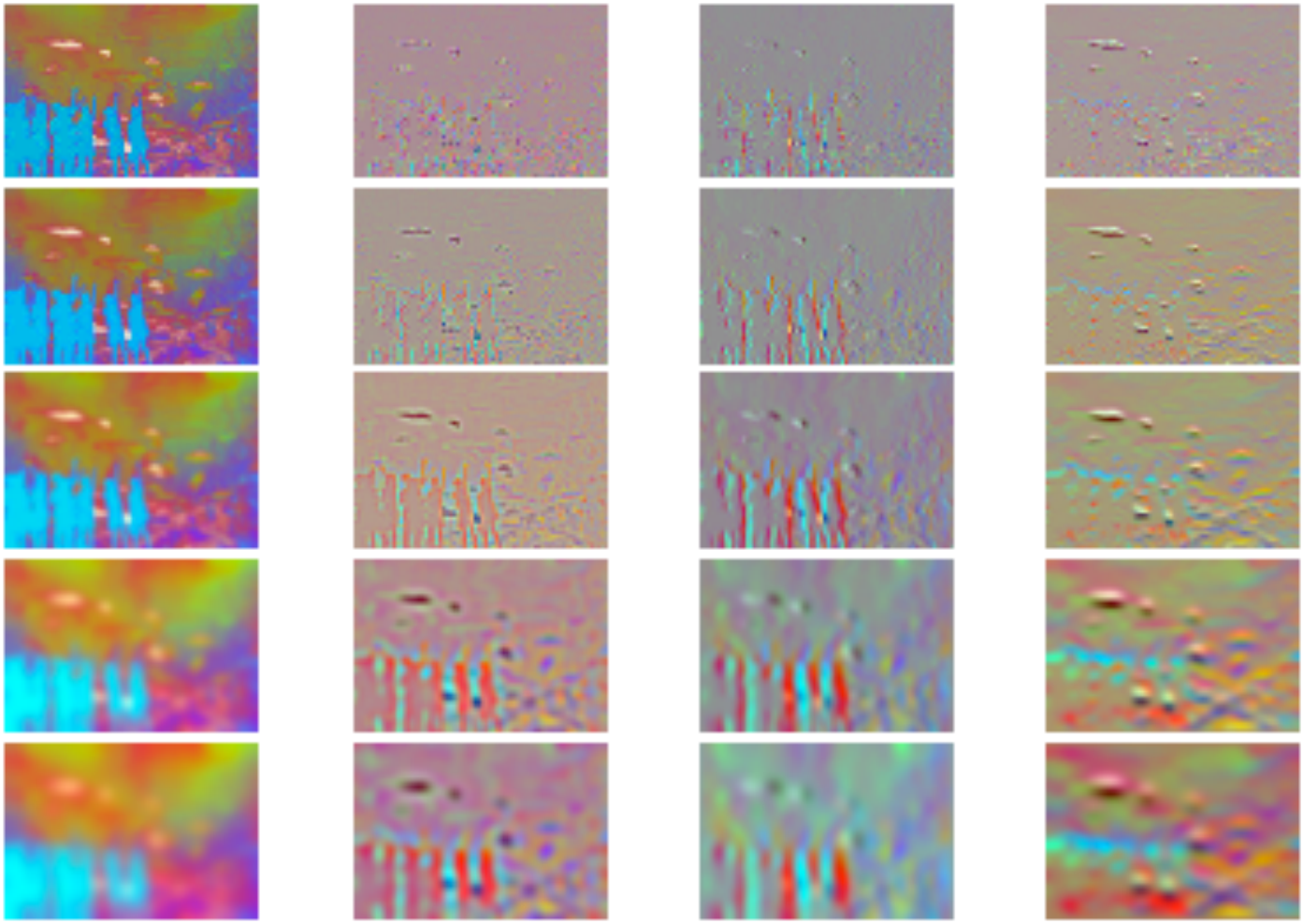
I'm not sure if this is totally right, but by the looks of it, the LoG seems like a combination of the Gaussian in the X and the Gaussian in the Y.

All of these filters need multiple scales so that they can pick up different sizes of features. As you go down the grid you'll see wider edges come into the response. Small scales pick up very fine edges, but larger scales pick up more gradual edges.

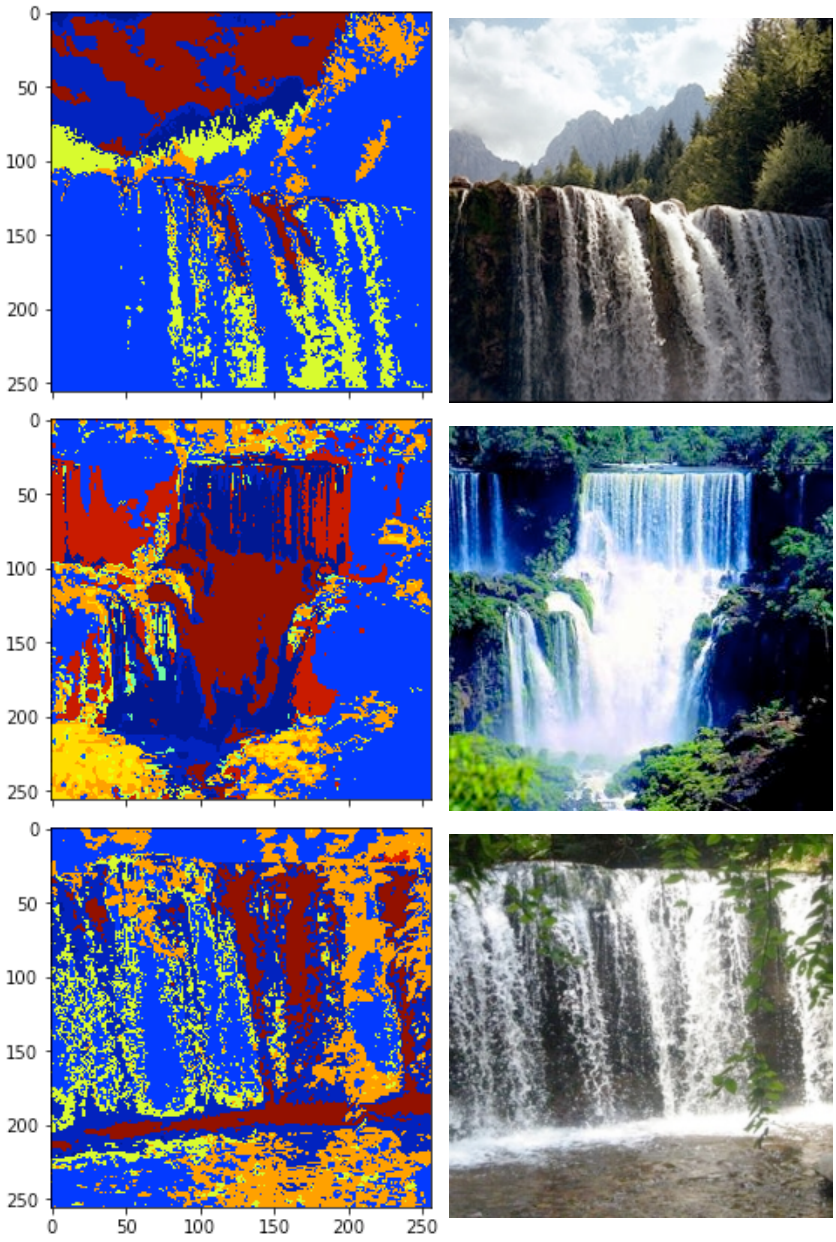
Q1.1.2 Visualization of filter responses on "aquarium/sun aztvjgubyrgvirup.jpg"

(scales increase down the rows)

Gaussian	Laplace of Gaussian	Gaussian in X	Gaussian in Y



Q1.3 Visualization of Wordmaps



The word boundaries here are fairly intuitive. They follow the contours of the image very well. We also start to see some patterns emerging. Green leaves surrounding the waterfalls are depicted as blue or orange. Also, well-lit whitewater shows up as red. The clouds in the first image also show up as red since that is fairly similar to the whitewater. The whitewater in the first image is in the shadows, so it shows up as a lime green color instead. This lime green is also visible in the third image.

Q2.5 Confusion Matrix and Accuracy Value

```
[[12.  0.  1.  0.  0.  0.  1.  0.]
 [ 1. 13.  0.  2.  0.  2.  0.  0.]
 [ 1.  1. 13.  5.  2.  0.  0.  3.]
 [ 1.  1.  4. 11.  0.  0.  3.  6.]
 [ 3.  1.  2.  0.  5.  1.  1.  0.]
 [ 0.  4.  1.  0.  1. 12.  3.  3.]
 [ 1.  5.  0.  0.  3.  3.  7.  2.]
 [ 1.  0.  3.  3.  1.  0.  2.  9.]]
0.5125
```

The rows indicate the ground truth labels and the columns indicate the computed labels.
The accuracy was 51.25%, which is way better than 12.5%!!!!

Q2.6 Hard Classes/Samples

6 highways were computed as windmills (conf[3][7]). This makes sense to me because the highway and windmill pictures have similar color schemes – greens, whites, blue skies, and brown dirt.

Also, 5 waterfalls were computed as parks (conf[6][1]). This makes sense as well since both waterfalls and parks depict untouched natural landscapes, with many similar features (grass, rocks, etc). There is even snow in a few park pictures that could be falsely thought to be whitewater.

Lastly, 5 deserts were computed to be highways (conf[2][3]). Both of these landscapes are fairly regular with not much change in landscape. They also share similar color schemes, especially in desert scenes that have more than just sand on the ground

```
[[12.  0.  1.  0.  0.  0.  1.  0.]
 [ 1. 13.  0.  2.  0.  2.  0.  0.]
 [ 1.  1. 13.  5.  2.  0.  0.  3.]
 [ 1.  1.  4. 11.  0.  0.  3.  6.]
 [ 3.  1.  2.  0.  5.  1.  1.  0.]
 [ 0.  4.  1.  0.  1. 12.  3.  3.]
 [ 1.  5.  0.  0.  3.  3.  7.  2.]
 [ 1.  0.  3.  3.  1.  0.  2.  9.]]
0.5125
```

Q3.2 Confusion Matrix and Accuracy for Deep Learning Implementation

```
[[12.  0.  0.  0.  1.  1.  0.  0.]
 [ 0. 17.  0.  0.  0.  0.  1.  0.]
 [ 0.  0. 24.  0.  0.  0.  1.  0.]
 [ 0.  0.  0. 25.  0.  0.  1.  0.]
 [ 0.  0.  0.  0. 12.  1.  0.  0.]
 [ 0.  0.  0.  0.  3. 21.  0.  0.]
 [ 0.  1.  0.  0.  0.  0. 20.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. 19.]]
0.9375
```

The rows indicate the ground truth labels and the columns indicate the computed values.
The accuracy was 93.75%, which is WAYYYYYYY better than 12.5%!!!!

My initial guess as to why the VGG implementation worked so well is because each feature in this implementation had 4096 dimensions while in the bag of words approach, each image only had K dimensions. But in spatial pyramid matching, each image feature is of dimension (K*(4^layer_num-1)/3), which in my case of K=200, is actually 4200. So my first intuition was incorrect.

I think the VGG implementation is so much better because it has been researched and optimized to look for image features, and because it goes through 30+ steps to bring all images down to a standard feature size. In contrast, our bag of words/SPM approach was only 3 layers deep. It stands to reason that as you add more layers to the network (within reason) the network will become a much better predictor of image scenes. Another reason why the VGG implementation is more effective is that it is more standardized. By standardizing the input size of the images into the VGG network to 224x224, the network has similar behavior over all images. In our bag of words/SPM implementation, we use a variety of image sizes, which most likely produces noisy results.