CENG 483

Introduction to Computer Vision

Spring 2018-2019

Take Home Exam 2

Object Recognition

1 Local Features

- SIFT vs Dense-SIFT: As the name suggests, Dense-SIFT densely extracts features from every location and scale of an image by using the actual SIFT algorithm. Extracting more useful information than SIFT for the same images, Dense-SIFT increases the chance of success in matching or recognition tasks.
- D-SIFT: kNN: 16, step: 3
  - k-means 16: 0.222
  - k-means 32: 0.293
  - k-means 64: 0.330
  - k-means 128: 0.373
- D-SIFT: kNN: 16, step: 5
  - k-means 16: 0.203
  - k-means 32: 0.247
  - k-means 64: 0.288
  - k-means 128: 0.305
- Interpretation: "step" parameter is used to control the density of sampling. This parameter determines how much displacement in horizontal and vertical directions from one feature center to the other will be made. To observe the effect of step size, step sizes 3 & 5 were used. While it is possible to extract around 64 descriptors per image by using a step size of 3, the number of descriptors reduces to 25 when a step size of 5 is used. Since larger step sizes imply smaller density for features, the outcome of decreasing density with the larger step size is lower accuracy.

2 Bag of Features

- Bag of Features Implementation:
  1. Detect features and extract descriptors from the image by using SIFT or Dense-SIFT
  2. Cluster all of the descriptors extracted to construct/learn the vocabulary by using K-means clustering

3. For every image:
   - Count how many times each visual word of the vocabulary occurs in the image,
   - Construct a histogram (one for each image) from the numbers of occurrences of every visual word in the image.

   Each histogram obtained is the Bag of Features of the corresponding image.

- From descriptors to dictionary:

  1. for every image:
     - descriptors=sift(image) (or dsift(image), parameters are omitted)
     - all_descriptors.append(descriptors)
  2. dictionary = kmeans(all_descriptors, k)

- From dictionary to the collection of Bag of Features:

  - for every image:
    * descriptors=get_sift_descriptors(image) (or, get_dsift_descriptors(images))
    * for word in dictionary:
      · frequency=descriptors.count(word)
      · bof(i)=frequency (assume word's index in the histogram is i)
    * bof=normalize(bof)
    * bofs.append(bof)

- SIFT: kNN: 16

  - k-means 16: 0.157
  - k-means 32: 0.178
  - k-means 64: 0.167
  - k-means 128: 0.159

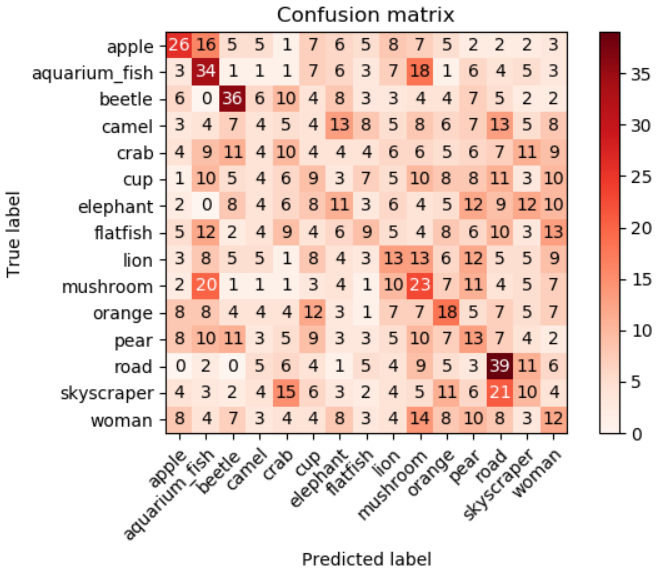
- D-SIFT: kNN: 16, step: 3

  - k-means 16: 0.222
  - k-means 32: 0.293
  - k-means 64: 0.330
  - k-means 128: 0.373

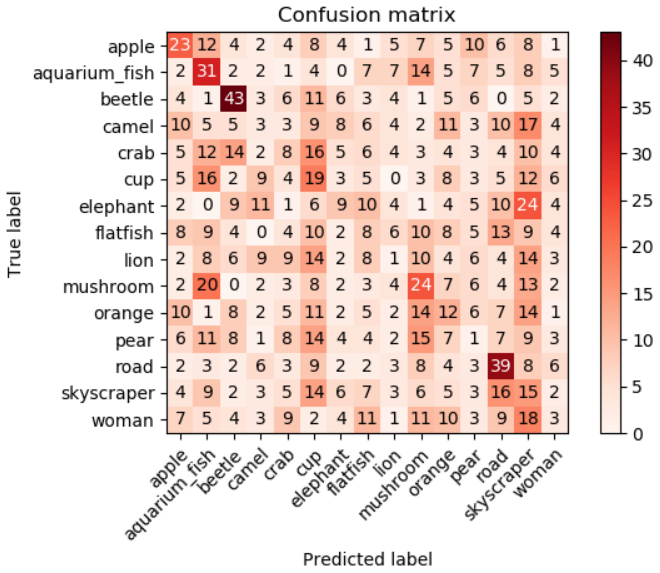
- Interpretation: Since SIFT yields fewer descriptors for the same images in the tests with the same kNN and k-means values, D-SIFT provided better results. Moreover, because of the small number of SIFT features and sparse clusters, after the k-means value passes 32, quantization errors start to affect the results of SIFT. However, in the D-SIFT case, despite the increasing number of clusters, it is still possible to find many descriptors to distribute each of the new clusters. This effectively means that the number of words in the dictionary, and, at the same time, the number of overall features used in BoF representation increase. As a result of that, the performance of D-SIFT increases consistently up to k-means value 128.

3 Classification

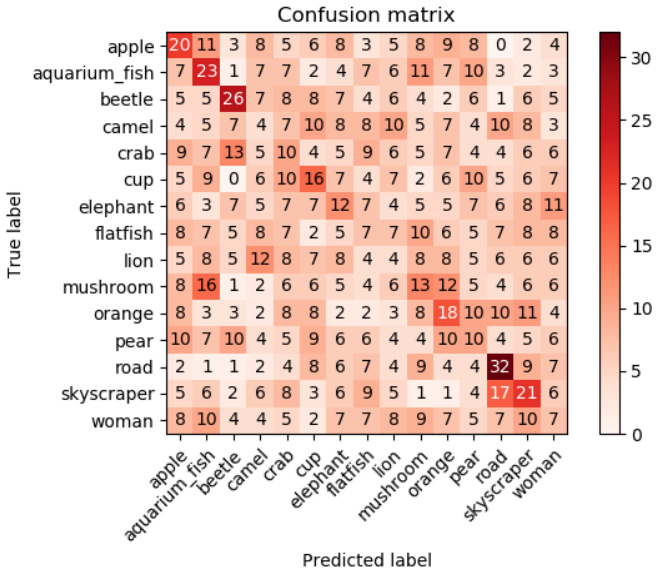
- SIFT: k-means: 128
  - kNN 1: 0.139
  - kNN 2: 0.139
  - kNN 4: 0.149
  - kNN 16: 0.159
- D-SIFT: k-means: 128, step: 3
  - kNN 1: 0.303
  - kNN 2: 0.303
  - kNN 4: 0.339
  - kNN 16: 0.373
- Interpretation: The majority voting is performed on the k-many label results provided by kNN to select top-voted label as the predicted label. In case of equality, the label whose BoF has the smallest distance to the BoF of the test image at the first occurrence in the sorted kNN list is chosen. This approach is the reason why kNN 1 and kNN 2 values give equal results. Although kNN 2 gives 2 results, these results will either directly yield a single label which is the same as the result that kNN 1 would predict, or result in two different labels suggesting again the same label as kNN 1 due to the choice of the first label in case of equal votes. It is expected that as k increases in kNN, accuracy is likely to increase because of the elimination of possible outliers in the long run. However, after a certain value, which varies with the parameter, if we increase k value further, it is likely to introduce outliers, instead of eliminating, which causes lower scores in this case. Only the positive effect of increasing k can be seen here. I tested with a few more k values in D-SIFT with k-means: 128 and step: 3 configuration. For this specific configuration, kNN: 16 happened to be the optimal value among the values tested. (kNN: 8 $\rightarrow$ 0.355, kNN: 17 $\rightarrow$ 0.370, kNN 19 $\rightarrow$ 0.368, kNN: 24 $\rightarrow$ 0.355, kNN: 32 $\rightarrow$ 0.365)
- Accuracy is a metric used to evaluate the performance of classifications. Better accuracy indicates a better classification performance. It is the ratio of correct predictions to all predictions made (for multiclass classification case). $\frac{\text{\# of Correct Predictions}}{\text{Total \# of Predictions}}$. In the code, this calculation is being performed on confusion matrix with the following line of code: *np.trace(matrix)/np.sum(matrix).*
- Confusion Matrices Obtained by Using Different Parameter Configurations:



(a) kNN: 16, k-means: 32

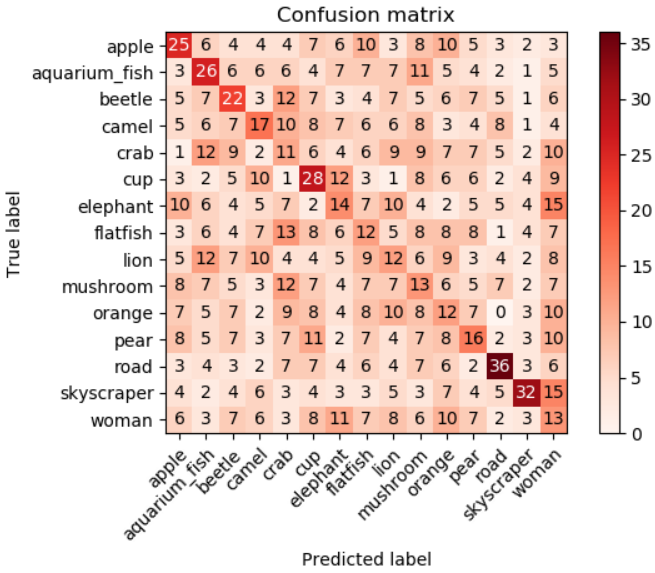


(b) kNN: 16, k-means: 128

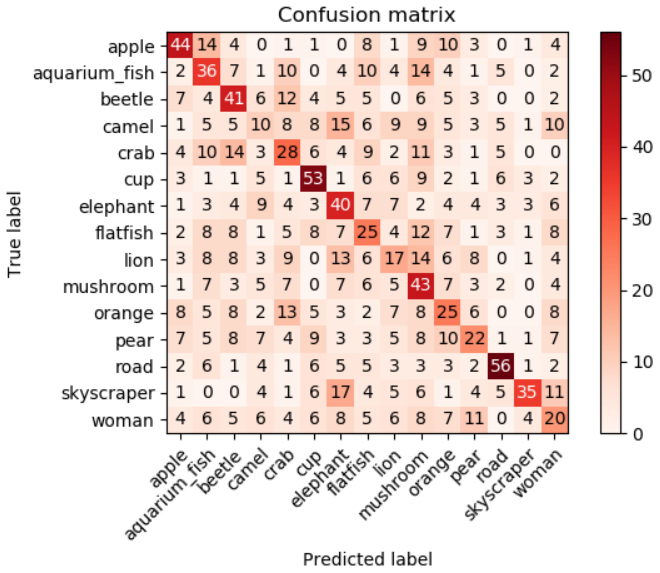


(c) kNN: 4, k-means: 128

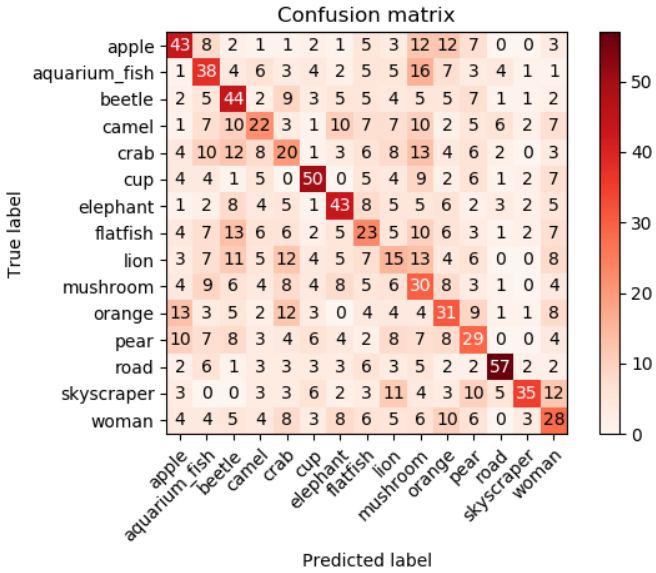
Figure 1: Confusion Matrices Obtained Using SIFT



(a) kNN: 1, k-means: 16

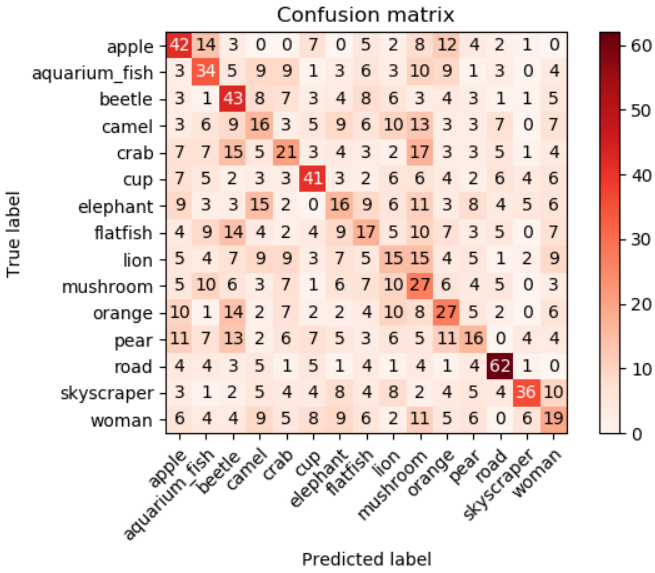


(b) kNN: 16, k-means: 64

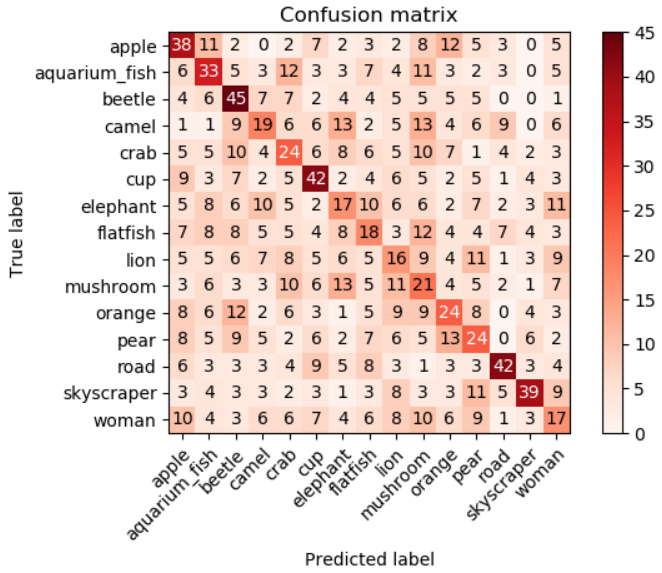


(c) kNN: 4, k-means: 128

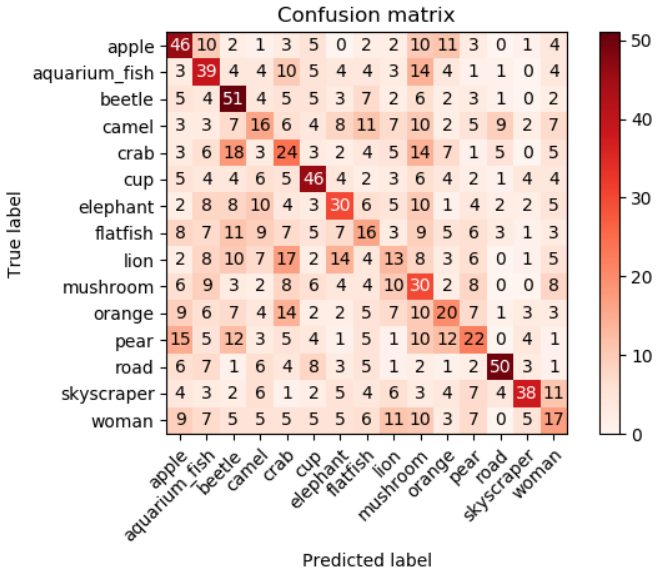
Figure 2: Confusion Matrices Obtained Using D-SIFT with step 3



(a) kNN: 16, k-means: 64



(b) kNN: 4, k-means: 128



(c) kNN: 16, k-means: 128

Figure 3: Confusion Matrices Obtained Using D-SIFT with step 5

4 Your Best Configuration

- The best accuracy achieved is: 0.373 with configuration: D-SIFT, kNN: 16, k-means: 128, step: 3
- The aforementioned configuration is decided based on the observations and conclusions stated in the discussions made regarding the effects of each parameter on the classification accuracy in the earlier sections:

  1. D-SIFT is advantageous over SIFT because of the number of features extracted. More features lead to better characterization of images.
  2. As the number of features extracted increases, it is possible that the number of different classes

of features also increases which, in turn, requires more clusters to group possibly new classes of features obtained.

3. Up to a certain k value, increasing k in the kNN can eliminate the negative effects of possible outliers on the overall accuracy by gathering more votes from possible genuine predictions to have a more reliable overall prediction.

- Command to run for the best configuration with full execution of training and test modes:

```
./the2.py --percent 5 --save --dense --stepsize 3 -p full \
--clusters 128 --trainfolder the2_data/train \
--knn 16 --cmshow --testfolder the2_data/validation
```

- The execution of pipeline can be divided to perform training and test separately. Note that, in this case, training results, vocabulary and bofs, must be saved as separate txt files to be used in testing phase later.

1. Command to run the pipeline in training mode only for the best configuration:

```
./the2.py --percent 5 --save -p train \
--clusters 128 --trainfolder the2_data/train --dense --stepsize 3
```

2. Command to run the pipeline in testing mode only for the best configuration:

```
./the2.py --percent 5 --save --cmshow -p test \
--knn 16 --dense --stepsize 3 --testfolder the2_data/validation \
--bovwfile bovwfilename --vocabfile vocabfilename
```

- Some notes about flags used:
  - "--percent" flag is optional. It can be used to get notification at desired intervals during the feature extraction parts of the pipeline.
  - "--cmshow" flag is also optional. It can be used to visualize the confusion matrix obtained at the end of testing phase.
  - "bovwfilename" and "vocabfilename" are the files created after training. Note that these files created only if "--save" flag is explicitly provided and must be provided if pipeline is executed in full mode.
  - Other flags are self-explanatory. In order to see full list of flags and for further details execute script with "-h" flag.

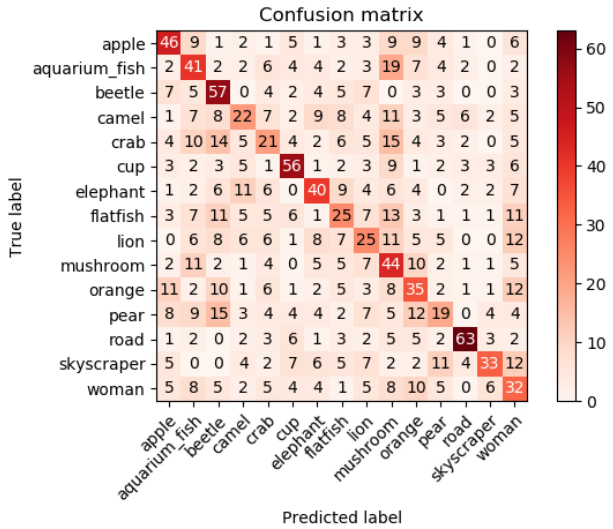


Figure 4: Visualization of Confusion Matrix Resulted from Optimal Configuration: D-SIFT, kNN: 16, k-means: 128, step: 3

- Interpretation: The highest true positives are achieved in the classification of road images. The reason might be the minor intraclass variations. In the road images in both training and test datasets, roads are generally situated in the middle of the image or somewhere close and the road edges distinctly separate roads from their surroundings. The lowest true positives occur with pear. When the confusion matrix is analyzed, it can be seen that classifier mostly confuses it with beetle and orange. The reason might be similarity in shapes. This similarity is somewhat more obvious for orange but if most keypoints extracted from beetles images are found around their body, the similarity of the shape of their body and pear's shape might be the reason for this confusion. Interestingly, though, the classifier has quite a good performance in classifying beetles and it does not confuse it with pear.

5 Additional Comments and References

- Following library functions & classes are utilized in the experiments:
  - For SIFT and D-SIFT: *sift*[1], *dsift*[2] methods under *cyvlfeat.sift*
  - For k-means clustering: *KMeans*[3] class under *sklearn.cluster*
  - For confusion matrix: *confusion_matrix*[4] method under *sklearn.metrics*
  - For plotting the confusion matrix: sample code from scikit learn examples[5]
- The *sift* and *dsift* methods of *cyvlfeat.sift* are used in the experiments with the following default configurations in addition to those explicitly stated in the previous sections:
  - sift(image, n_octaves=None, n_levels=3, first_octave=0, peak_thresh=0, edge_thresh=10, norm_thresh=None, magnification=3, window_size=2, frames=None, force_orientations=False, float_descriptors=False, compute_descriptor=True, verbose=False)
  - dsift(image, step=variable[6], size=3, bounds=None, window_size=-1, norm=False, fast=variable[6], float_descriptors=False, geometry=(4, 4, 8), verbose=False)

[1] `https://github.com/menpo/cyvlfeat/blob/master/cyvlfeat/sift/sift.py`

[2] `https://github.com/menpo/cyvlfeat/blob/master/cyvlfeat/sift/dsift.py`

[3] `https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html`

[4] `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html`

[5] `https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html#sphx-glr-auto-examples-model-selection-plot-confusion-matrix-py`

[6] These values can be set with "--step" and "--fast" flags. "--fast" flag was not used in the experiments.