Entiol Liko: Lab 4

enliko@student.ethz.ch

1 Bag of Words Classifier

We briefly describe the implementation for each of the subtasks for the question 1: Implementing a bag of words classifier. Then we give results and observations for the task.

1.1 Feature points on a grid

This is implemented in the grid points() function. We have to divide each image into a nPointsXÃ nPointsY grid, where nPointsX = nPointsY = 10 We leave out border number of points from the beginning and end of each axis. I calculate the step for each direction by dividing the width/height - 2*border by the number of required points. I add each coodrinate to a list which is then returned as the result of the function.

1.2 Histogram of oriented gradients

Now our task is to create a feature descriptor for each image. This is implemented in descriptors *hog*() function:

1. For each image we have the set of 10×10 grid points. For each of the gridpoints we consider a neighbourhood of pixels (given by a rectangular patch centered at the grid point of interest). The rectangular patch is divided into 16 cells, each cell itself consisting of 16 points.
2. For each of the 16 cells, we have to compute the histogram of oriented gradients. The final feature vector for each gridpoint is given by concatenation of histogram for each of the 16 cells.
3. For each of the $4 \times 4 = 16$ points in each cell, we have to compute the orientation of the gradient. This is given by an angle in the range $[0, 2\pi)$. We are only interested in orientation being in one of 8 equidistant directions. Also, due to circular nature of phase, phases rounded to 2π and 0 are equivalent and this must be taken care of.
4. The above is performed for each of the 16 points in the cell. This is done for each cell, and there are 16 cells in total, thus each gridpoint has a 128 d descriptor.
5. We have a 100×128 descriptor for each image, since there are a total of 100 gridpoint locations.

1.3 Codebook construction

This function and the following are mostly pre-implemented. Here we just call the 2 functions I implemented above. The calls I made are the following:
$vPoints = grid_points(img, nPointsX, nPointsY, border)$
$descriptors = descriptors_hog(img, vPoints, cellWidth, cellHeight)$
$vFeatures.append(descriptors)$

1.4 Bag of words histogram

In the above section, we obtained the k means cluster centers. Each image has a number of points, in this case 100 points in total. For each point we compute the "word" using the centers of the k means clustering we did in the previews step. After the procedure we have a list (histogram) of "word" for each point of the image.

1.5 Nearest neighbour classification

Now that we have a histogram for an image we can call $finddnn$ to find the image which has the closest histogram to our image. We give the image the same label as the found image. This is out prediction.

1.6 Results and Observations

I found that I got good results with $k = 7$ and $numiter = 10$

```
 03/exercise4_object_recognition_code/bow_main.py"
creating codebook ...
100%|| 100/100 [00:23<00:00,  4.26it/s]
number of extracted features:  10000
clustering ...
creating bow histograms (pos) ...
100%|| 50/50 [00:12<00:00,  4.16it/s]
creating bow histograms (neg) ...
100%|| 50/50 [00:12<00:00,  4.15it/s]
creating bow histograms for test set (pos) ...
100%|| 49/49 [00:11<00:00,  4.15it/s]
testing pos samples ...
test pos sample accuracy: 1.0
creating bow histograms for test set (neg) ...
100%|| 50/50 [00:12<00:00,  4.16it/s]
testing neg samples ...
test neg sample accuracy: 1.0
```

2 CNN based classification

I implemented the architecture as specified in the assignment description. The code is very similar to the code I wrote for the previews lab. The following is the result, I got a final score of 82.49%

```
sions, a channel dimension,
  warnings.warn(warn_msg)
79it [00:18,  4.34it/s]
test accuracy: 82.49
```