ECS797P COURSEWORK ASSIGNMENT 1: IMAGE CLASSIFICATION USING BOW

Animesh Devendra Chourey – 210765551

Queen Mary University of London – March 2022

2. Dictionary Creation – feature quantization

1. To load the pre-computed features for the training and test images the following command was executed

```
load('data/global/all_features');
```

   TrainMat (270000x128) and TestMat (90000x128) are the two variables that are loaded with the sift features having 128 dimensions.

2. A dictionary of 500 words(i.e. codewords) is created. The parameter *DictionarySize* controls the size of the dictionary. This dictionary is stored in the variable C(500x128). The *DictionarySize* determines the numbers of cluster centroids for the dataset and then with these created clusters the features are assigned to their corresponding clusters.

3. In this section, the euclidean distance between the image descriptors and the codewords(i.e. cluster centers) is calculated with the help of EuclideanDistance() function. This euclidean distance is used to cluster features into visual code words.

4. Here, each feature in the training and the test set is assigned the nearest codeword in the dictionary using EulideanDistance function. The index of the closest dictionary word is assigned. As a result *index_train*(1x270000) and *index_train*(1x90000) are created, which contains the indices of the assigned codewords.

```
% Initializing the vectors
index_train = zeros(size(TrainMat,1), 1);
index_test = zeros(size(TestMat,1), 1);


% Assign descriptor for every training image
for i = 1: size(TrainMat,1)
    % Find the nearest codeword using the min function
    % Euclidean distance between all the elements of dictionary and the descriptor
    [minv,index] = min(EuclideanDistance(TrainMat(i,:),C));
    index_train(i,1) = index;
end

% Assign descriptor for every test image
for i = 1: size(TestMat,1)
    % Find the nearest codeword using the min function
    % Euclidean distance between all the elements of dictionary and the descriptor
    [minv,index] = min(EuclideanDistance(TestMat(i,:),C));
    index_test(i,1) = index;
end
```

3. Image representation using Bag of Words representation

Each row in the row of matrix d represents the euclidean distance between the corresponding feature and the codewords in the dictionary. Afterwards from the minimum value from each of the row, the corresponding index is incremented into the histogram. Then the histogram has been normalized using the *do_normalize()* function. The histogram is normalized to represent the image using bag of words.The BoW matrix stores both the train and the test dataset of which 1-300 are of training and 301-400 are test dataset. Therefore, the Bow matrix is of size 400x500.

```
d = EuclideanDistance(features.data, C);
histogram = zeros(1,vocbsize); %(1,500)
for i =1:size(d,1) %900
    [minv,index] = min(d(i,:)); %Get index with the minimum value
    %Increment the histogram value for the corresponding index
    histogram(1,index) = histogram(1,index) + 1;
end
BoW(ii,:) = do_normalize(histogram);
```

4. Image Classification using a Nearest Neighbour (NN) classifier

1. Here the L2 distance (i.e. euclidean distance) is calculated between the test and the training images and each test image is assigned or classified to its nearest neighbour class form the training set. To perform this operation *knnsearch.m* is used.

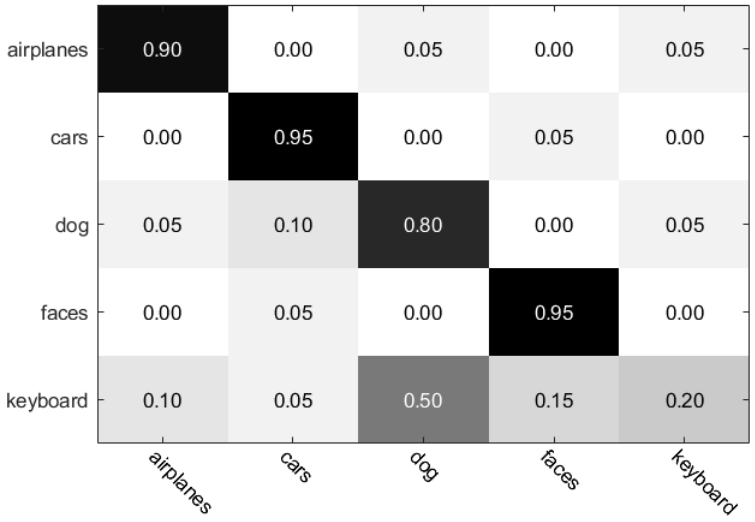
2. (a) The overall classification error :

   24%

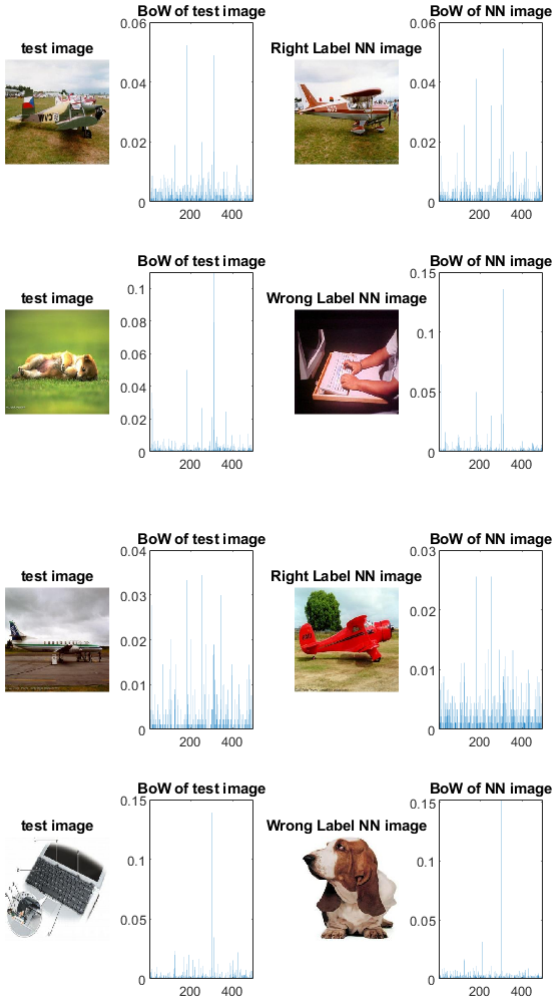
   (b) The classification error per class :

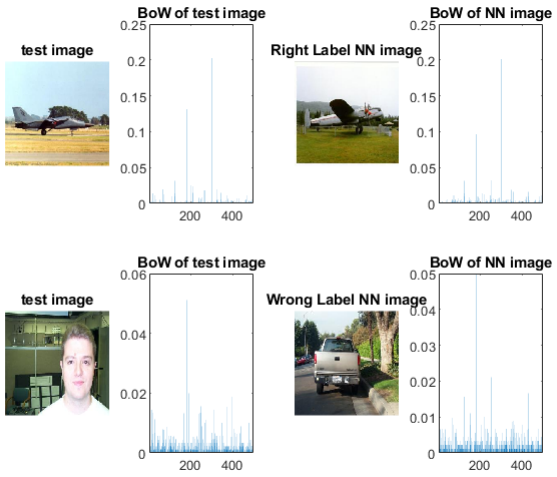
Errors per class	
Class	**Error**
Airplanes	10%
Cars	5%
Dog	20%
Faces	5%
Keyboard	80%

3. The confusion matrix is as follow :



4. Some images that are correctly classified and some images that are incorrectly classified :





One of the reason that the images have been miss-classified is that the wrongly classified images may have been affected because of the illumination and the background. Another reason could be that the margin defined by the classifier is not optimum one or it could be that these images are the outliers to the margin defined by the classifier. One more reason could be that in some of the miss-classified images, the structure could be extremely similar to the miss-classified images which results in similar BoW histogram.

5. Computing the histogram intersection between two histograms :

```
d = 1;
for index = 1:p
    %Get the smaller value amongst a and b
    d = d - min(a(1,index), b(1,index));
end
```

In this section method=2 (i.e. Histogram intersection) is used

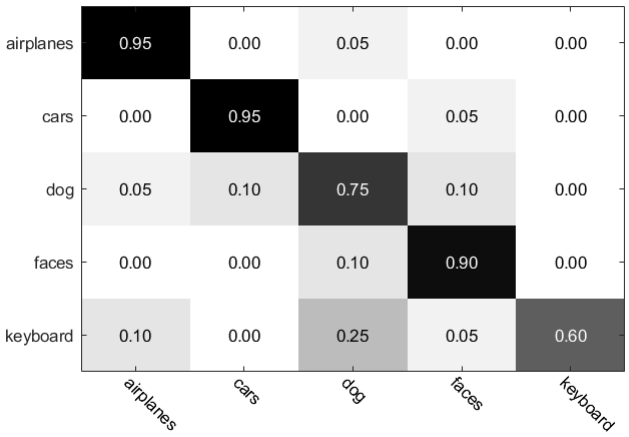
(a) The overall Classification error : **17%**

Histogram intersection can be used to significantly improve the generation of visual codebooks. Histogram Intersection is used as the similarity measure in clustering feature descriptors that are histograms, the generated visual codebooks produce better code words and as a consequence, improves Bag of Words classifiers.

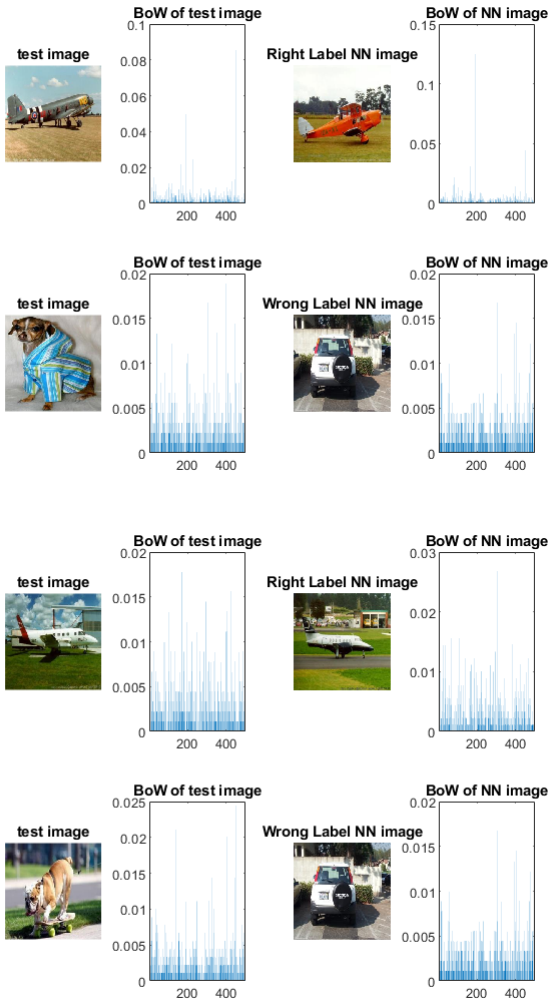
(b) The classification error per class :

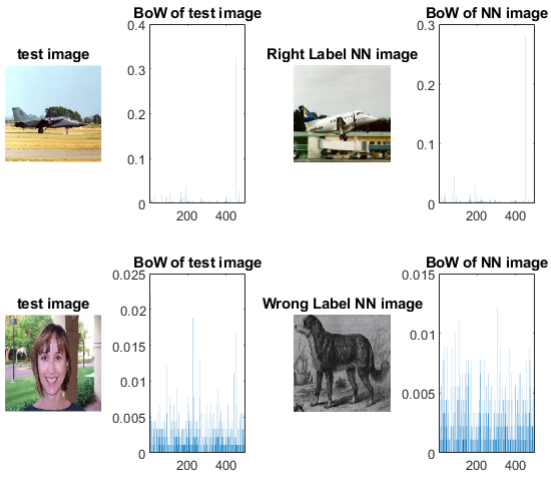
Errors per class	
Class	**Error**
Airplanes	5%
Cars	5 %
Dog	25%
Faces	10%
Keyboard	40%

(c) The confusion matrix is as follow :



(d) Some images that are correctly classified and some images that are incorrectly classified:





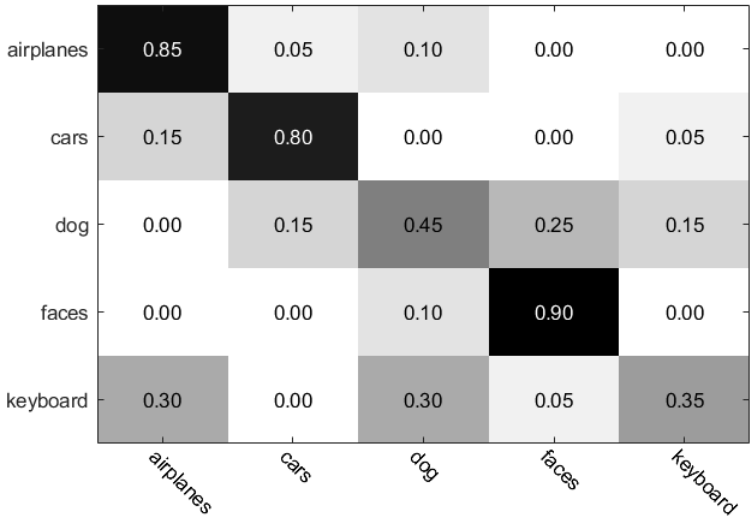
Changing the method to histogram intersection to compute the similarity of histograms results in decrease in overall error rate from 24% to 17%. Also per class classification accuracy also increases in some cases.

5. Dictionary Size

1. After converting the dictionary size to 20 the BoW matrix size now becomes 400x20.
   (a) **Method 1 (L2 Distance):**
      i. The overall Classification error : **33%**
      ii. The classification error per class :

Errors per class	
Class	**Error**
Airplanes	15%
Cars	20%
Dog	55%
Faces	10%
Keyboard	65%

iii. The confusion matrix is as follow :



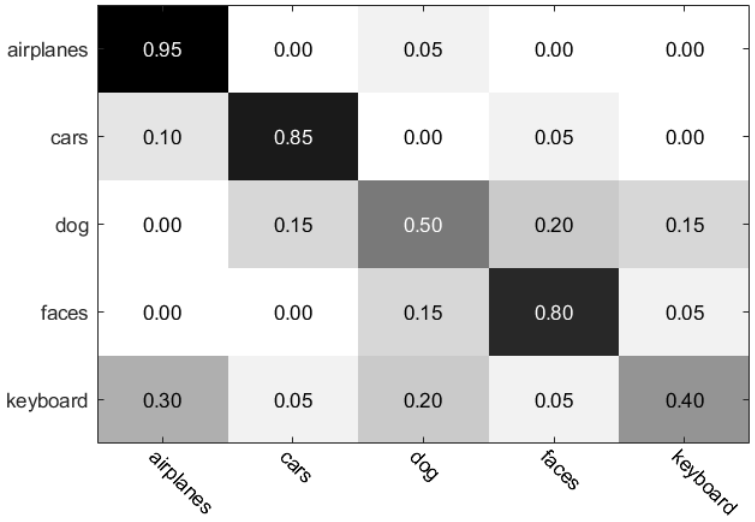
(b) **Method 2 (Histogram Inspection):**

i. The overall Classification error : **30%**

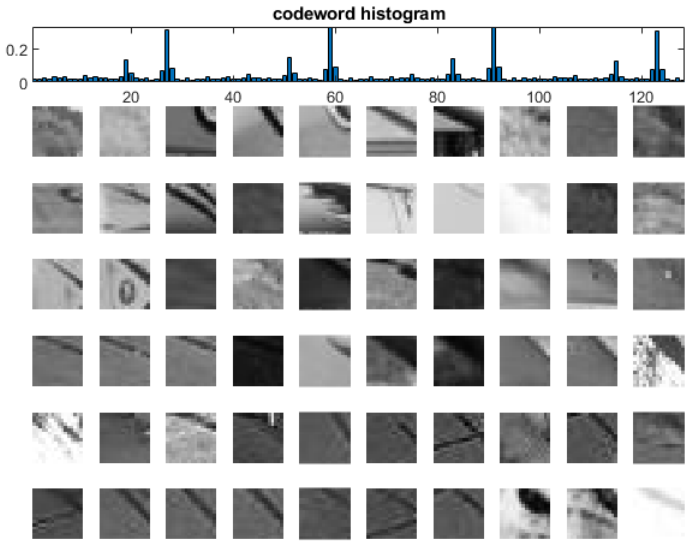
ii. The classification error per class :

Errors per class	
Class	**Error**
Airplanes	5%
Cars	15%
Dog	50%
Faces	20%
Keyboard	60%

iii. The confusion matrix is as follow :



2. There has been a drop in the performance after changing the cluster size to 20. This results in feature descriptors not being similar as it can be seen from the figure below. Since the dictionary is too small, it is not able to comprehend the training and the test data efficiently and thus this results in underfitting. Several feature descriptors are relatively similar to the corresponding codeword i.e. the distance between the codeword and the feature is smaller than the distance between the features and the other codewords. The smaller the distance, the more similarity between the two histograms and thus our performance drops. From the figure we can see that there are multiple repetitive patches in the dictionary.



6. Image classification using a Support Vector Machines (SVM) classifier

1. In this section optimal values for the parameter C are calculated using cross validation . Given a set of histograms, SVM classifier seeks a subset of the divided region that retains a large portion of the histograms. It takes into consideration the shape and density of the histogram distribution. It seeks to include most of the histograms in a compact hypersphere in the feature space, while paying less attention to those borderline cases. The output we get is as follow:

```
Cross Validation Accuracy = 76%
10 1.5 76 (best c=32, g=2.14355, rate=78)
Elapsed time is 10.152270 seconds.
```

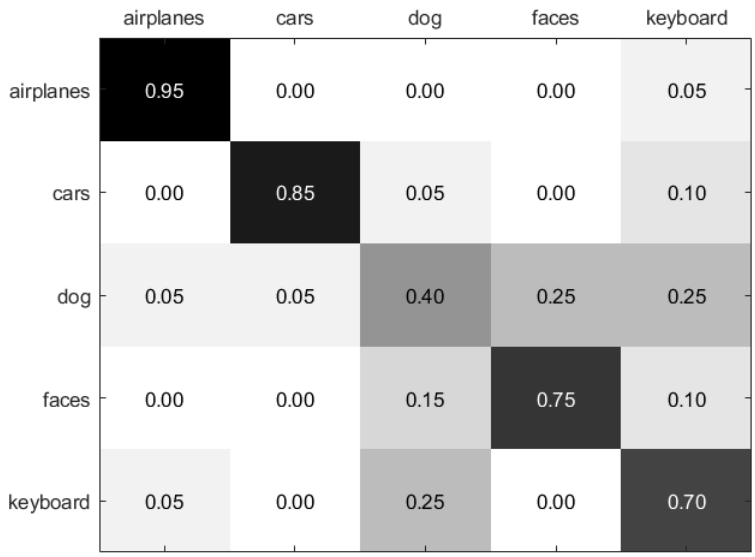
2. After applying the linear SVM classifier to the test images we get the accuracy of **73%**.

3. (a) The overall Classification error : **27%**

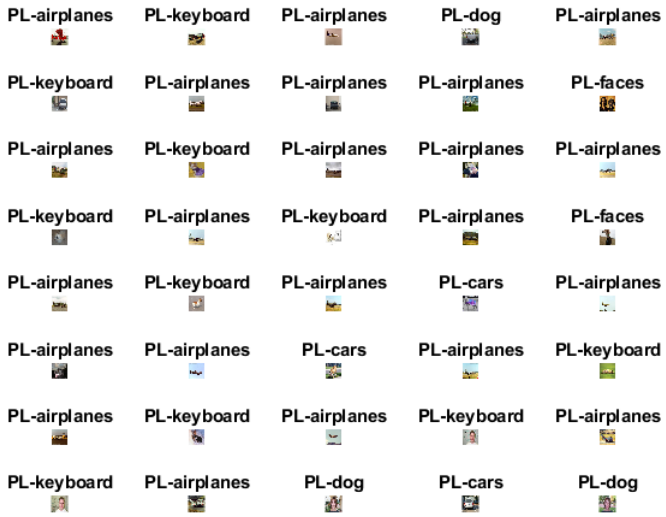
   (b) The classification error per class :

Errors per class	
Class	**Error**
Airplanes	5%
Cars	15%
Dog	60%
Faces	25%
Keyboard	30%

4. The most prominent change in accuracy we can see is in terms of classifying the keyboard class. SVM has performed extremely well when compared to the previous cases of KNN. It has gone up to 65% with SVM. The confusion matrix is as follow:



5. Some images that are correctly classified and some images that are incorrectly classified are shown in the following figure.



One of the reason that SVM classifies poorly could be because of the margin established by it. The margin defined might not be maximum margin for the dataset, as the one with maximum margin is better classifier. Because if this the data points might not be well separated. Since some images are quite similar to each other it gets extremely difficult for the classifier to define the margin.