Computer Vision

Image Categorization

Name: hanxue Liang

Student ID: 16-912-156

In this exercise, I implemented an object recognition algorithm using the bag-of-word framework.

1. Local feature extraction

For local feature extraction, firstly, we need to compute the grid points for an image. To do this, we calculate the space between two points in the x- and y-direction given the size of the image, the border we leave on each side and the granularity. Then, we store the points' coordinates in a nPointsX*nPointsY x 2 matrix.

For the histogram of gradients (HOG), we added pi to all negative angles because it still represents the same line. So that our histogram is for values between zero and π. And I also normalize the magnitudes within each cell so that the disparity between different cells will not be a problem.

2. Codebook Construction

In this part, we create a codebook using K-means clustering to find the k-center of all the features extracted from positive examples. First, we find the nearest neighbor (in Euclidean distance) between all the features and the set of k randomly assigned cluster centers. Then we shift the position of each cluster center to the mean of its assigned points. We repeat this algorithm for a defined number of steps. With all the car images provided, the output of the *visualize_codebook* is shown in Fig.1. We do recognize some parts of a car, mainly the edges.

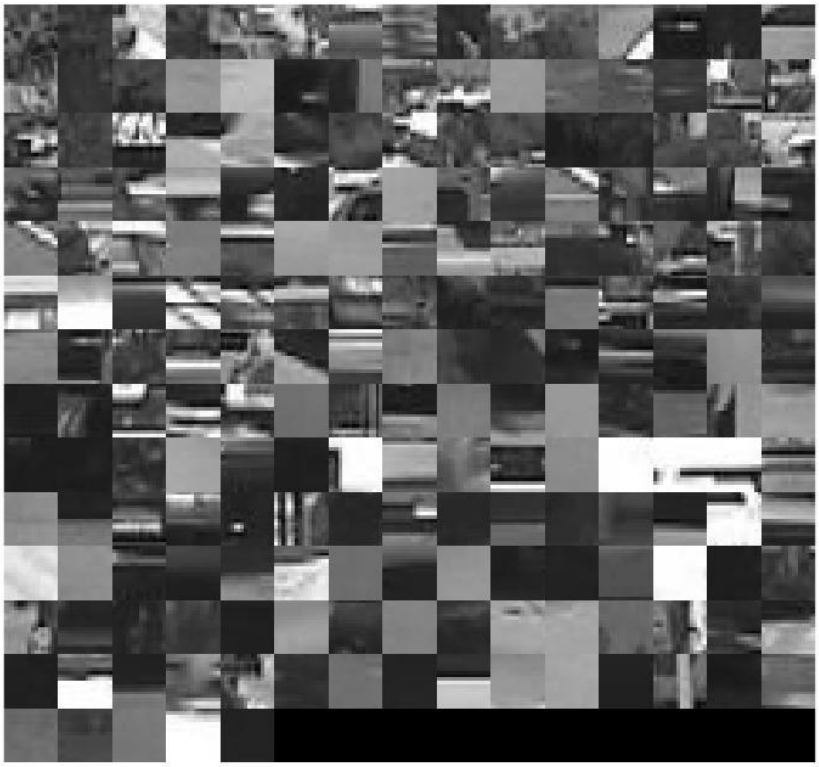

Figure1: Example of an obtained codebook

3. Bag-of-words Image Representation

In this part, given a set of descriptors extracted from one image and the codebook of cluster centers, we match each descriptor to the nearest cluster center. Then we count the number of descriptors for each cluster center and this composes the histogram for one image. For all positive and negative training examples, we compute their histogram and construct histogram box for positive examples (*vBoWPos*) and negative examples (*vBoWNeg*).

4. Classification

Nearest Neighbor Classification

In this method, we compare the smallest distance between an image's BoW histogram and the histograms box of the positive and negative sets. It assigns the image to the closest set.

Bayesian Classification

In this method, we implement a probabilistic classification scheme based on Bayes'

theorem. Firstly, we implement a simple function *computeMeanStd* to get the mean and standard deviation of the positive and negative observed histograms from the training images. Then for a test sample, we can now compute the joint likelihoods P(hist | car) and P(hist | !car) assuming that the words appear independently. We assume the prior distribution P(car) and P(!car) is both 0.5. Finally, we use bayes' theorem to get P(car | hist) and P(!car | hist). And assign the image to the set with bigger probability.

5. Results

To evaluate the performance of the two classifiers we took the mean of multiple tests because of the random and deterministic parts of the code.
When we choose size of codebook to be 200 and number of iteration to be 20, we find that the mean is 0.9582 and standard derivation 0.0212, the Bayesian classifier gives us a mean of 0.9183 and a standard derivation of 0.0198.
The nearest neighbor classifier is slightly better than that of Bayesian classifier. The nearest neighbor classifier has a much higher complexity. In our example, when the test data is very similar to training examples, the NN classifier can provide a good result. By contrast, Bayesian classifier is much faster but there are several defects in our case. We assume conditional independence between different features, which makes it slightly less efficient in this case. And the training data is too small, which make the average and derivation of the features not precise.

When we tune down the number of codebook from 200 to 100, the result of nearest neighbor begins to fall down and the Bayesian classifier begins to raise up to 96%.
The raising of the result of bayesian classifier can be explained in such a way: as we computer cluster centers and calculate the mean and derivation for different centers, the result becomes more accurate when we choose less cluster centers. Meanwhile, the difference between the negative example and positive example becomes bigger. By contrast, for NN classifier, we choose less cluster centers, so the classifier function has a less number of complexity. So its performance becomes worse in the case that test data has a good similarity with the training data.

6. Bonus

I try the algorithm in my dataset. For the positive samples, I downloaded the image from http://imagenet.stanford.edu/internal/car196/car_test. For the negative samples, I downloaded the image from cityscape dataset https://www.cityscapes-dataset.com/downloads/. Each set contains 100 images.
It turns out that for Nearest Neighbor Classification method, I find the mean is 0.84562, the Bayesian classifier gives us a mean of 0.91929. The bad performance of NN method can be explained that the there is smaller similarity between the testing and training dataset that NN method fail to find the corresponding features, especially for negative samples. While Bayesian classifier can still provide good accuracy.