HW1 Write-Up

16-720: Computer Vision

Spring 2016

Cole Gulino

January 27, 2016

1. Representing the World with Visual Words

Q1.0: 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.

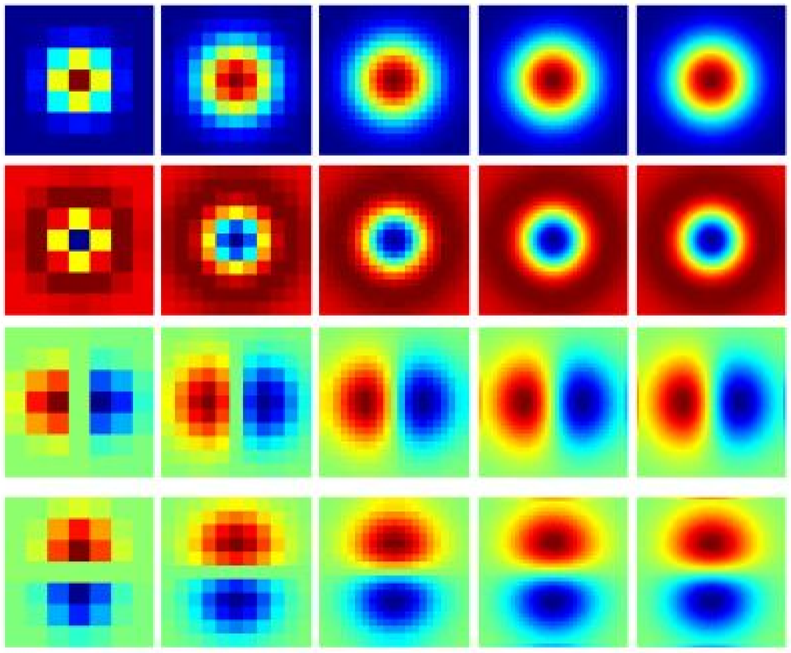


Figure 1) Filter Bank

The first group of filters are a set of single Gaussians. This group corresponds to the first and second row. These are arranged from left to right. The leftmost is a smaller filter that corresponds to a finer selection that filters out less of the image and the rightmost corresponds to a coarser/larger selection that filters out more of the image. So the ones on the right will pick up larger features, while the ones on the left will pick up smaller details. The first two rows will filter out points. So the first row will filter out the dark portions of the image. The second row will filter out the lighter portions of the image. This is apparent by the single Gaussian which forms a circle in a 2-D space. This will wrap around circular shapes.

The second group are a set of dual filters that filter for lines. So the filters in row three filter for vertical lines. The filters in row four filter for horizontal lines. This is apparent from the transition from positive to negative values of the shapes. These will correspond to moving from high intensities to low intensities which will be picked up by the filter.

Q1.1: Extracting Filter Responses

The filter response is a M x N x 3F matrix, which I have represented as a M*N x 3F matrix. Each row in the matrix represents one of the pixels, and the 3F rows represents a filter response to that image. Figure 2 shows the matrix structure.

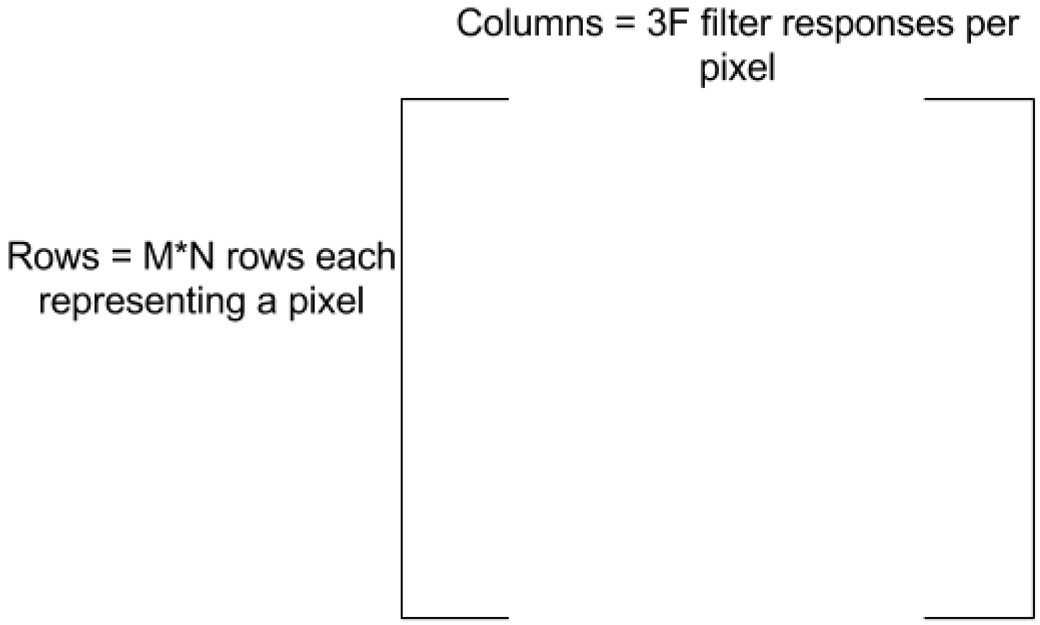

Columns = 3F filter responses per pixel
Rows = M*N rows each representing a pixel

Figure 2) Matrix Representation of Filter Responses

I looped through the filter bank which was provided. This filter bank contains 20 filters as shown in Q1.0.

To convolve the image with the filter, I use the imfilter() MATLAB command. I used the ‘conv’ command in order to choose convolution over the default value of correlation. I also padded the image using the “mirror” method which I specified as the ‘symmetric’ command. The final parameter I issued was ‘same’ so that the filtered image would be the same size as the original image.

Figure 3 shows the before and after of the image through one of the filters in the filter bank provided.

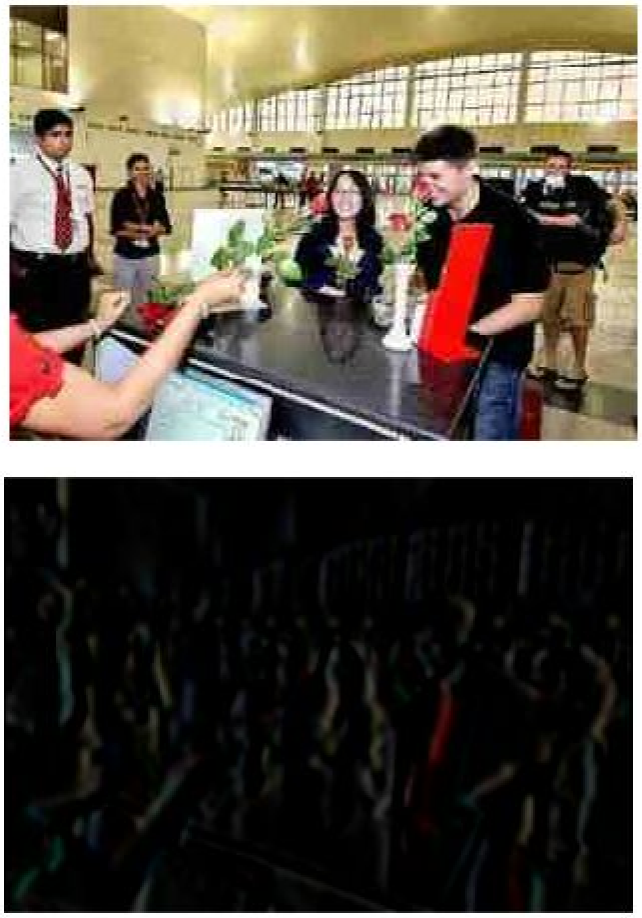


Figure 3) Before and After Use of a Filter from the Filter Bank using imfilter

Q1.2: Creating Visual Words

In this section, we were tasked to create a dictionary of the visual words using K-means clustering grabbing random pixels as features. The original space can be upwards of 70,000 pixels, so we can grab a few random pixels from each image and cluster them in order to create visual words that represent parts of the image.

We are also tasked with providing whatever filter bank we used. I used the filter provided to us and called createFilterBank() in order to return that as well.

We also need to provide the dictionary of visual words. The matrix structure for the visual word dictionary is shown in Figure 4.

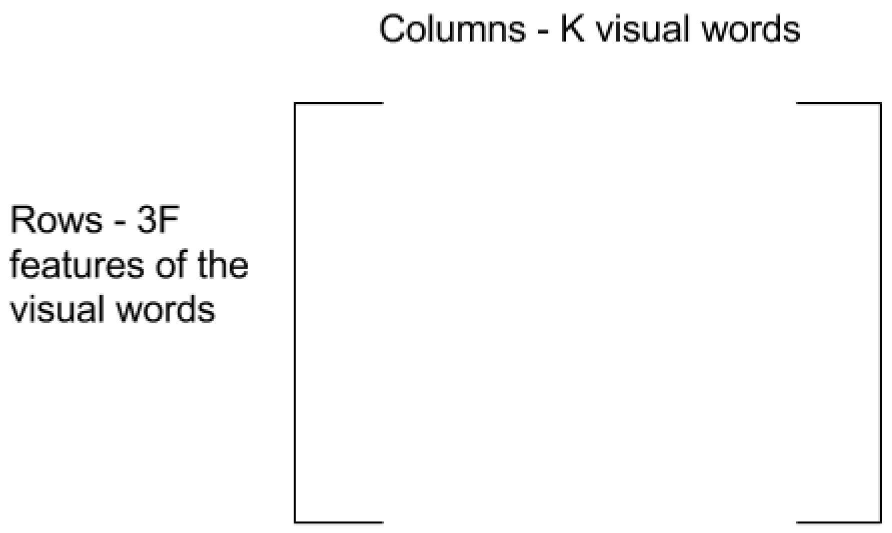


Figure 4) Matrix Structure of Dictionary of Visual Words

Table 1 below shows the parameters that I used for k-means clustering.

K	**Alpha**
250	300

Table 1) Parameters used for K-means Clustering

The K value controls the number of clusters used for k-means clustering and also controls how many visual words are in the dictionary. Higher values of K can lead to overfitting of the training set, while lower values of K can fail to provide accurate classification by being too general. I found that 250 was a good value for K.

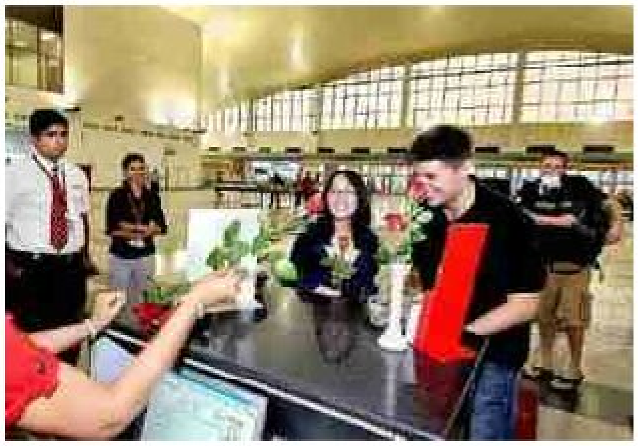
Alpha controls how many pixels from each image are sampled randomly. The higher alpha in theory the better the performance, but this comes with a computation speed trade off. Higher values of alpha take longer to compute.

Q1.3: Compute Visual Words

This is a function that uses the filter response of an image in order to find the closest match at each pixel of the filter response to one of the K visual words. Each filter response for the pixels is 3F matching the visual word 3F features. The index corresponding to the visual word that each feature closely matches is placed at each pixel creating a word map. The word map is a

WxH matrix the size of the original image that contains values corresponding to the indices of the dictionary 1-K.

Figure 5 shows an example of the visualized wordmap before and after.



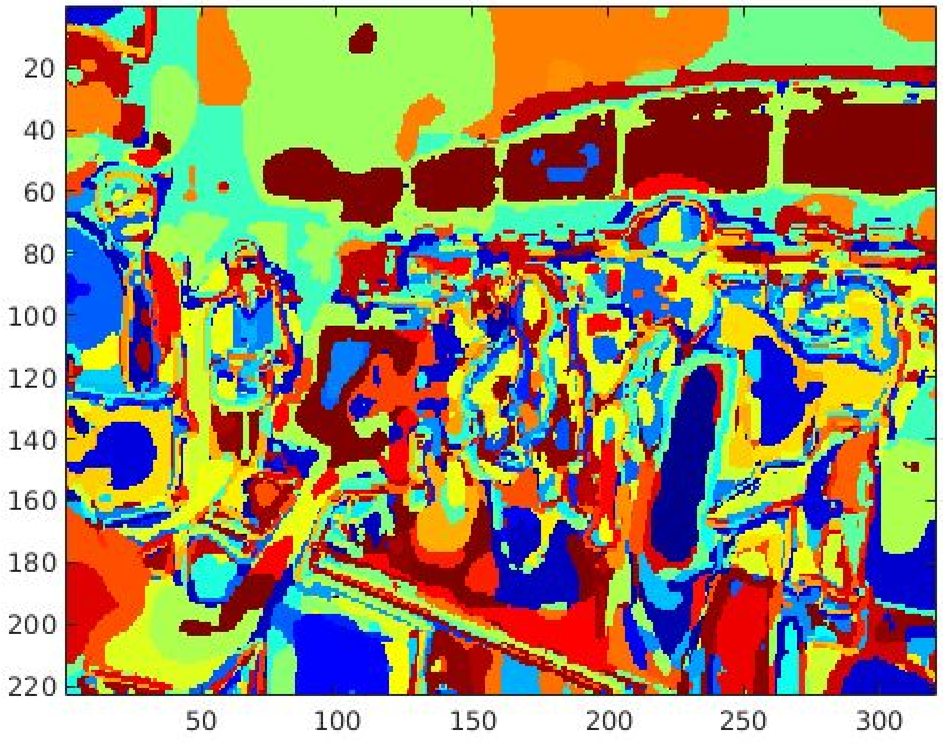


Figure 5) Before and After Generating the Visual Words

I used the matlab function pdist2 in order to get the smallest euclidian distance of the columns of the filter response repeated over many columns and the dictionary.

This method has high computational performance.

Q2.1: Extracting Features

This function generates a histogram showing how many times each visual word appears in an image. I used the hist function in order to generate this.

Figure 6 shows an example of a histogram from the visual word shown above.

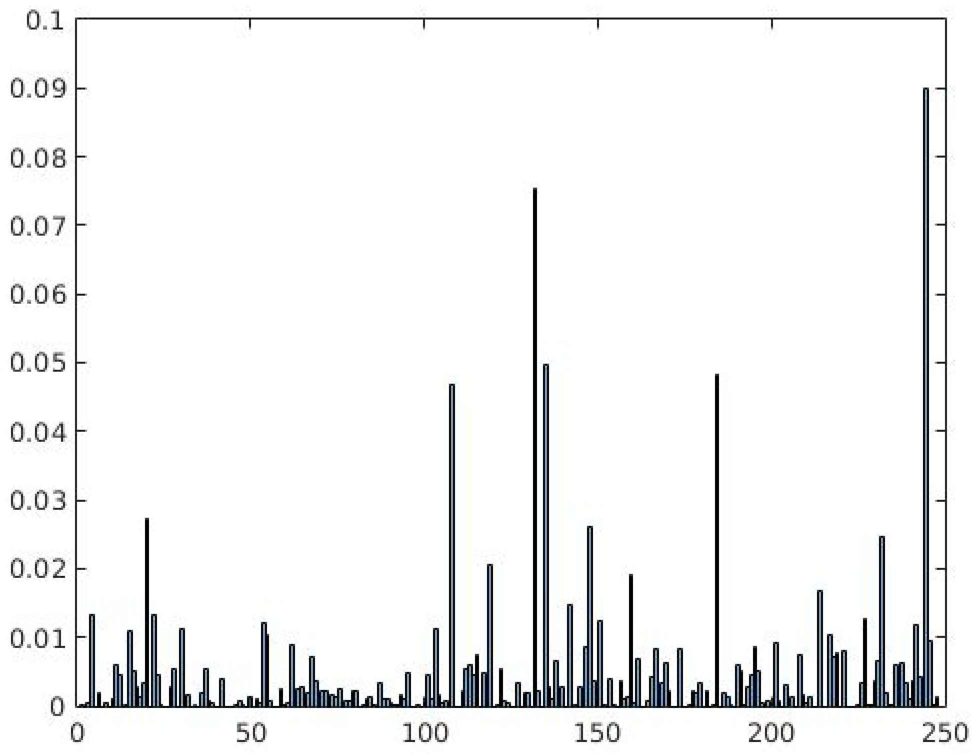


Figure 6) Histogram of Visual Words L1 Normalized for Image in Figure 5

Q2.2: Multi-resolution: Spatial Pyramid Matching

Figure 7 shows a representation of the Spatial Pyramid Matching scheme for 2 levels.

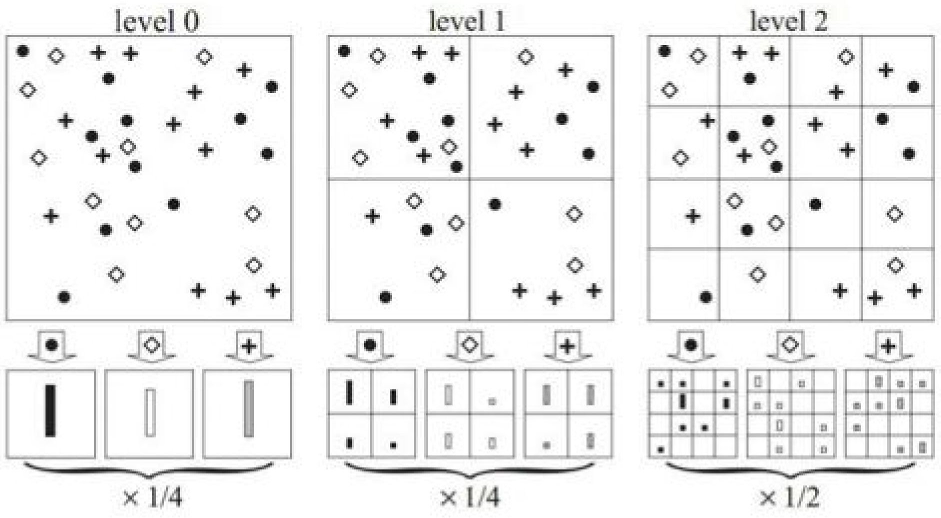


Figure 7) Pyramid Matching Scheme

Spatial Pyramid Matching attempts to utilize the spatial information of the image by concatenating histograms from each cell in each layer in one large feature vector. Each of these are weighted appropriately. The corresponding vector is the L1 normalized.

My scheme uses a loop to go through the number of layers specified and then breaks up the image appropriately for the layer number.

I did not aggregate the larger histograms from the finer layer histograms as recommended in the prompt. I computed each of the histograms each time. I did this, because it was easier to generalize this algorithm for any number of layers requested by the algorithm.

Figure 8 shows the feature vector as a bar chart.

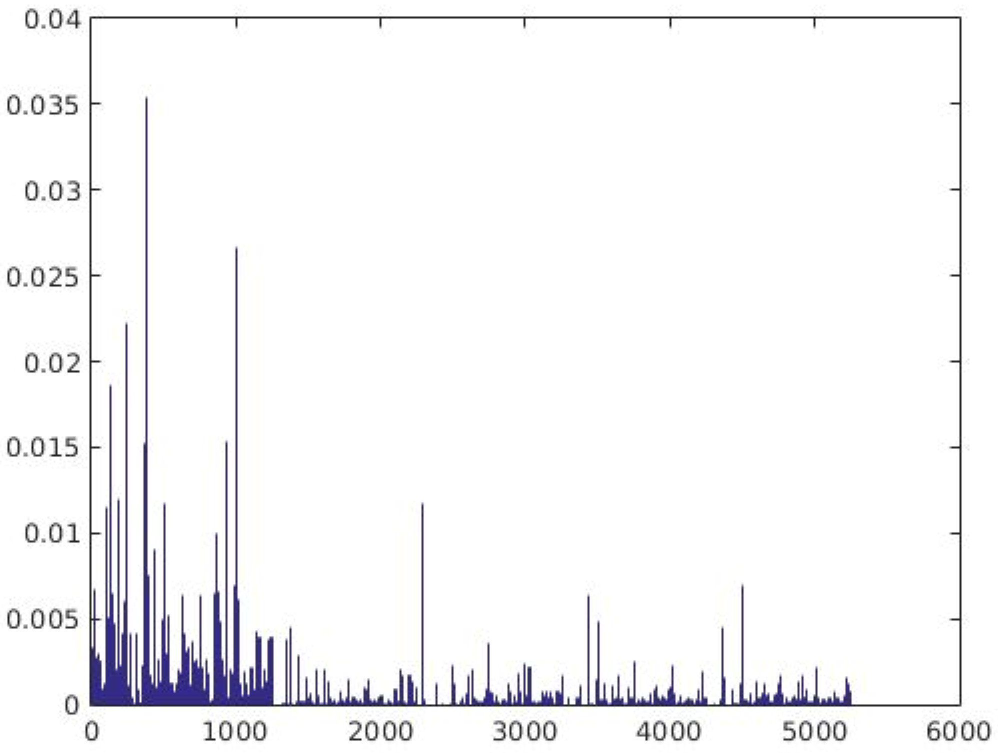


Figure 8) Spatial Pyramid Feature Vector as a Bar Chart

Figure 8 showcases that my algorithm computes the histograms of the earlier layers first. The vector here has larger values in the front where the higher weighted histograms are shown in my algorithm. These features will have a larger effect on the classification scheme than the smaller valued features.

Q2.3: Comparing Images

This section creates a way to compare two histogram feature vectors to see which it is closest to. A bunch of histogram feature vectors were calculated for the training set. Now whenever you calculate the features for the test set, you can compare them with a distance function in order to find the “closest” match.

The distance function used is:

$$\sum_{i=1}^{n} \min(x_i, y_i)$$

Where:

$x_{i:n}$ and $y_{i:n}$ are two histogram feature vectors

The first thing that I did was to use repmat to make the feature vector the same size as the dictionary of the training set words.

I then used bsxfun to run the unary @min function element by element. Summing along the rows gets a similarity score of the test image to all of the training images.

Q2.4: Building a Model of the Visual World

This step involves building up the training set to be used in Q2.3. In order to build the system we must include:

filterBank: bank of filters used
dictionary: visual word dictionary
train_features: histogram feature vector for all training images
train_labels: classification labels for all training images

The filter bank and dictionary were calculated in Q1.2. The train labels were provided with the data we were given.

To get the train_features, I go through the list of training image names provided with the prompt. I load in the visual words computed in Q1.3. I then gather the histogram feature vector as shown in Q2.2. I put all of these in a matrix with the structure shown in Figure 9.

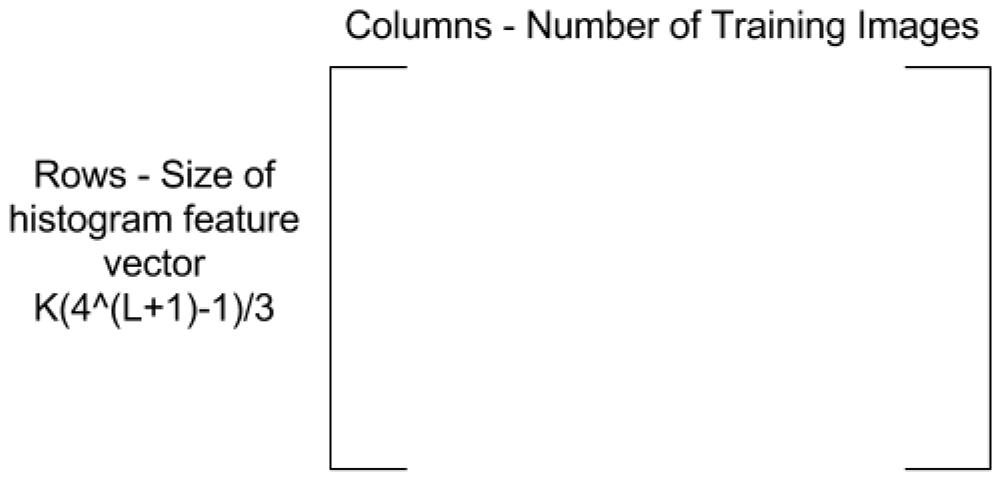

Columns - Number of Training Images
Rows - Size of histogram feature vector
K(4^(L+1)-1)/3

Figure 9) Matrix Structure of train_features

To create the train_features matrix, I used a two layer spatial pyramid matching.

Q2.5 Quantitative Evaluation

This is the section where I get the quantitative numbers on how well my training scheme works on the test set.

I loop through the list of names from the test set provided in the prompt. I load the image, get the filter response, and then get the histogram feature vector for the image.

I then use the distance metric specified in Q2.3 in order to find what training image is a closest match to the test image. The label for that training image is then predicted as the label for the test image.

Finally I update the confusion matrix.

Table 2 shows the final confusion matrix.

14	2	0	2	0	0	0	2
2	13	2	0	1	1	1	0
1	4	11	2	0	2	0	0
1	2	1	10	1	2	2	1
0	3	1	3	7	1	5	0
3	1	3	1	1	8	3	0
3	1	1	6	1	1	5	2
2	0	0	1	0	1	1	15

Table 2) Confusion Matrix

Numbers on the diagonal correspond to correct classifications while off-diagonal numbers correspond to incorrect classifications.

The accuracy can be specified by:

$$\text{accuracy} = \text{trace}(C) / \sum(C(:))$$

The total accuracy was: **51.88%**

Q2.6: Find out failed cases (Optional Extra Credit, upto 10 points)

Looking at the Confusion Matrix in Table 2 shows what are some of the classes that the classifier works well on and those that it has trouble with.

High classification accuracy corresponds to labels: ‘airport’, ‘auditorium’, ‘bedroom’, ‘campus’, and ‘airport’.

Lower accuracy was gotten when classifying: ‘desert’, ‘football_stadium’, and ‘landscape’.

The ‘football_stadium’ label was incorrectly identified as ‘airport’, ‘bedroom’, and ‘landscape’ and equal number of times. Without any sort of pattern, we can only assume that the reason that ‘football_stadium’ is more difficult to recognize is that it has visual features that are common to many other image types. The ‘football_stadium’ images are mainly green fields and seats. These are features that correspond to ‘landscapes’ and ‘airports’ frequently. Another problem is that because stadiums are very large, the vantage point of the camera greatly changes the types of features in the image. Figure 10 shows an example of this.

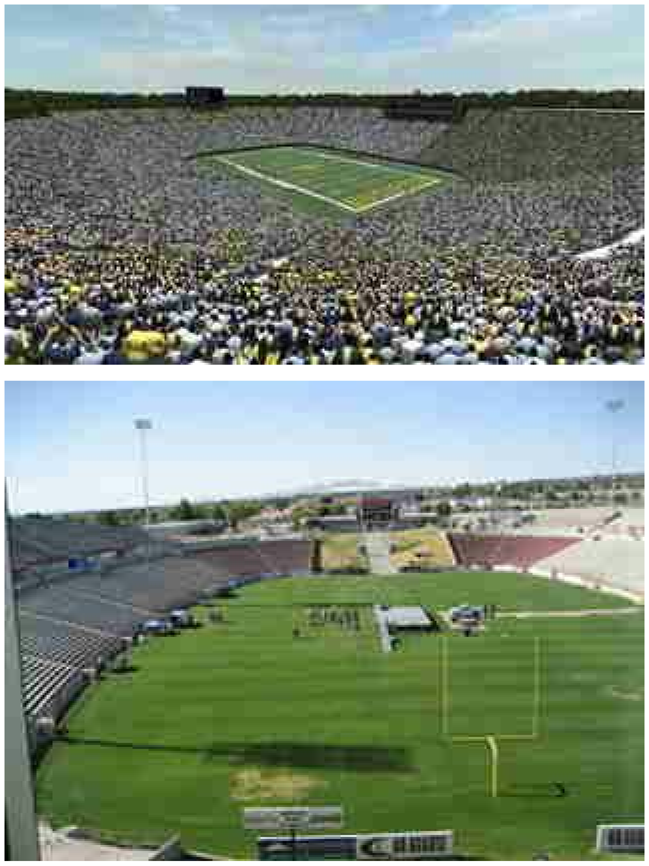


Figure 10) Two ‘football_stadium’ Images

Figure 10 shows that the large variations in image features from the two will cause problems for classification.

Another conclusion obvious from the two images is that there are similar features in both images to other images in the set. The first image has a dense population of people that is similar to features found in the 'airport' images, while the second image has features that would be found in many 'landscape' images.

Another image class that is misclassified often is the 'landscape' class. There was a class it was misclassified as more often than others: 'campus'.

Figure 11 shows a landscape and campus image that showcase the failure modes of the 'landscape' class.

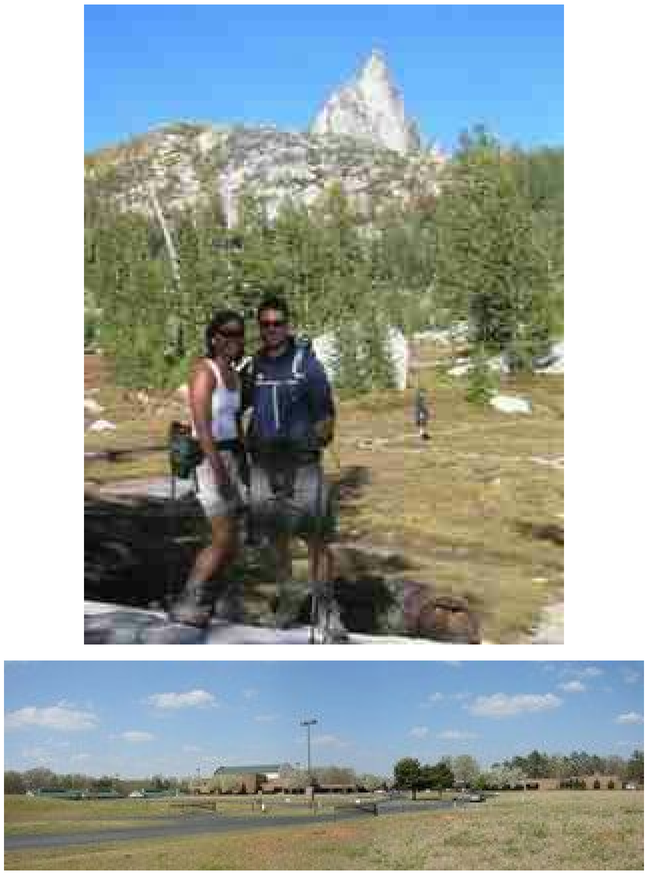


Figure 11) Top: Landscape; Bottom: Campus

As figure 11 shows the color scheme of the 'campus' and the 'landscape' can be very similar. Because the 'campus' and 'landscape' images are showcased at many different landscapes (snow, desert, field, etc.), many of their image features would be similar. Often, there is not much in the image that would distinguish a 'landscape' and a 'campus' image. Other times buildings and mountains are present that will distinguish.

This showcases a failure mode. If a training set has only a few distinguishing features, test images without those few distinguishing features will not be recognized correctly. There are common features in most images (grass, trees, people, etc.) that are in most images. If a test image or a training set is composed of mostly these common features, the accuracy testing against that set will be low.

The final class that had poor classification performance was the ‘desert’ class. It was often misclassified as ‘landscape’. The reason was similar to what was shown in Figure 11. There are not enough distinguishing features in the ‘desert’ training set for high classification modes.

These are the major failure cases in the classification scheme.