Project 3 Writeup

Instructions

- Provide an overview about how your project functions.
- Describe any interesting decisions you made to write your algorithm.
- Show and discuss the results of your algorithm.
- Feel free to include code snippets, images, and equations.
- List any extra credit implementation and result (optional).
- Use as many pages as you need, but err on the short side.
- **Please make this document anonymous.**

Project Overview

In this project, we created a variety of ways to evaluate and classify different scenes based on feature labels. First, we processed the image. Then, based on these images we created a vocabulary of words that we then used to create histograms to classify features extracted from the image. Finally, based on these histograms we created methods to compare each feature to the vocab words to best classify the image.

Implementation Detail

In our implementation, we created three different recognition schemes. The first used the function get_tiny_images() to represent the images as the array of the image vectors. From these image vectors, we can then use _nearest_neighbor_classify() to compare these image vectors to a variety of training labels and images. The training label that is closest to the vector will indicate how the image is classified. In the second implementation, we use a bag of words representation of the image by creating the functions build_vocabulary and get_bag_of_words. Based on common features that are found through the KMeans clusters, we can create vocabulary histograms that we can then use in our _nearest_neighbor_classify() to find the closest label. Finally, we can also use this bag of words representation with the SVM classifier, which uses trained SVM classifiers to classify the test images created with the bag of words.

My code snippet highlights an interesting point. In this project, it was super important to accurately build the returned arrays. One issue I ran into was that by appending to arrays rather than indexing, I was creating the wrong shape. So, I fixed this by writing the code below.

```
# create array with rows equal to number of images, number of columns
                                    equivalent to vocab size
    finalArray = np.zeros((num_imgs, vocab.shape[0]))
    ...
    #add histogram to final array
        finalArray[i, :] = vocabsort
```

Result

1. My tiny images and nearest neighbor set up resulted in a 19.933% accuracy rate.
2. My bag of words and nearest neighbor set up resulted in a 39.600% accuracy rate.
3. My bag of words and SVM set up resulted in a 41.133% accuracy rate.

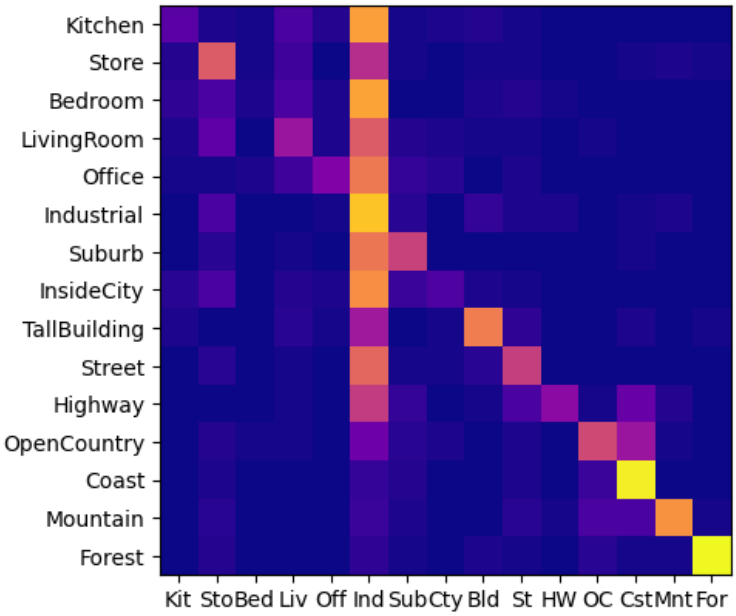

Figure 1: A
bove is the confusion matrix for my bag of words and SVM set up