Image and Vision Computing Lab-3

In this lab we will classify photo images into different scene types (Kitchen, Bedroom, Office, etc). There are two parts that will explore using two different representations (raw images, and SIFT Bag of Words) and two different classifiers (KNN and SVM). Specifically, we will classify using the 15 categories of images in the Lazebnik et al 2006 CVPR paper, and work towards building one of the baseline methods evaluated in that paper (SIFT Bag of Words with SVM).

Acknowledgements: The idea for this task is borrowed from the computer vision homework assignment set by Prof. James Hayes at Brown. This is a simplified version of James' original assignment.

Part 1. Tiny Images and NN Classifier

The first task implements a simple version of the nearest neighbor classifier for image classification. Load lab3_utils.py. This file has the code to find all the image files on disk (get_image_paths). This function also lets you choose the number of training samples per class. To start, use 100 samples per class. The utils.py file has functions to read them from disk and store them as small 16x16 thumbnails. Take a look at these files and make sure you understand how they work. Here you already have the representation and your main task is to implement a nearest neighbor classifier in function nearest_neighbour. This routine should accept the NxD array of training images train_image, the Nx1 cell array of training image labels (train_labels) corresponding to the input images, and the MxD array of test images (test_image). It should output predicted_labels, the Mx1 cell array of test image labels estimated by your NN classifier.

Recall that a NN classifier estimates the label of an input test image by:

1. Comparing the testing image to all training images,
2. Finding the most similar (closest) training image,
3. Predicting the label of that closest training image.

Interpreting the results: A basic implementation of a 1-NN classifier here should get about 19% accuracy with 100 training samples per class. This is not great, but significantly higher than the chance level of 1/15=6.6%. Start by implementing a 1-NN classifier and you can update it to then be a k-NN classifier. Additionally, you can have a look at the effect of using different k values and the overall accuracy.

Bonus: The basic 1-NN classifier can be improved in various ways to get higher than 19% performance. Try: (i) Using more of the training data than the fraction used in the example code, (ii) Varying the image thumbnail size used, (iii) Standardizing the scale of the data before classification, (iv) Exploring what other distance metrics we can use.

Part 2. SIFT BOW and SVM

The second task is to implement a simple version of the SIFT Bag of Words SVM classifier. Load up bow_template.py. This file has the code to find all the image files on disk (get_image_paths), and classify the encoded images with SVM (svm). What is missing here are the methods to train the codebook (build_codebook) and to use the learned codebook to turn each image into a bag of words representation (bag_of_words).

In build_codebook(), you should loop over all the image paths, load up each image, and call cv.SIFT_create() using which we can then call sift.detectAndCompute() which detects and computes descriptors for each image. Using this, we should pass them through a k-means function (can use built-in sklearn.cluster). The function should return the cluster centers. Using the codebook, you should represent the images as bag-of-words.

Interpreting the results: A basic implementation of a SIFT-BOW + SVM classifier here should get about 24.5% accuracy, higher than the 18% from 1-NN.

Bonus: If you reached this far you have implemented a simple SIFT-BOW pipeline. Correctly tuning all parts of this pipeline (well tuned SIFT extraction, well tuned codebook, well cross-validated SVM) should be able to push the accuracy to around 70%.
Bonus: Alternatively, you can try plugging in a deep feature extractor and see how it performs.
Bonus: Also, try using HOG descriptors instead of SIFT as this would help with your project.