CENG 483

Introduction to Computer Vision

Spring 2018-2019

Take Home Exam 2

Object Recognition

Student Random ID: 15

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 (15 pts)

1.1 Difference of SIFT and Dense-SIFT

The difference is that by default, SIFT uses Lowe's algorithm for keypoints and descriptor detection. Therefore an image might have a lot of different descriptor or none at all. With Dense-SIFT, we sample keypoints with steps. So an image with no keypoints in normal SIFT might have arbitrary amount of keypoints and descriptor with Dense-SIFT. Note that these arbitrary descriptors are less meaningful compared to the Lowe method.

1.2 Quantitative Results and Conclusions

Here are some results with SIFT and Dense Sift with different configurations where cluster size for K-Means is 32 and the K for KNN is 4

1. SIFT : 16.25 %
2. Dense-SIFT with step size 5 : 26.66 %
3. Dense-SIFT with step size 3 : 29.73 %

Note that other hyperparameters are held constant and are

1. nfeatures = 0

2. nOctaveLayers = 3
3. contrastThreshold = 0.014
4. edgeThreshold = 10
5. sigma = 0.27

As seen above, acurracy with descriptors obtained via SIFT is much lower than the Dense-SIFT case. The reason for this is that with Dense-SIFT we can sample images with some regular step size, therefore we gain a spatial information which is crucial for the training data-test data pair we have. Note that as we decrease the step size, we increase the sampling rate, gain more information hence, usually higher accuracy rates. Note that for a single image, the SIFT gives much more useful descriptors compared to Dense-SIFT. Please note that due to vanilla SIFT's inability to find descriptors at times, Dense-SIFT method is assumed from here on out.

2 Bag of Features (35 pts)

2.1 Implementation

For BoF implementation, firstly the descriptors of the all images are obtained and put into an array. Then this array is clustered using SciKit KMeans library. After all the clusters are found, each sample point's cluster is found.(This is a compact operation thanks to sklearn's fit_predict function) Afterwards, the dictionary is created for each image in the training set, using the cluster centers as quantization levels. At this point, BoF implementation is complete and all that's left is repeat the same procedure for test images and compare the results.

2.2 Dictionary creation pseudocode

Algorithm 1 Creation of bag of features dictionary

procedure CREATE_DICTIONARY
Result: Array of cluster points which serve as the codebook i.e dictionary
unclustered_array $\leftarrow$ []
labels $\leftarrow$ []
foreach $img \in img\_samples$ **do**
    descriptor_array $\leftarrow sift(img)$
    unclustered_array.append(descriptor_array) labels.append(label_of_img)
end
return unclustered_array along with labels

2.3 Bag of features representation pseudocode

Algorithm 2 Creation of bag of features dictionary

procedure GET_BOF_REPRESENTATION(clustered_array,descriptor_array)
clustered_array is the dictionary
descriptor_array is the array of number of descriptors per image
Result: Array of histograms where each histogram represents an image in terms of features in BoF
if *Dense-SIFT* **then**
    Num_Descriptor_PerImg $\leftarrow get\_num\_descriptor\_perimg()$
    clustered_reshaped $\leftarrow reshape(clustered\_array, Num\_Descriptor\_PerImg)$
    histogram_arr $\leftarrow create\_histogram(clustered\_reshaped, bins)$
    return normalize(histogram)
else
    histogram_arr $\leftarrow$ []
    foreach $descriptor\_num \in descriptor\_array$ **do**
        sliced $\leftarrow slice\_array(clustered\_arr, descriptor\_num)$
        histogram $\leftarrow create\_histogram(sliced, bins)$
        histogram_arr.append(histogram)
        return histogram_arr
    end
end

2.4 Quantitative Results

Here are the results with respect to hyperparameters, where the KNN parameter is held constant at 1:

- KMeans clusters : 16,
  - Step size 5, accuracy : 23.73 %
  - Step size 3, accuracy : 23.40 %
- KMeans clusters : 32,
  - Step size 5, accuracy : 25.66 %
  - Step size 3, accuracy : 31.86 %
- KMeans clusters : 64,
  - Step size 5, accuracy : 28.93 %
  - Step size 3, accuracy : 30.93 %
- KMeans clusters : 128,
  - Step size 5, accuracy : 32.13 %

  - Step size 3, accuracy : 35.06 %

As seen above, as we increase the number of cluster centers in KMeans, we see a general increase in accuracy. This is because increasing the number of clusters corresponds to expanding the codebook. As we expand the codebook, we are more expressive in terms of describing images because we have more 'words' for describing possible distinct regions between images. As discussed above, as we decrease the step size, we have more descriptors which helps us identify distinct regions of each category of images.

3 Classification (20 pts)

3.1 Quantitative Results

Holding cluster size = 64 and step size = 3 constant:

- KNN with K = 1 , Accuracy : 30.93 %
- KNN with K = 2 , Accuracy : 29.13 %
- KNN with K = 4 , Accuracy : 31.53 %
- KNN with K = 16 , Accuracy : 34.93 %

Here are some combinations of KNN variables with various cluster sizes and step sizes:

- Cluster size : 128 , KNN : 4
  - Step size 3, Accuracy : 37.3 %
  - Step size 5, Accuracy : 34.3 %
- Cluster size : 128 , KNN : 2
  - Step size 3, Accuracy : 34.39 %
  - Step size 5, Accuracy : 29.93 %
- Cluster size : 128 , KNN : 1
  - Step size 3, Accuracy : 35.06 %
  - Step size 5, Accuracy : 32.13 %
- Cluster size : 32 , KNN : 16
  - Step size 3, Accuracy : 33.46 %
  - Step size 5, Accuracy : 30.86 %
- Cluster size : 32, KNN : 4
  - Step size 3, Accuracy : 29.73 %
  - Step size 5, Accuracy : 26.66 %

- Cluster size : 32, KNN : 2
  - Step size 3, Accuracy : 28.26 %
  - Step size 5, Accuracy : 26.20 %
- Cluster size : 32, KNN : 1
  - Step size 3, Accuracy : 31.86 %
  - Step size 5, Accuracy : 25.66 %

We can see that as we increase the K, we see an increase in accuracy, with the exception of K= 2. This is due to the fact that the algorithm to vote for the majority selects one of the choices randomly if they differ. With K = 1 , we just select the closest one we have. With K = 16, we have the highest accuracy because images with similar category tend to have similar histograms, where they pile up near the top, when we apply nearest neighbors algorithm. So having higher K in this case helps us determine what type of image we have in the grand scheme of things. One other thing to mention is that at cluster size 32, the KNN value of 1 is better than 2 and 4. What this tells us is that the amount of cluster sizes we have isn't sufficient enough since the clusters, or the words we have doesn't convey enough information about the distinct image properties. At cluster size 128, we see that this isn't the case. So it's an indicator that 128 is the codebook size we should be using for distinguishing.

3.2 Accuracy and Evaluation

Accuracy is simply the correctly classified samples divided by the whole sample size. In more technical terms, it's the ratio of the true positives and true negative over the sample size. To calculate accuracy, I've obtained a vector for the classification results, then I subtracted the correct labels from this matrix and counted the non zero entries. Note that this gives us 1-accuracy since we are counting the number of wrongly classified entries. For that reason I subtracted that 1(actually 100 since we are multiplying by 100 prior) to get the accuracy.

3.3 Confusion Matrices

Before presenting the confusion matrices, note that each label is given with an index between 0 and 14. Here are the mappings between indices and classification labels:

$0 \longrightarrow apple$
$1 \longrightarrow aquarium\_fish$
$2 \longrightarrow beetle$
$3 \longrightarrow camel$
$4 \longrightarrow crab$
$5 \longrightarrow cup$
$6 \longrightarrow elephant$
$7 \longrightarrow flatfish$
$8 \longrightarrow lion$
$9 \longrightarrow mushroom$
$10 \longrightarrow orange$
$11 \longrightarrow pear$

12 $\longrightarrow$ $road$
13 $\longrightarrow$ $skyscraper$
14 $\longrightarrow$ $woman$
Here are some confusion matrices with various configurations of the subset above :

- Cluster size : 32, KNN : 1, Step size 5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	25	8	3	7	7	5	2	6	8	5	14	7	0	0	3
1	5	27	3	4	11	3	2	6	9	11	8	3	0	0	8
2	2	2	40	6	8	7	6	3	4	2	2	10	2	1	5
3	2	2	13	12	9	2	13	8	8	8	5	5	6	4	3
4	3	13	6	7	15	4	7	8	5	9	12	4	1	2	4
5	3	1	6	3	6	41	5	5	2	5	5	8	1	2	7
6	4	2	6	10	6	4	26	8	7	2	9	5	1	8	2
7	5	9	3	11	5	5	5	17	5	9	11	7	2	2	4
8	4	6	10	3	9	3	7	7	18	9	5	5	0	3	11
9	6	15	0	7	6	6	2	6	14	13	3	11	4	3	4
10	12	11	4	5	8	1	6	7	10	9	16	2	1	3	5
11	10	3	5	7	9	5	3	5	4	8	11	18	1	7	4
12	0	1	0	8	4	0	2	2	2	3	3	3	69	3	0
13	3	0	0	4	2	9	11	5	5	3	1	8	3	34	12
14	4	3	11	5	7	6	6	5	8	8	8	5	3	7	14

- Cluster size: 32, KNN: 4, Step size 3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	54	15	5	1	3	2	1	1	3	3	7	2	1	0	2
1	9	47	3	4	10	2	3	2	4	14	0	0	1	1	0
2	5	7	52	7	13	4	5	2	1	0	3	0	1	0	0
3	10	12	6	16	10	5	14	2	2	5	3	2	6	4	3
4	9	22	22	6	13	4	2	0	4	9	6	0	2	0	1
5	7	6	6	3	3	51	4	1	0	4	2	2	4	1	6
6	6	4	10	12	9	2	31	7	4	3	3	0	0	3	6
7	18	16	12	11	4	6	5	11	2	4	3	1	2	2	3
8	10	11	9	12	14	1	6	5	13	6	2	5	0	2	4
9	13	18	6	3	9	4	2	3	7	23	4	2	2	0	4
10	14	11	3	5	11	3	6	6	10	8	14	3	0	2	4
11	18	15	12	6	9	1	5	0	3	2	7	16	0	1	5
12	4	6	3	4	2	5	5	2	0	1	1	0	64	2	1
13	4	5	4	7	1	7	10	5	8	3	3	3	5	27	8
14	18	10	5	9	4	6	8	4	4	3	4	4	1	6	14

- Cluster size: 128, KNN: 4, Step size 3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	55	13	5	5	1	2	4	2	0	4	7	1	0	0	1
1	10	50	3	5	5	0	2	2	5	12	4	0	1	0	1
2	6	4	55	7	8	2	3	3	2	3	3	1	1	0	2
3	4	11	10	25	12	9	6	0	4	5	3	2	4	2	3
4	7	18	16	6	29	3	2	4	2	7	4	1	0	0	1
5	5	5	6	6	2	62	1	0	3	2	2	2	0	1	3
6	8	5	7	16	8	1	40	5	0	3	3	1	0	0	3
7	18	11	3	7	11	3	3	19	1	10	4	3	0	3	4
8	2	9	13	15	7	6	4	7	19	9	2	3	0	2	2
9	6	17	1	6	7	2	4	4	5	35	6	2	0	2	3
10	16	11	10	4	11	2	5	2	4	5	26	2	0	0	2
11	16	9	7	8	5	2	3	4	6	3	10	21	0	1	5
12	1	4	2	4	4	3	2	4	0	2	2	0	67	3	2
13	5	5	3	4	6	5	7	2	5	1	2	5	1	41	8
14	15	10	7	10	3	3	4	4	8	4	7	5	0	4	16

For the discussion of confusion matrices, please refer to the 'Best Configuration' section.

4 Your Best Configuration (30 pts)

4.1 Best Accuracy

With the conclusions and inferences above, it's natural to assume we'd be having the highest accuracy with these hyperparameters:

- Cluster size for K-means : 128
- Step size of 3
- Nearest neighbors : 16

Alas, doing an exhaustive search between different combinations yields the best accuracy with the configuration given above. The highest accuracy rate reached with this configuration is 39.0 % .

That is, with K-means value of 128, we have expressibility, with step size of 3, we have enough samples for descriptors to determine distinct features, with nearest neighbors of 16, we look at the overall picture.

4.2 Setup

To obtain the best accuracy, simply run the following command:

python3 hw.py best-configuration

After waiting for a while(maybe 2 hours) you can see the accuracy along with confusion matrix.

Also, to run the test configuration you can run:

python 3 hw.py test

Note that opencv-python 3.4.2.17 and opencv-contrib-python 3.4.2.17 versions must be used for the project. Moreover, best-configuration might take couple of hours since the MiniBatch version of KMeans

haven't been used. However MiniBatchKMeans can be used instead of normal KMeans as it's only a change in one line of code.(You can change Line 13 to MiniBatchKMeans with appropiate sample size at the cost of some accuracy.)

For the other information, please refer to the source code.

4.3 Confusion Matrix and Interpretations

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	47	13	2	2	4	1	4	2	3	8	11	2	0	1	0
1	1	53	3	2	8	0	4	0	4	15	4	3	2	0	1
2	5	1	62	5	6	1	2	5	2	4	2	1	0	2	2
3	2	11	9	25	4	3	14	4	7	9	1	3	4	3	1
4	8	19	19	5	22	2	2	3	3	9	3	0	2	0	3
5	5	5	7	3	1	60	2	0	2	2	4	1	2	1	5
6	0	1	8	14	6	0	53	4	2	3	5	1	0	1	2
7	9	9	7	7	7	7	4	14	11	7	6	6	1	2	3
8	4	7	6	9	7	3	4	5	24	11	7	7	0	2	4
9	3	10	2	2	8	5	5	2	5	45	9	1	1	1	1
10	17	8	8	3	9	1	7	1	9	4	31	1	0	1	0
11	13	12	9	7	2	3	4	1	11	6	13	16	0	0	3
12	4	1	2	2	2	5	3	3	0	4	0	0	71	3	0
13	4	3	1	2	0	4	8	6	8	2	5	9	1	45	2
14	5	8	2	3	7	11	2	8	12	7	10	3	0	5	17

Here in the confusion matrix, each row represents a class(as indicated by its label) and each column is the predicted value corresponding to that class. Ideally, we would want this matrix to be as diagonal as possible. That's why in the first confusion matrix the dioganality was really low however we see a huge improvement with these configurations. Moreover, we can see that the category with index 12, i.e road has the highest accuracy with 71% whereas the category with index 7, i.e flatfish has the lowest accuracy with 14%. Also we can see that our model classified category 8, lion with flatfish similarly. An obvious observation is that each row sums up to 100, which is the number of images per category.

5 Additional Comments and References

One thing to observe is that in all the experiments made, we can see that accuracy drops down between KNN values of 1 and 2. The reason is that with 2 nearest neighbor cases, we might have ties which we might be forced to pick some value arbitrarily. If we use an odd value, like 3 instead of 2 for the value of KNN, we expect that we can see a increase in the accuracy. For example using the configurations of 3.1 except with KNN = 3, the accuracy has risen to 30%.