CENG 483

Introduction to Computer Vision

Fall 2021-2022

Take Home Exam 2

Object Recognition

Student Random ID: 2172195

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?**

  SIFT is an algorithm used to detect the key points and the corresponding local descriptors that are invariant to scaling, rotation, noise, and illumination; by means of which comparison, and hence classification, of images can be done. In order to achieve scale-invariance, SIFT first creates a stack of images in a Gaussian-scale-space, where smoothing with different scales are applied on the image. And then, the differences between consecutive scaled/smoothened images are calculated (results are called DoG images). And each pixel is compared against its neighbors in the same DoG, and against the ones in the corresponding window in both predecessor and successor DoGs so that the scale in which the point is best represented, ie, potential key points, are extracted. Then, the points with little response and points on edges are eliminated using thresholds (via contrastThreshold and edgeThreshold parameters in OpenCV SIFT respectively). After the strong key points are detected, their rotation is calculated using gradient magnitude and directions of pixels within their neighborhood. For 16x16 neighborhood, 8-bin orientation histogram is calculated, which in the end yields a 128-bin descriptor for a key point.

  For Dense-SIFT algorithm, however, instead of operating on the entire image to find the key points and local descriptors, a window defined by step size parameter is used to calculate the descriptors. That is, Dense SIFT is merely running the SIFT algorithm on a dense grid of locations.

- **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.**

For SIFT:

We have five different parameters whose default values are indicated next to them as follows: nfeatures = 0, nOctaveLayers = 3, contrastThreshold = 0.04, edgeThreshold = 10, sigma = 1.6. And the resultant accuracy for these default parameter setting for the given number of clusters (128) and number of nearest neighbors (8) is 0.11.

*Changing the **nfeatures = 10** while keeping the other parameters as their default, **the accuracy = 0.125**
As the number of descriptors for an image increases, the potential to find false key points also increases, especially for simple images. However, keeping it 10, algorithm is able to find a few of these interesting points, which seems to be the cause the minimal rise in the accuracy. Furthermore, accuracy takes only the correct matches into account, ie, does not consider False Positives, which is another reason for the small rise in accuracy.

*Changing the **nOctaveLayers = 4** while keeping the other parameters as their default, the **accuracy = 0.125**
This parameter specifies the number of layers in each octave, that is, with more layers we get more comparison among different scales, hence we should obtain stronger key points, and hence a rise in predictive performance. However, there is a trade-off, ie, with more layers, the run time gets higher and higher. This really small change in this parameter caused a really minimal change in the accuracy.

*Changing the **contrastThreshold = 0.06** while keeping the other parameters as their default, the **accuracy = 0.13**
This parameter is used to define the threshold to eliminate the low-contrast key points, that is, potential key points with a response less than the contrastThreshold are eliminated. In other words, a greater contrastThreshold means more potential key-points to be eliminated, which should give a rise in the precision, as we would end up with key points we are more sure about. This is again a trade-off between the accuracy and precision. And a really small rise in contrastThreshold yielded a slightly better accuracy.

*Changing the **edgeThreshold = 20** while keeping the other parameters as their default, the accuracy = 0.145
This parameter is used to define the threshold to eliminate the potential key points on edges. The logic behind the relation between performance metrics and the threshold values is as explained for the contrastThreshold parameter.

*Changing the **sigma = 0.8** while keeping the other parameters as their default, the **accuracy = 0.139**
This parameter defines the scale of smoothing at the Gaussian scale-space. If the images already do not have high resolution, ie, the descriptive points are not distinctive already, as is the case with the images in our dataset, it might be a better idea to reduce it. And reduction here led to a slight rise in the accuracy.

For Dense SIFT:

Only the step size parameter is adjusted for dense-SIFT while keeping the other parameters for SIFT as their default.

*For **step size = 5, accuracy = 0.216**
*For **step size = 10, accuracy = 0.16**
*For **step size = 15, accuracy = 0.17**

The step size parameter defines the window size, resulting in a dense grid, from which the key points are to be detected. That is, with smaller step size values, denser grids are to be searched through, which should detect more key points with greater accuracy. The results seem convenient with that, ie, step size = 5 yielded a (more distinctively) greater accuracy. However, with smaller step sizes, the computational complexity also grows.

2 Bag of Features (45 pts)

- **How did you implement BoF? Briefly explain.**

  First, I loaded images in the training dataset to a dictionary where they had "key" as the file names and "value" as their SIFT descriptors. After the dictionary to store the training images' descriptors complete, I applied k-means clustering on the descriptors, where the cluster centers represent descriptive parts or objects in the images. Then, based on the cluster centers, I formed histograms for each training sample. For this, each descriptor in the training set is predicted by the kmeans model had been formed. For each image a histogram is created based on the predictions, and stored in the dictionary.

- **Give pseudo-code for obtaining the dictionary.**

```
allDescriptors = []
for each img in training set:
        keypoints, descriptor = SIFT(img)          // find descriptor for img
        allDescriptors.append(descriptor)


clusters = KMEANS(allDescriptors)


dictionary = dict()          //dictionary : <img> : <histogram>
for each img in training set:
        pred = clusters.predict(img.descriptor)          //predict the clusters for descriptors
        hist = histogram(pred, clusters)          //form histogram of descriptors based on cluster
        dictionary[img] = hist          // store historgrams for each image
```

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

```
// clusters is obtained in the training phase as shown above
keypoints, descriptors = SIFT(img)
preds = clusters.predict(descriptors)
hist = histogram(preds, clusters)
// hist is the BoF representation of image img
```

- **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.**

  Three different values for the number of clusters in kmeans have been tried; namely, k = 32, k = 128, k = 256 with dense SIFT with step size = 5, and the default parameters of SIFT. The k values have been chosen starting from k = 32 to keep the run feasible. Also, I have subsampled training instances to shorten the run time further, ie, 100 images from each class has been used to train the kmeans classifier. Hence, the given accuracy results below are for kmeans classifier trained with subsampled datasets.

for k = 32: accuracy = 0.186
for k = 128: accuracy = 0.189
for k = 256: accuracy = 0.192

Here, it seems greater k values yield slightly better accuracy. The dictionary formed over k-means has k clusters where each center is to represent an interest point or an object from the images. That is, the average number of feature descriptors per a class would be a feasible choice for k. However, greater k values, although yield better predictive performance, prone to cause overfitting. Take one extreme for example, where we have k = number of data points, ie, each point is its own cluster, we would have no error, this but would be obviously meaningless. One could use the elbow method to determine the optimal number of clusters.

After choosing the number of clusters, no subsampling is appleid. That is, classification results are based on classifiers trained with whole dataset.

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.**,

  The following shows the accuracy values obtained for k values for number of nearest neighbors of k-NN.

  for k = 16: accuracy = 0.225
  for k = 32: accuracy = 0.218
  for k = 64: accuracy = 0.2206

  Although there is no clear performance difference, k = 16 seems to outperform the others. Additionally, we do not see consistent rise or fall in the performance based on rise/fall in the number of nearest neighbors, for instance, k = 64 yielded a greater accuracy than k=32.
  In general too small k values are known to lead to overfitting and be susceptible to outliers, as only the closest point has en effect on the classification. On the other hand, large k values cause a bias towards the class with more annotations. However, here the dataset is balanced, so I would expect greater k values, although run much slower, to yield better accuracy. Here, it may be the case where the data points in the resulting dictionary formed over the dataset spread out so evenly in the space, the change in the number of neighbors does not have much impact on the result.

- **What is the accuracy values, and how do you evaluate it? Briefly explain.**

  Accuracy is the proportion of the correctly classified instances to all classifications. That is, it only considers the correctly-labeled data.
  accuracy = (number of correctly labeled data) / (number of total prediction)

- **Give confusion matrices for classification results of these combinations.**

  *** For k = 16 nearest neighbors**

$$\begin{bmatrix}
29 & 4 & 2 & 3 & 10 & 1 & 3 & 5 & 8 & 6 & 14 & 5 & 0 & 5 & 5 \\
4 & 34 & 4 & 4 & 8 & 5 & 4 & 4 & 6 & 10 & 5 & 6 & 0 & 3 & 3 \\
3 & 3 & 36 & 8 & 13 & 3 & 6 & 2 & 4 & 2 & 1 & 9 & 2 & 3 & 5 \\
4 & 7 & 6 & 17 & 8 & 5 & 12 & 9 & 5 & 4 & 3 & 9 & 6 & 3 & 2 \\
7 & 12 & 7 & 11 & 13 & 3 & 6 & 9 & 4 & 5 & 9 & 5 & 1 & 5 & 3 \\
5 & 5 & 5 & 1 & 4 & 38 & 4 & 2 & 5 & 8 & 2 & 2 & 5 & 9 & 5 \\
3 & 3 & 9 & 10 & 5 & 7 & 18 & 5 & 4 & 0 & 6 & 8 & 8 & 7 & 7 \\
8 & 10 & 4 & 7 & 9 & 3 & 5 & 15 & 4 & 4 & 8 & 8 & 4 & 7 & 4 \\
9 & 6 & 5 & 2 & 6 & 6 & 7 & 9 & 10 & 8 & 7 & 10 & 2 & 3 & 1 \\
5 & 9 & 1 & 1 & 10 & 8 & 2 & 6 & 12 & 16 & 8 & 6 & 2 & 2 & 12 \\
7 & 8 & 2 & 8 & 7 & 9 & 6 & 5 & 7 & 5 & 15 & 9 & 2 & 3 & 7 \\
4 & 9 & 12 & 7 & 8 & 5 & 1 & 6 & 7 & 5 & 5 & 18 & 3 & 3 & 7 \\
1 & 2 & 0 & 2 & 1 & 5 & 4 & 7 & 2 & 3 & 3 & 0 & 65 & 2 & 3 \\
2 & 2 & 1 & 2 & 2 & 9 & 5 & 3 & 5 & 4 & 7 & 7 & 6 & 30 & 15 \\
5 & 0 & 5 & 5 & 5 & 10 & 5 & 3 & 6 & 8 & 5 & 8 & 4 & 13 & 18
\end{bmatrix}$$

  *** For k = 32 nearest neighbors**

$$\begin{bmatrix}
28 & 13 & 4 & 3 & 6 & 4 & 1 & 3 & 2 & 4 & 11 & 9 & 1 & 7 & 4 \\
5 & 36 & 2 & 3 & 5 & 5 & 3 & 8 & 6 & 11 & 5 & 7 & 0 & 1 & 3 \\
4 & 3 & 30 & 11 & 12 & 4 & 8 & 7 & 2 & 0 & 3 & 10 & 1 & 2 & 3 \\
6 & 8 & 4 & 14 & 6 & 12 & 10 & 5 & 5 & 4 & 5 & 5 & 9 & 5 & 2 \\
11 & 15 & 8 & 5 & 17 & 3 & 3 & 8 & 4 & 11 & 4 & 5 & 2 & 2 & 2 \\
8 & 2 & 7 & 3 & 3 & 41 & 6 & 1 & 4 & 6 & 3 & 3 & 3 & 6 & 4 \\
5 & 3 & 10 & 11 & 6 & 5 & 14 & 4 & 10 & 2 & 4 & 8 & 6 & 5 & 7 \\
6 & 8 & 8 & 5 & 5 & 5 & 3 & 14 & 6 & 9 & 10 & 5 & 3 & 8 & 5 \\
3 & 4 & 6 & 2 & 7 & 10 & 3 & 5 & 15 & 8 & 11 & 9 & 1 & 7 & 9 \\
7 & 7 & 3 & 10 & 3 & 5 & 6 & 3 & 6 & 19 & 4 & 12 & 3 & 8 & 4 \\
12 & 10 & 5 & 3 & 9 & 9 & 3 & 8 & 5 & 6 & 10 & 6 & 2 & 5 & 7 \\
8 & 13 & 8 & 8 & 3 & 10 & 5 & 3 & 5 & 4 & 7 & 19 & 2 & 4 & 1 \\
5 & 0 & 0 & 6 & 3 & 6 & 6 & 1 & 2 & 1 & 2 & 2 & 60 & 5 & 1 \\
4 & 2 & 1 & 5 & 2 & 3 & 2 & 5 & 4 & 8 & 4 & 8 & 5 & 36 & 11 \\
6 & 3 & 5 & 6 & 3 & 6 & 6 & 10 & 10 & 7 & 6 & 7 & 3 & 12 & 10
\end{bmatrix}$$

*** For k = 64 nearest neighbors**

$$\begin{bmatrix}
28 & 10 & 3 & 1 & 11 & 0 & 2 & 6 & 6 & 4 & 12 & 6 & 1 & 5 & 5 \\
6 & 29 & 1 & 8 & 7 & 8 & 1 & 10 & 6 & 9 & 2 & 6 & 2 & 1 & 4 \\
5 & 3 & 39 & 6 & 6 & 3 & 7 & 7 & 3 & 1 & 4 & 12 & 0 & 3 & 1 \\
2 & 7 & 7 & 8 & 6 & 4 & 6 & 9 & 9 & 6 & 5 & 9 & 6 & 10 & 6 \\
12 & 15 & 6 & 5 & 16 & 3 & 8 & 8 & 8 & 5 & 7 & 3 & 0 & 2 & 2 \\
6 & 3 & 7 & 3 & 2 & 34 & 1 & 4 & 4 & 11 & 3 & 6 & 2 & 9 & 5 \\
3 & 6 & 10 & 6 & 7 & 5 & 19 & 9 & 6 & 6 & 5 & 8 & 6 & 2 & 2 \\
5 & 7 & 4 & 12 & 5 & 2 & 2 & 16 & 8 & 4 & 4 & 9 & 6 & 9 & 7 \\
10 & 5 & 8 & 6 & 4 & 3 & 0 & 6 & 14 & 9 & 8 & 3 & 0 & 8 & 16 \\
6 & 8 & 4 & 4 & 8 & 7 & 3 & 5 & 8 & 20 & 9 & 5 & 3 & 7 & 3 \\
10 & 6 & 5 & 8 & 7 & 5 & 4 & 2 & 10 & 8 & 16 & 8 & 2 & 5 & 4 \\
12 & 8 & 10 & 6 & 2 & 6 & 7 & 3 & 4 & 5 & 9 & 13 & 3 & 6 & 6 \\
0 & 4 & 0 & 3 & 2 & 3 & 8 & 3 & 4 & 2 & 1 & 4 & 60 & 5 & 1 \\
1 & 1 & 3 & 1 & 2 & 8 & 4 & 4 & 9 & 8 & 3 & 5 & 5 & 34 & 12 \\
6 & 6 & 3 & 2 & 6 & 11 & 8 & 1 & 10 & 3 & 8 & 4 & 2 & 16 & 14
\end{bmatrix}$$

4 Additional Comments and References

(if there any)