Report of Anton Maksimov (antonma, 16-952-137), Task 10 "Image Categorization" on ETHZ course "Computer Vision".

Creating codebook.

First we extract features from all training images just creating grid of 10×10 points and calculating HOG in 4×4 cells near this grid points by calculating histograms of gradient directions in 4×4 pixels in each of these cells. We use 8 bins for histograms, so map $[-\pi, \pi]$ to integers from 1 to 8 using division by $\pi/4 + \epsilon$ ($\epsilon = 10^{-10}$ in order to map -2π correctly, without using special bin for it) and `floor` the result with shifting to positive integers by summing with 4. In order to have all pixels inside the image we create the grid with borders 8 pixels from each side. We concatenate all histograms from all cells to one array per grid point.
Then, we cluster these arrays from all grid point from all training images using kmeans with k clusters (fig. 1 ??). Later we vary this k from 3 to 1000 (where results are good before dropping).

Bag of words

To create bag of words we take testing images (separately positive and negative), extract patches as before, but then find the nearest in euclidean metric cluster center from codebook and add one to correspondent bin in bag-of-words (BoW) histogram for each training image. So, we have positive (PosBoW) and negative (NegBoW) sets of histograms.

Classification

Nearest neighbor

For each testing image we use `knnsearch` to find nearest BoW of negative and positive training images, and then assign positive label, if distance to the nearest positive one is less than to the negative one.

Bayesian

We model distributions of histograms for one Word in PosBoW and NegBoW as normal with inferred mean and std of the extracted empirical ones. Then we assume all these words to be independent and model probability of being our image positive from Bayes' formula assuming prior probability to find a car as 0.5. Because small probabilities multiplied many times can lead to numerical instabilities, we calculate sum of natural logarithms of pdfs calculated in point — number of appearances of the word in this particular image, when pdf's mean and std are taken from correspondent positive or negative distributions in testing images.
Also, in order to avoid NaNs we skip in the sum words which have zero mean and std.
As a result, assuming denominator not zero in Bayes' formula (it's almost impossible with normal images), we compare numerators for positive and negative variants and choose the biggest one's variant (negative if equal).

Results

We obtain fraction of rightly classified training images, repeating the algorithm for 10 times (then better quality of statistics) for different size of codebook (see the table 1) (full results in .xlsx file in results folder).
As can be seen from results (fig. 2 ??), in provided dataset nearest neighbor classification algorithm works more stable in wider range and comparable with Bayesian one + it has less fluctuation. But Bayesian is better with small number of words (even the best is with 5-10 words), because these are the most significant, but nearest neighbor is worse there because of too small density of centers in space, so it's more usual to assign to far centers, which have lesser common with current image).

Table 1: Results for provided dataset

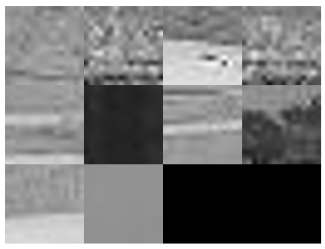
Codebook size	Nearest neighbor		Bayesian	
	mean	std	mean	std
10	0.89193	0.05588	0.92526	0.037302
25	0.88789	0.039493	0.86062	0.037737
50	0.91314	0.023557	0.89294	0.035908
75	0.90506	0.035336	0.92627	0.040664
100	0.91213	0.048868	0.87577	0.044726
200	0.89496	0.038908	0.89092	0.034636
500	0.85961	0.02837	0.7677	0.034102
1000	0.84547	0.057143	0.56065	0.026434

Bayesian approach doesn't work with big number of words because then there are less appearances of one word and less of them could be approximated as normal (because normal model, probably, comes from the law of big numbers, when there are big numbers in histograms).

In general, we can't say that one approach is better than another from our results, because among sizes of codebook where both work the best, their difference is insignificant due to estimated variance.

Bonus

Also, we collected in Zurich and Dietikon our own dataset (google disk) of images of cars «in profile» as positives and images of road and surroundings as negatives (60 images — for 15 in each of positive–negative and training–test). In order to compare with provided dataset we resize images to to comparable size (10 times less in each dimension from 3928 × 2208). Other parameters are the same. (we changed names of folders in `bow_evaluation.m`, and ".png" to ".jpg" plus resized images in `create_bow_histograms.m` and `create_codebook.m`.
Because this dataset is smaller, variance of results is bigger, in general (discretization error (due to discrete possible values of resulting ratio) was $1/100/2 = 0.005$ before, now it' $1/30/2 \approx 0.015$ only because of the smaller number of data). Also, results for Bayesian labeling are better, because, maybe, it's more uniform dataset, so distribution among histograms is better modeled with Gaussians. Plus, images are of a better quality even after resizing. (fig 3**??**, table 2)



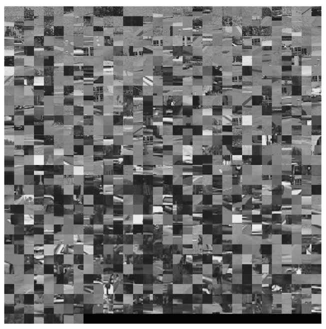


Figure 1: Codebooks for 10, 100 and 1000 patches (black squares in the last line are just filling, it's not extracted patches).

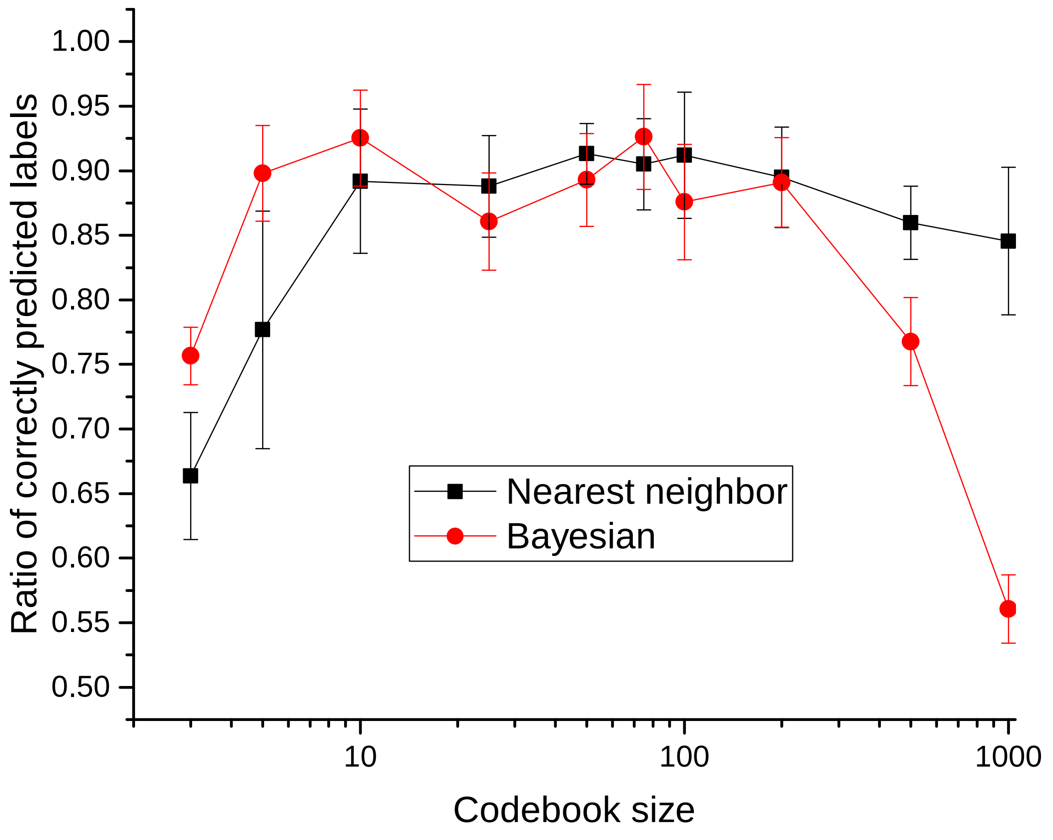


Figure 2: Results for two types of label assigning for provided dataset.

Table 2: Results for self-obtained dataset

Codebook size	Nearest neighbor		Bayesian	
	mean	std	mean	std
10	0.89001	0.039575	0.93999	0.02002
50	0.90667	0.046669	0.94668	0.037123
100	0.93667	0.043333	0.95334	0.034003
200	0.90335	0.058592	0.95001	0.04014
500	0.91999	0.042695	0.86667	0.055763

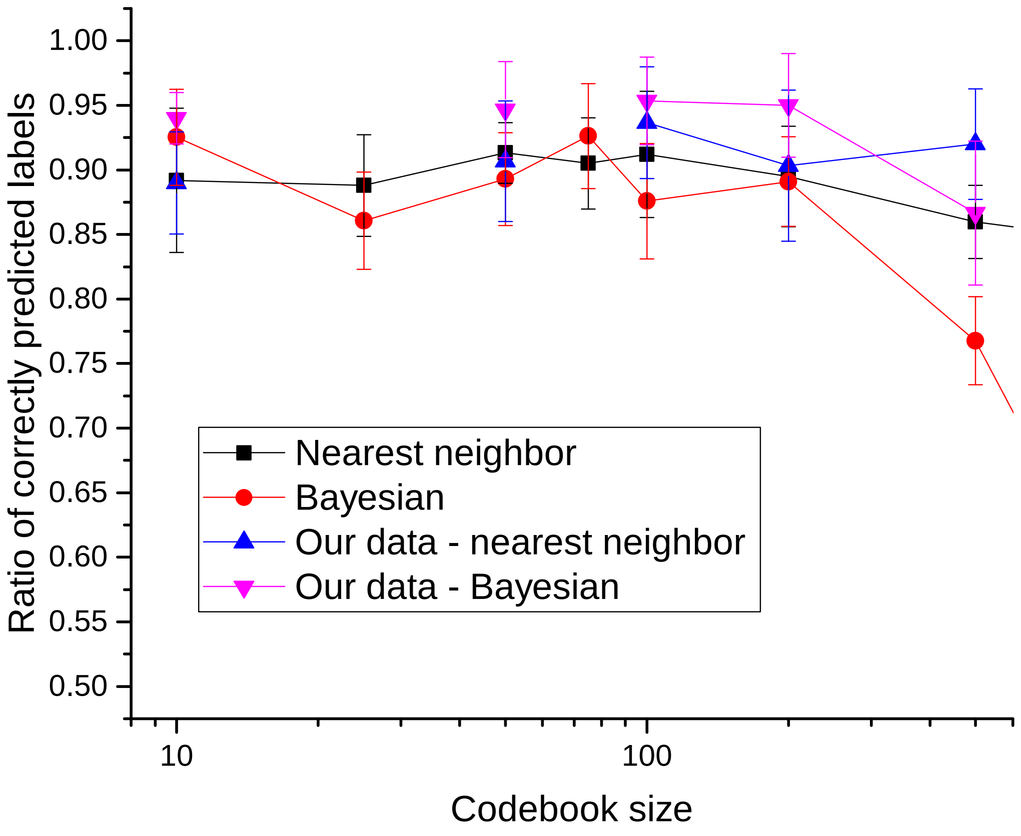


Figure 3: Results for two types of label assigning for self-obtained and provided dataset.