CENG 483

Introduction to Computer Vision

Fall 2021-2022

Take Home Exam 2

Object Recognition

Student Random ID: 2260073

Please fill in the sections below only with the requested information. If you have additional things to mention, you can use the last section. Please note that all of the results in this report should be given for the **validation set**. Also, when you are expected to comment on the effect of a parameter, please make sure to fix other parameters.

1 Local Features (25 pts)

- Explain SIFT and Dense-SIFT in your own words. What is the main difference?
- Dense-SIFT extracts features densely from every location at the image by applying SIFT meanwhile SIFT extracts according to Lowe's algorithm. In Dense-SIFT, image is divided into grids and descriptors are extracted from every grid densely, producing better and much descriptors meanwhile SIFT decides on descriptors based on Lowes Algorithm, which generally yields worse results then Dense-SIFT on most datasets.
- Put your quantitative results (classification accuracy) regarding 5 values of SIFT and 3 values of Dense-SIFT parameters here. In SIFT change each parameter once while keeping others same and in Dense-SIFT change size of feature extraction region. Discuss the effect of these parameters by using 128 clusters in k-means and 8 nearest neighbors for classification.
- All models trained with 100 random sampled images from each category.

  - SIFT with default parameters
    -Acc : 0.150
  - SIFT with only 5 best features
    -Acc : 0.133
  - SIFT with 5 octave layers
    -Acc: 0.138
  - SIFT with 0.08 contrastThreshold
    -Acc: 0.152
  - SIFT with 15 edgeThreshold
    -Acc: 0.168

- SIFT with 1.8 sigma
  -Acc: 0.154

-As expected, limiting best features only to best 5 values decreased the accuracy.
-Since we increased octave layers, we are increasing the gaussian iterations, therefore features are getting lost thus decreasing the accuracy.
-Increasing contrast threshold yields slightly increased result since we are eliminating the low-contrasted area features which are possibly not effective as others.
-Increasing edgeThreshold yields in better results since we can extract better features by avoiding more edges, since DoG responses to edges frequently we need to eliminate them.
-Finally, increasing sigma slightly yields slightly better results, since the noise and low-frequency features are eliminated by increasing gaussian filter's strength.

- Every DSIFT is calculated with fast approximation parameter, 128 clusters and 8-Nearest Neighbours. Since fast is an approximation, it may yield worse results than expected.

  - DSIFT with step = 4 (8x8 grid)
    -Acc : 0.265
  - DSIFT with step = 8 (4x4 grid)
    -Acc: 0.193
  - DSIFT with size = 16 (2x2 grid)
    -Acc : 0.198

- Dense-sift increases the accuracy because we are now gathering descriptors from individual grids on the image, hence we can capture more unique descriptors. However, as we are increase the size of dense-sift grids, less features will be extracted since we are broadening our search area. This yields in loss of important features in bigger size windows, yet still better than the default SIFT.

2 Bag of Features (45 pts)

- How did you implement BoF? Briefly explain.
  I implemented BoF by iteration. I first loaded all the images, and after that calculated descriptors of each of them. Afterwards, I applied kmeans on them to get cluster points. Later, for each image I compared their descriptors with cluster points by using euclidian distance to measure the closest ones, and counted occurences on a histogram. after normalizing the histogram, I saved them in a text as arrays.

- Give pseudo-code for obtaining the dictionary.

```
dictionary(images, grayscales, k):
        file = open('bagoffeatures.txt')
        bofdictionary = array.empty()
        alldescriptors, items = extractdescriptors(images, grayscales)
        clusters = kmeans(alldescriptors, k)
        for image in items : {
                hist = bagoffeature(elem, clusters)
                bofdictionary.append(hist)
                file.write(hist) }
```

- Give pseudo-code for obtaining BoF representation of an image once the dictionary is formed.

```
bagoffeature(image, params):
    grayscale = convertgray(image)
    sift = sift(params)
    keypoints = sift.detect(gray)
    drawKeypoints(gray, keypoints, image)
    keypoints, descriptors = sift.detectAndCompute(gray)
    if keypoints empty:
        descriptors = zeroarray(128)
    for elem in descriptors:
        dist = euclidiandistance(clusters - elem)
        index = argmin(dist)
        hist[index] += 1
    return normalize(hist)
```

- Put your quantitative results (classification accuracy) regarding 3 different parameter configurations for the BoF pipeline here. Discuss possible reasons for each one's relatively better/worse accuracy. You are suggested to keep $k \leq 1024$ in k-means to keep experiment durations managable. You need to use the best feature extractor you obtained in the previous part together with the same classifier.
- Using Dense-SIFT with (8x8) grids (step=4),
  - k-means : 128, with 8 Nearest Neighbour -Acc: 0.265
  - k-means : 256, with 8 Nearest Neighbour -Acc: 0.281
  - k-means : 512, with 8 Nearest Neighbour -Acc: 0.250
  - Using k-means of 256, my trained kNN yields better performance and I believe that's because 256 is possibly near the elbow of the objective-function.

3 Classification (30 pts)

- Put your quantitative results regarding k-Nearest Neighbor Classifier for k values 16, 32 and 64 by using the best k-means representation and feature extractor. Discuss the effect of these briefly.
  - using Dense-SIFT with step=4 (8x8 Grids), with k-means value of 256,
    - 16-Nearest Neighbor Classifier
      -Acc : 0.264
    - 32-Nearest Neighbor Classifier
      -Acc : 0.287
    - 64-Nearest Neighbor Classifier
      -Acc : 0.295
    - 128-Nearest Neighbor Classifier
      -Acc : 0.282
  - Using the best results found in previous sections, it's seen that accuracy tends to improve up to a point (k=64) then starts to drop. This shows us that the best k-value for kNN is between 64-128. Reason for this is, the borders of clusters are probably not diversed enough so that it confuses the prediction of values which stands around borders of clusters.
- What is the accuracy values, and how do you evaluate it? Briefly explain.

– Accuracies are shown in previous section. For accuracies, I'm using the following psuedo-code:

```
accuracymetric(images, labels):
        total = 0
            for image in images:
                predictedlabel = kNN(image)
                if label[image] == predictedlabel:
                    total += 1
        return total / images.size
```

- Give confusion matrices for classification results of these combinations.

Figure 1: 16-Nearest Neighbor Confusion Matrix

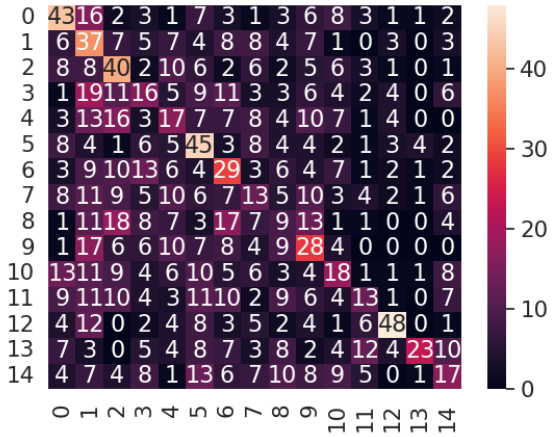


Figure 2: 32-Nearest Neighbor Confusion Matrix

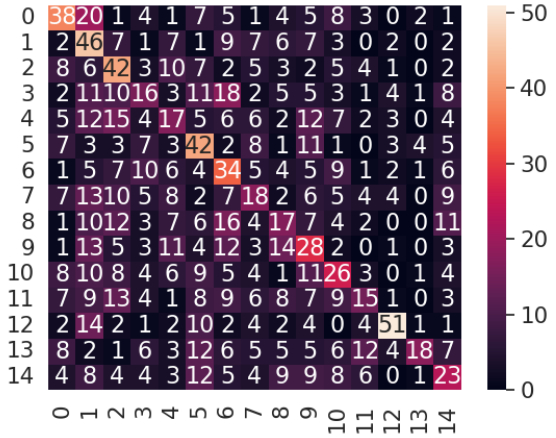
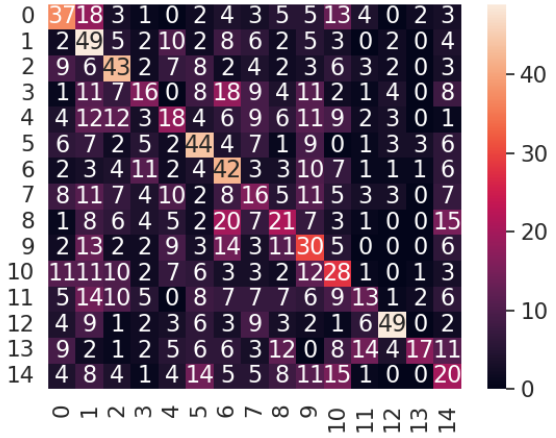


Figure 3: 64-Nearest Neighbor Confusion Matrix



4 Additional Comments and References

- Since I couldn't find an implementation in openCV for dense-SIFT, I've used VLFeat Library
- I also used sklearn.metrics for confusion matrix graphs.
- I've added a readme.txt file to guide you how to reproduce results.
- I've also added my best bag of features and cluster points in the 7z file.