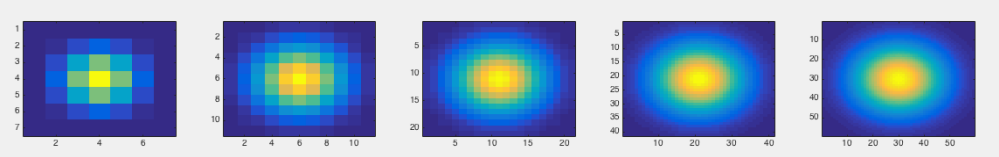
1: The First Problem

What properties do each of the filter functions (See Figure 3) pick up? You should group the filters into broad categories (i.e., all the Gaussians). Answer in your write-up.

(a) Solution:

The filter bank consists of four categories and each of them has 5 different scales.

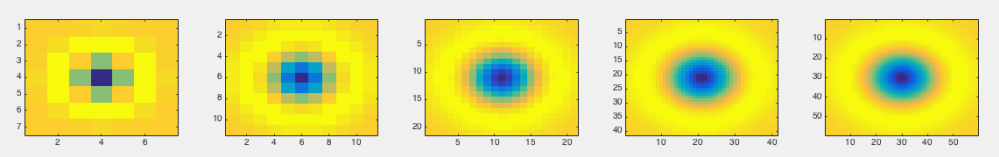
Figure 1: Gaussian



From left to right, the scales are $1, 2, 4, 8, 11.31$ and the size are $7, 11, 21, 41, 59$

The Gaussian filters smooth the images at different scales and remove high frequency noisy.

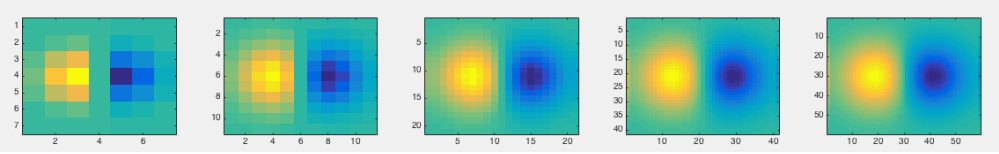
Figure 2: Laplacian of Gaussian



From left to right, the scales are $1, 2, 4, 8, 11.31$ and the size are $7, 11, 21, 41, 59$

The LoG filters give us the high responses at corners.

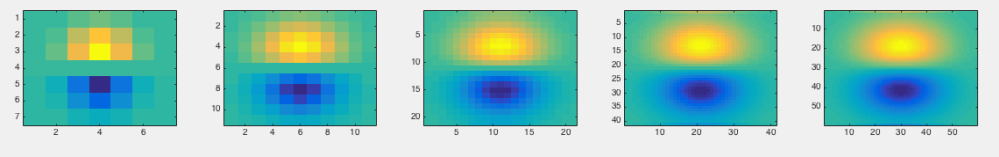
Figure 3: Derivatives of Gaussian along X-axis



From left to right, the scales are $1, 2, 4, 8, 11.31$ and the size are $7, 11, 21, 41, 59$

Derivatives of Gaussian along X-axis can give us high responses at edges along y-axis.

Figure 4: Derivatives of Gaussian along Y-axis



From left to right, the scales are $1, 2, 4, 8, 11.31$ and the size are $7, 11, 21, 41, 59$

Derivatives of Gaussian along Y-axis can give us high responses at edges along x-axis.

2: The second Problem

2.5 Quantitative Evaluation

(a) Solution:

Basic results here:

Figure 5: The final result of confusion matrix

```
rate =

   55.6250

>> result

result =

    15     1     2     0     0     0     0     2
     4    13     1     0     0     1     1     0
     3     3    12     1     0     1     0     0
     0     2     0     6     3     2     6     1
     0     3     1     1    13     0     2     0
     1     0     0     3     3     6     6     1
     0     1     0     4     4     0     8     3
     2     0     0     1     0     0     1    16
```

3: The Third Problem

Find out the failed cases :

(a)

Solution:

There are several reasons leading to failed cases. The first one is the input image has complex structures or in other words, the input image has the structure that more similar to other cases, like figure 6. The reflection of ceiling and complex structures more look like leafs in forest, which leads to the wrong labeling.

The second reason is that the the major color in input images is more similar to other cases, like figure 7. The main color of figure 7 is green, which leads it to the wrong cases.

The last reason is that some images are ambiguous, which makes them hard to classify, even for humans. Like figure 8, the correct label shows it is a Landscape, but it indeed also looks like a Desert. Thus, it is hard to correctly label such cases. Another case shows below, which should be landscape, but classified as desert.

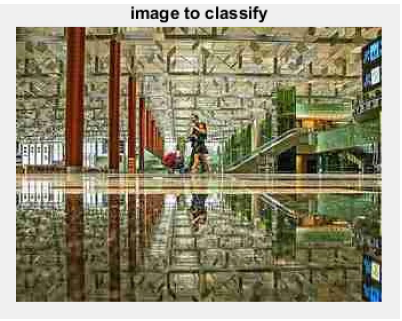
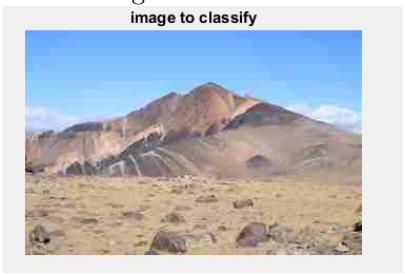


Figure 6: Forest

Figure 7: Forest

Figure 8: Landscape

Figure 9: Desert



4: The Fourth Problem

Improving Performance

(a)

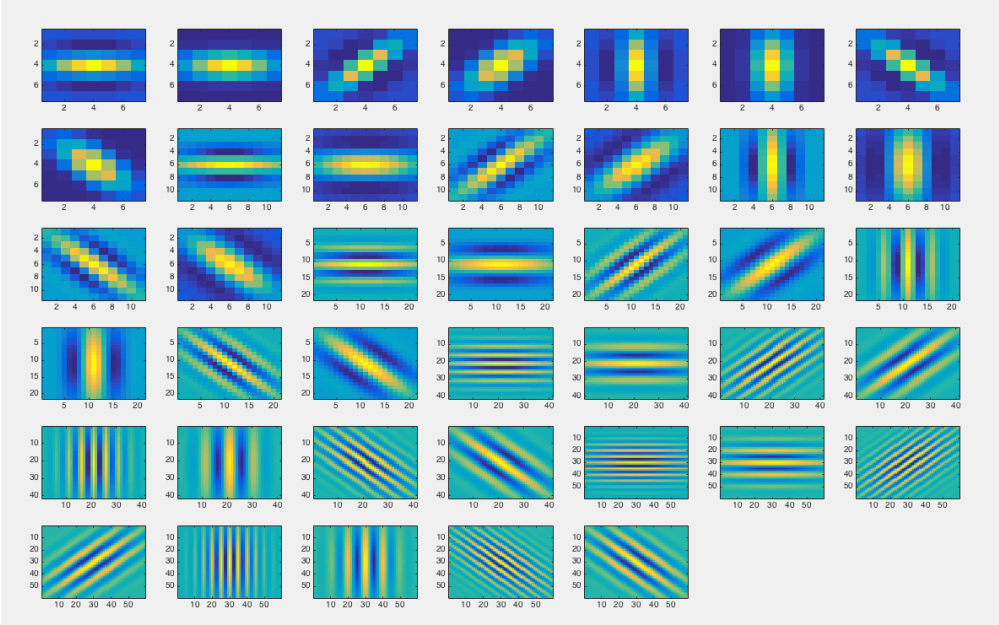
Solution:

After several different methods, the success rate on test set has achieved 77% $\sim$ 80% based on the Leung Malik (LM) Filter Bank and a simple Neural Network approach.

Before that, I also tried other approaches. At first, I applied the Gabor filter (*See /custom/createGarborFilter.m*). The filter bank has total 40 filters based on 5 different scales, 4 different rotations and 2 different frequency. See figure below:10

Garbor Filter

Figure 10: the Garbor filter bank



The reason that I choose Gabor filter is that it has much more variances besides scale and rotations. In this case, I double the size of filter bank to include two different frequencies. The configurations are shown here: $sigma = 1, 2, 4, 8, 11.31, theta = 0, \frac{1}{4}pi, \frac{1}{2}pi, \frac{3}{4}pi$ and $lamba = 5, 10$

Using Garbor filter indeed has better performance than the provided multi-scale filter bank, which I achieved approximate 65% success rate.

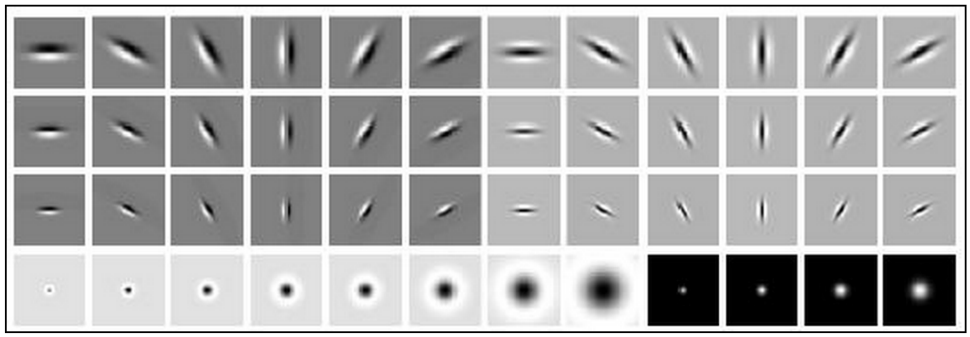
*Leung Malik (LM) filter**

*Refer: the figure and Matlab code of LM filter comes from
https://www.robots.ox.ac.uk/ vgg/research/texclass/filters.html

Besides Garbor filter, I also tried another filter called Leung Malik (LM) filter, which is a multi-scale, multi-orientation filter bank with 48 filters. There has first and second derivatives of Gaussians at 6 orientations and 3 scales, 8 Laplacian of Gaussian (LOG) filters and 4 Gaussians. See figure 11

Using LM filter back, the final accuracy is approximate 67.5%

Figure 11: The Leung Malik (LM) Filter Bank*



Cosine Similarity

Then, in order to continuously improve the final result, I used a common strategy in Text mining called cosine similarity. Cosine similarity gives the similarity between two documents. Thus, I also used cosine similarity here to define the similarity between two images. And we have equation below:

$$similarity = cos(\theta) = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{1}$$

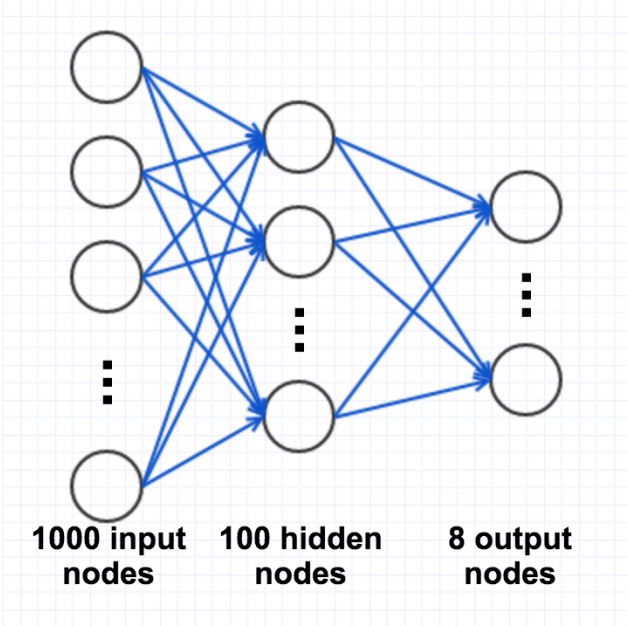
However, after testing Cosine Similarity approaches on LM filter bank, the accuracy still remains about 67.5%. I believe the main reason is that there are some words in the histogram misleading the classification results, which means some words are not distinguish enough or they show up in both two or more classes. For example, the word "grass" may both show up in class "forest" and class "football stadium". Even though, "grass" may be a main feature in this image, but it is not a distinguish feature to classify.

Neural Network

Refer: the code of Neural Network is based on my implementation of homework in course: "Machine Learning" by Andrew Ng on Coursera.

Finally, in order to find the most appropriate way to define the similarity of two images based on their bag of words histograms, I designed a quite simple neural network with only one hidden layer to select the most useful features. Thus, using a hidden layer, the neural network selects the most distinguish 100 features to classify these image. See figure below 12

Figure 12: A simple Neural Network with one hidden layer



Due to the limits of training set, I have to minimize the numbers of input parameters, thus I used only two layers in Spatial Pyramid Matching, which reduces the number of inputs to 1000. The input of neural network is the histogram using bags of words approach and the output is the label of classification. I trained the neural network using our training set. After testing with different parameters, even though the cost decreases in a slow manner after 1000 iterations in back-propagation, I achieved the highest success rate about $77\% \sim 80\%$ (depends on the training results).

Results:

Figure 13: Final success rate based on the ML filter bank and neural network

```
rate =

   78.1250

>> result

result =

    15     1     3     0     0     0     1     0
     4    14     2     0     0     0     0     0
     0     2    16     0     0     0     2     0
     4     2     0    12     0     0     2     0
     0     0     0     0    16     0     4     0
     0     2     0     1     0    17     0     0
     1     1     0     1     2     0    15     0
     0     0     0     0     0     0     0    20
```

Figure 14: Final success rate based on the ML filter bank and neural network

```
result =

    15     1     2     0     0     1     0     1
     4    14     1     0     1     0     0     0
     0     2    17     0     0     0     1     0
     4     1     0    13     1     0     1     0
     0     0     0     0    17     0     3     0
     0     2     0     1     0    17     0     0
     1     1     0     1     1     0    16     0
     0     0     0     0     0     0     0    20

>> rate

rate =

   80.6250
```