School of Electronic Engineering and Computer Science

ECS797 Machine Learning for Visual Data Analysis

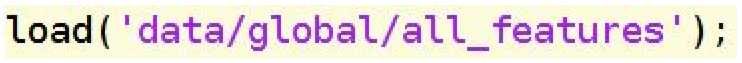
Lab 1: Image Classification using a Bag-of-Words model

Name: Yogen Parekh

Student ID: 190306340

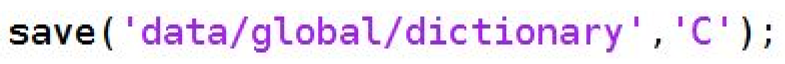
2. Dictionary creation – feature quantization

1. The given command was executed and the pre-computed features for the training and test images were loaded. The variable TrainMat(270000 X 128) and TestMat (90000X128) are loaded and the sift features have 128 dimensions. The executed code was.



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

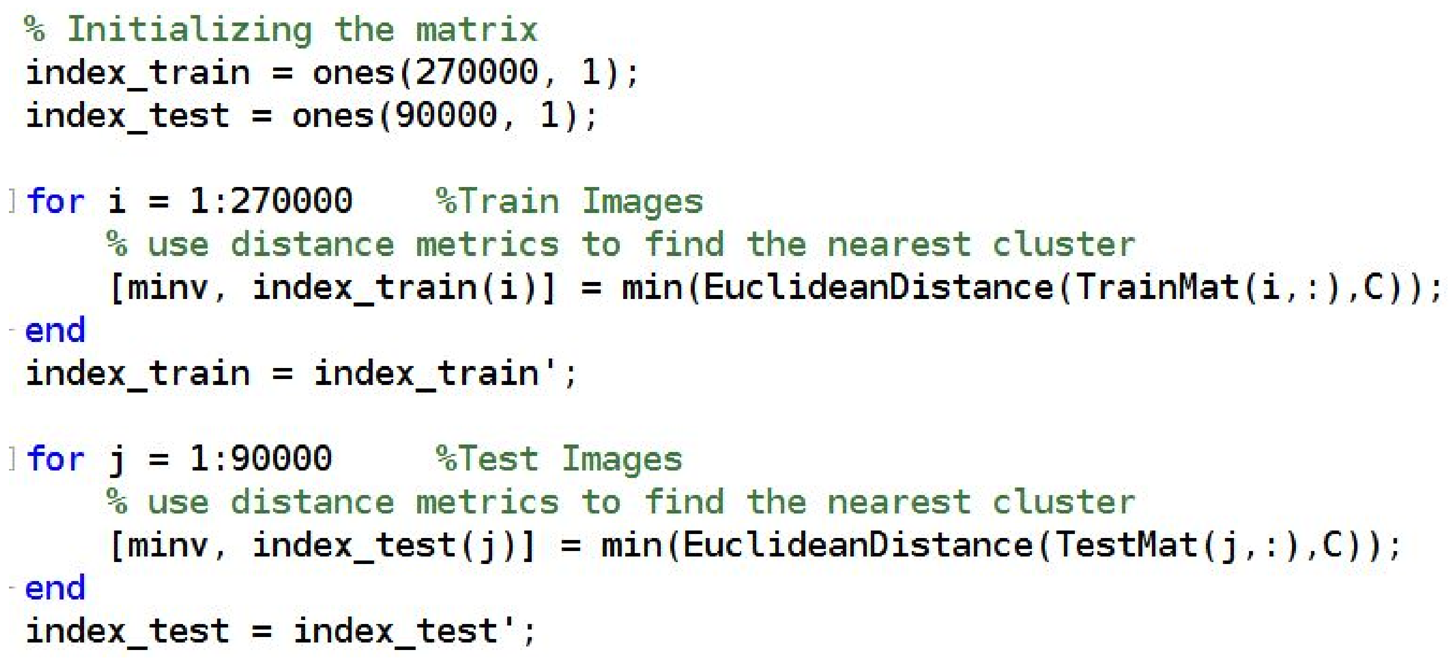
2. The given code in 2.2 was executed and a dictionary of 500 words has been created by using the k-means. The size of the dictionary is determined by the number of k-means clusters. After execution of the code, the dictionary is stored in the variable C, a 500X128 matrix which gets saved in dictionary.mat



```
save('data/global/dictionary','C');
```

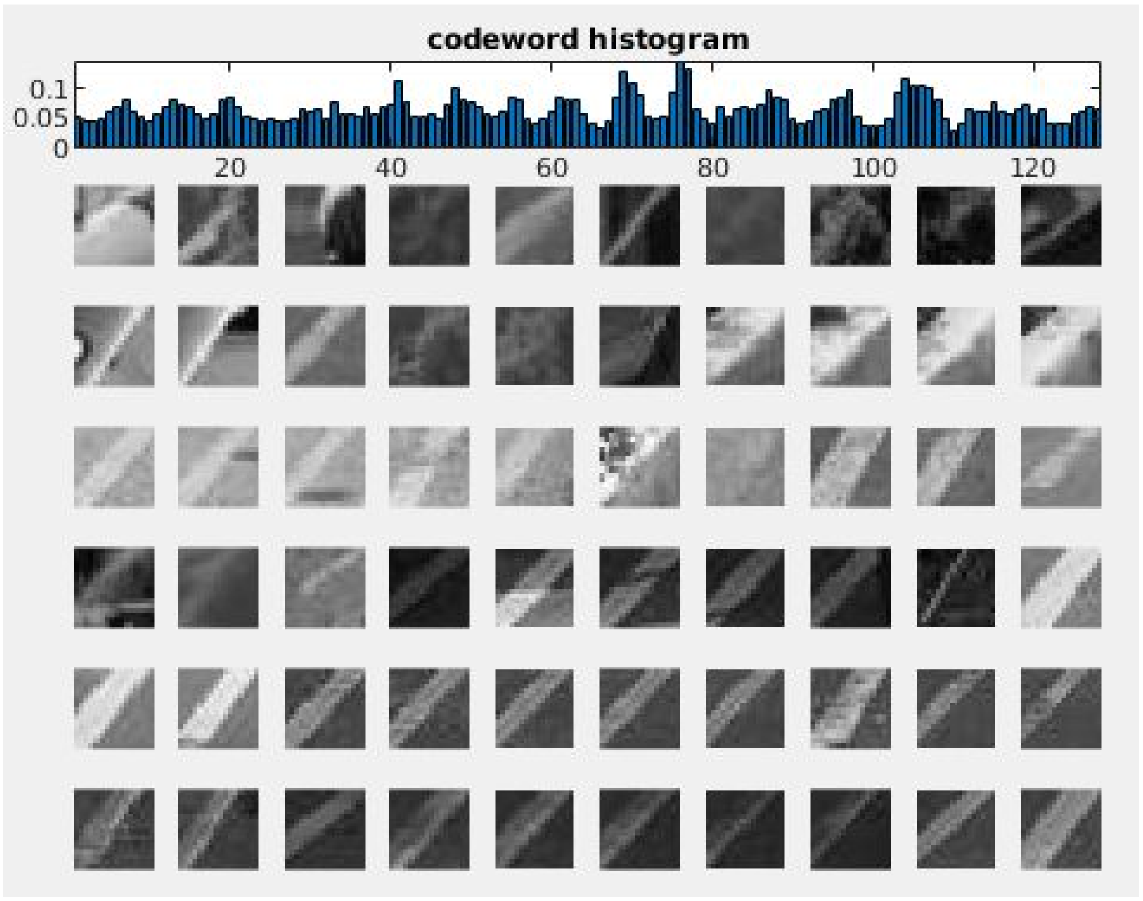
3. The code given in section 3.3 calculates the Euclidean distance between image descriptors and the codewords using the function defined in EuclideanDistance.mat

4. Given below is the code I designed to assign each descriptor in the training and test set to the nearest codeword cluster, using function min(). This creates matrices index_train and index_test which are then transposed to get the required index_train(1X270000) and index_test(1X90000). These contain indices of the assigned clusters.



```
% Initializing the matrix
index_train = ones(270000, 1);
index_test = ones(90000, 1);

for i = 1:270000     %Train Images
    % use distance metrics to find the nearest cluster
    [minv, index_train(i)] = min(EuclideanDistance(TrainMat(i,:),C));
end
index_train = index_train';

for j = 1:90000      %Test Images
    % use distance metrics to find the nearest cluster
    [minv, index_test(j)] = min(EuclideanDistance(TestMat(j,:),C));
end
index_test = index_test';
```

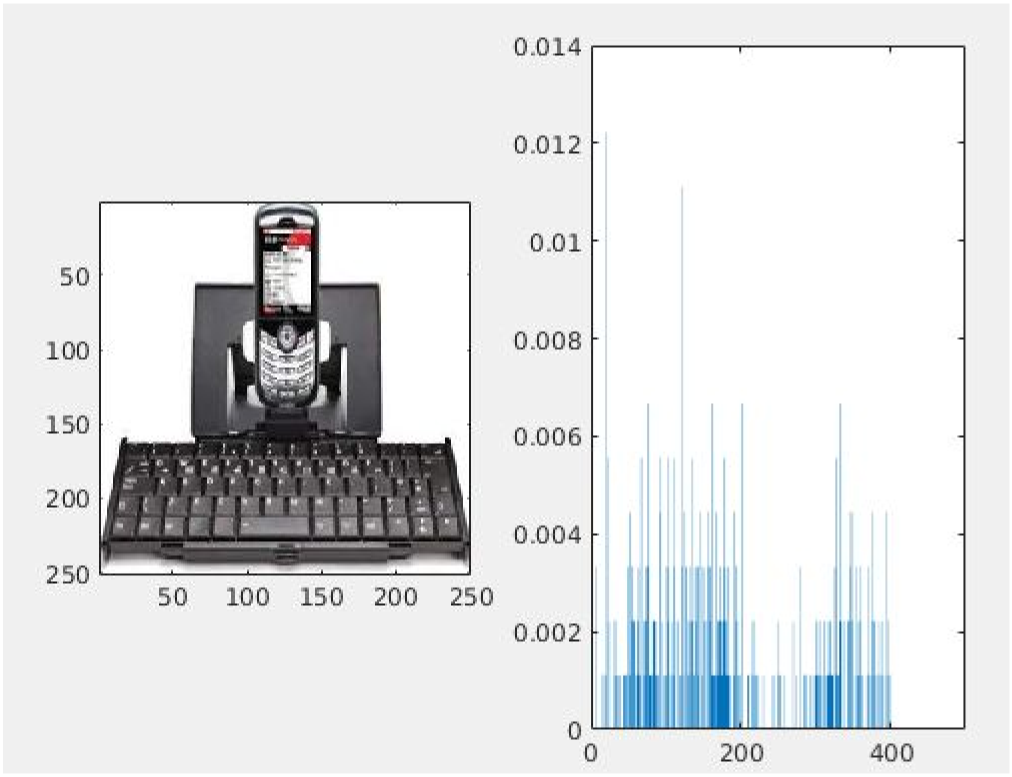
5. The image patches that were assigned to the same codeword cluster in the previous section have been visualised using the given code and function.



3. Image representation using Bag of Words representation

1. Each image is represented as histogram of the bag of words using the L1 norm. The function do_normalise() has been used for normalisation. BoW stores the images from the training and test dataset where indices 1 to 300 represent training images and 301 to 400 represent test images. Hence, Bow is a 400X500 matrix. Following code was used to calculate and implement Bow.

```
% calculate min distance bw features and codeword
d = EuclideanDistance(features.data, C);
row = size(d, 1); % Initialize
clusters = ones(1, vocbsize); % Initialize
for i = 1:row
    [minv, index] = min(d(i,:)); % find index for min value
    clusters(1, index) = clusters(1, index) + 1; % assign and move to next
end
BoW(ii, :) = do_normalize(clusters); % reshape 900 to 500
```



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

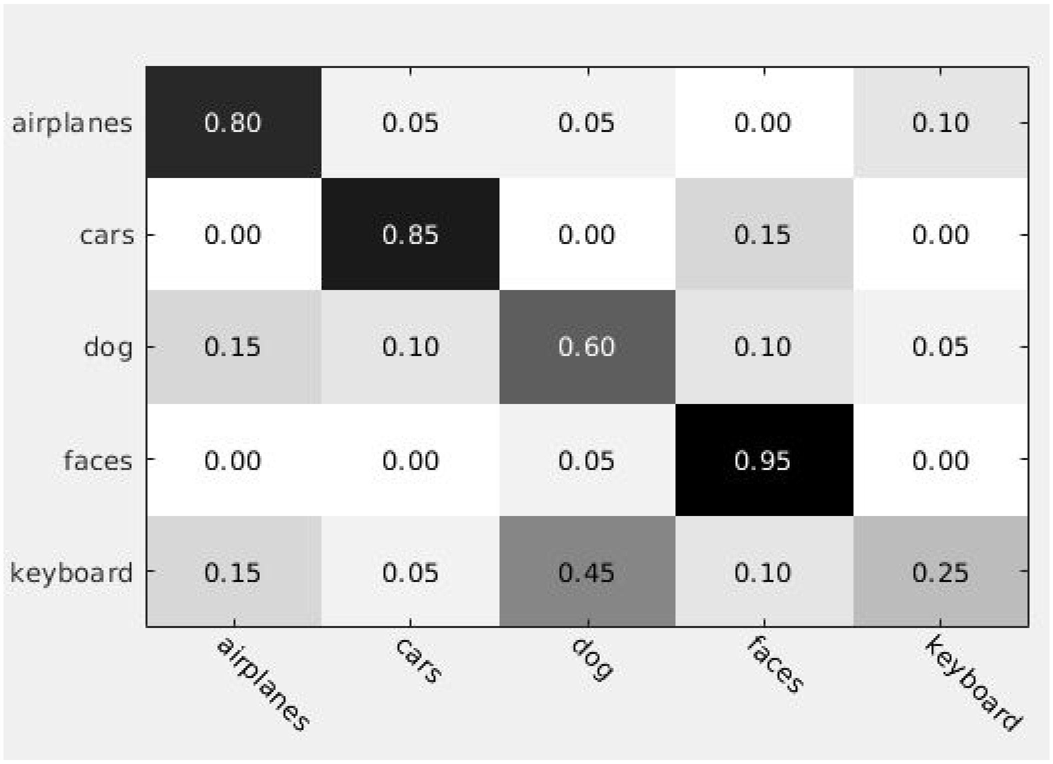
1. The steps mentioned for k-NN have been executed accordingly and the required output is generated. Euclidean distances(L2) have been calculated between the test and training images and each test image has been assigned to the class of its nearest neighbor from the training set. The code uses the function developed in section 2.3 and also uses the function in file knnsearch.m

2. Section 4.2 of the code is executed which gives the classification errors for each class and also the overall error for method 1.

Per Class Errors	
Airplane	0.2
Car	0.15
Dog	0.4
Faces	0.05
Keyboards	0.75

Overall Error
0.31

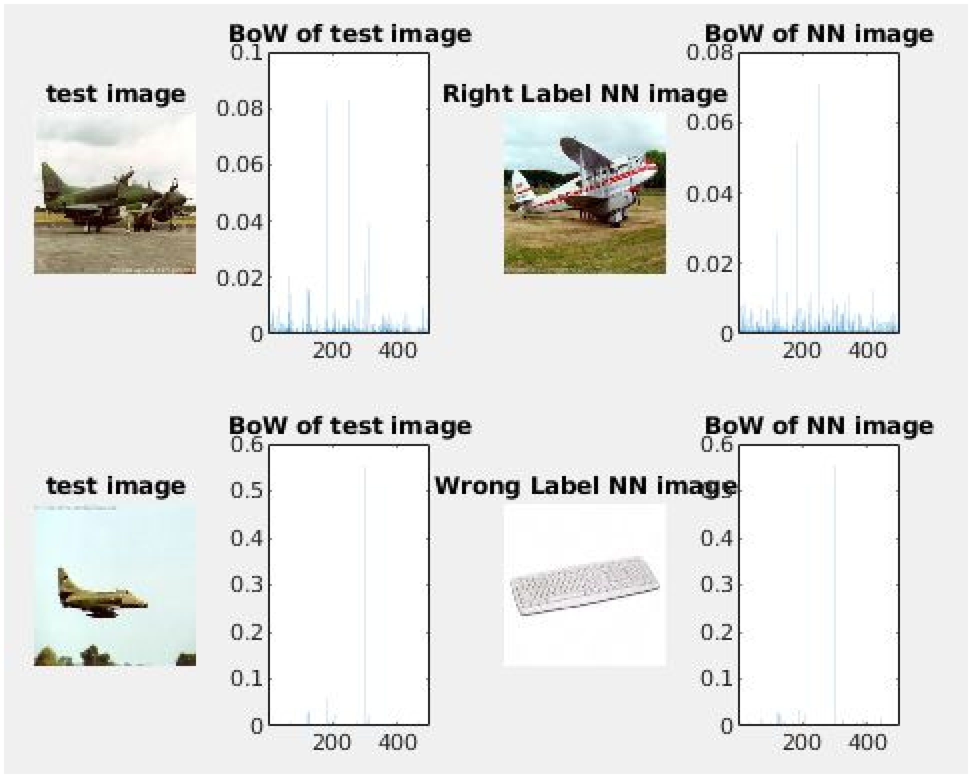
3. The resulting confusion matrix is shown below.



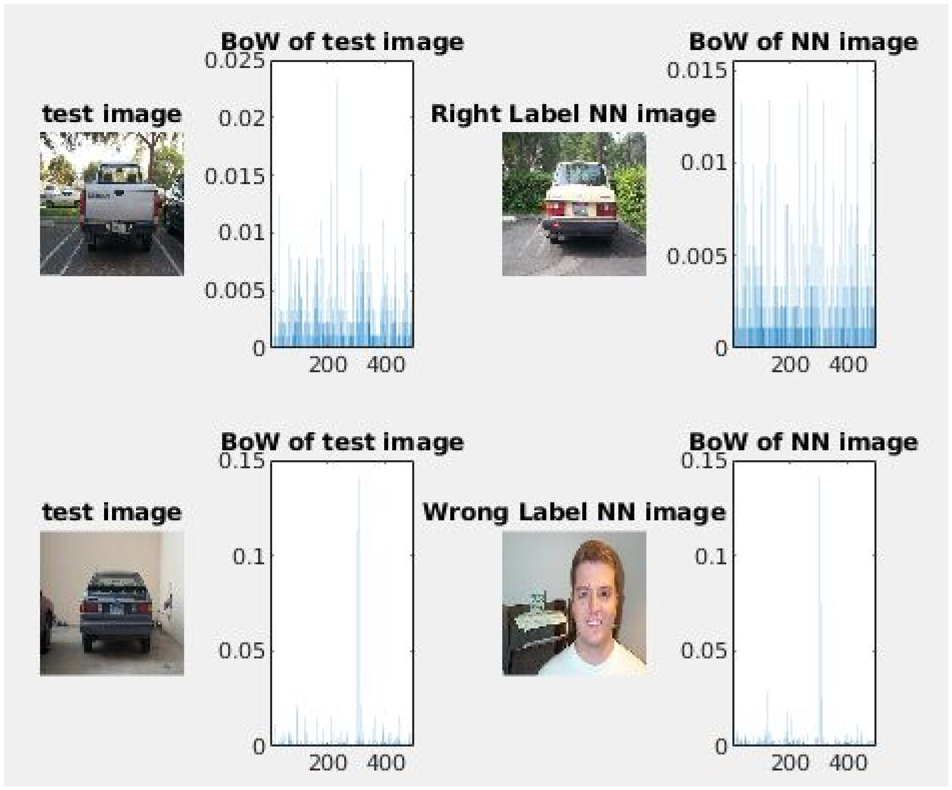
	airplanes	cars	dog	faces	keyboard
airplanes	0.80	0.05	0.05	0.00	0.10
cars	0.00	0.85	0.00	0.15	0.00
dog	0.15	0.10	0.60	0.10	0.05
faces	0.00	0.00	0.05	0.95	0.00
keyboard	0.15	0.05	0.45	0.10	0.25

4. For each class image correctly classified and incorrectly classified is shown below.

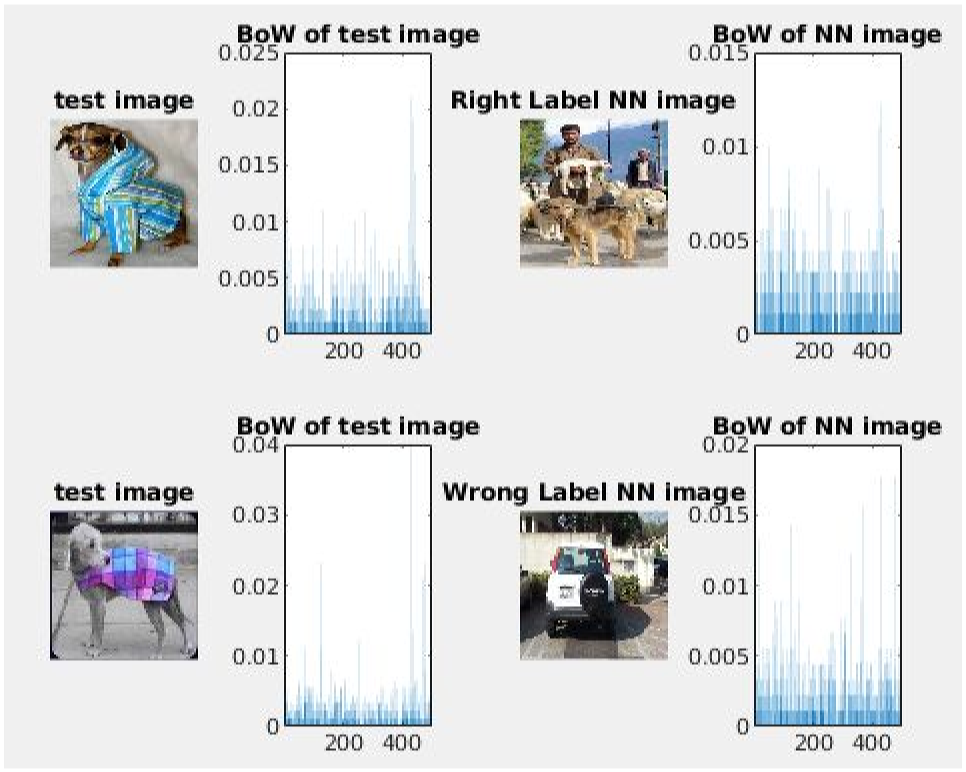
Airplane:



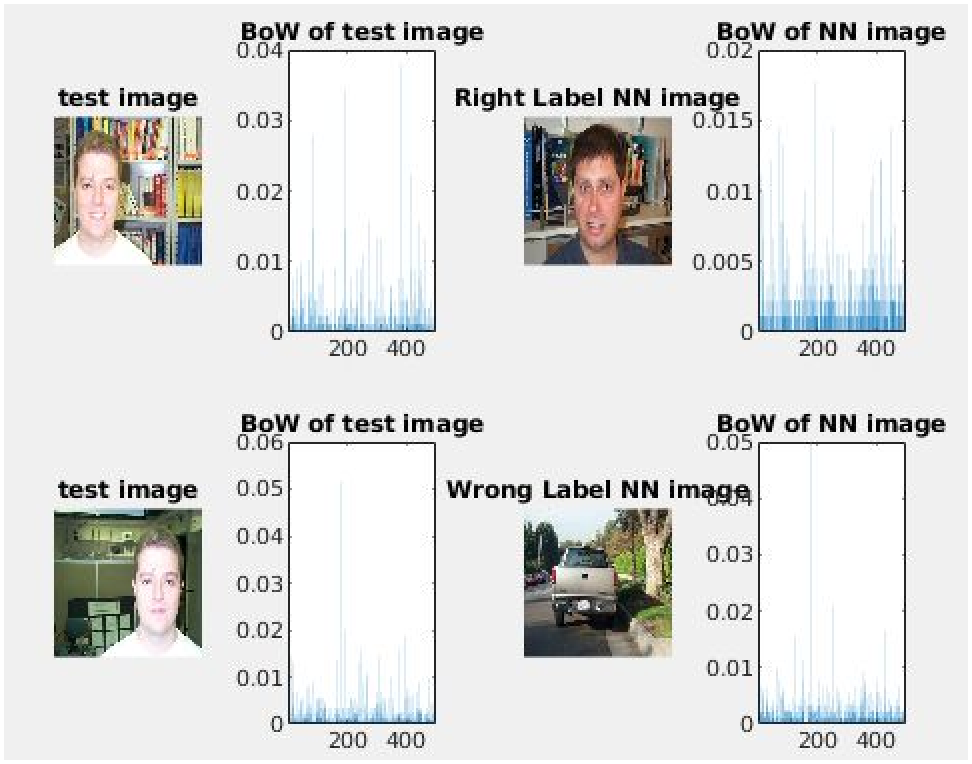
Cars:



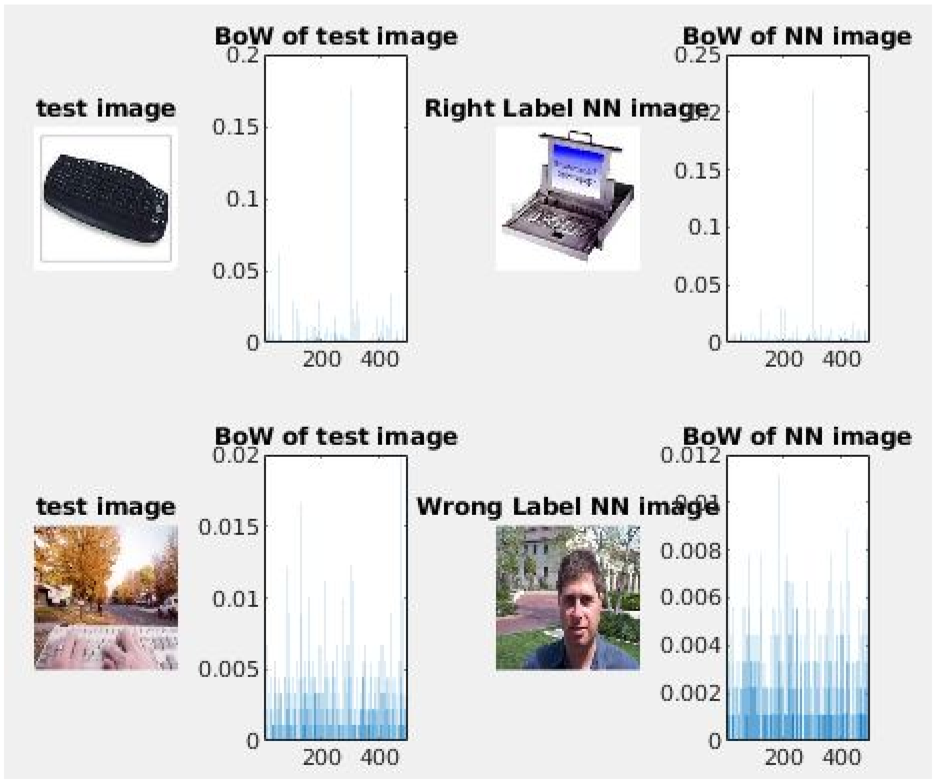
Dogs:


BoW of test image
BoW of NN image
test image
Right Label NN image
BoW of test image
BoW of NN image
test image
Wrong Label NN image

Faces:


BoW of test image
BoW of NN image
test image
Right Label NN image
BoW of test image
BoW of NN image
test image
Wrong Label NN image

Keyboards:



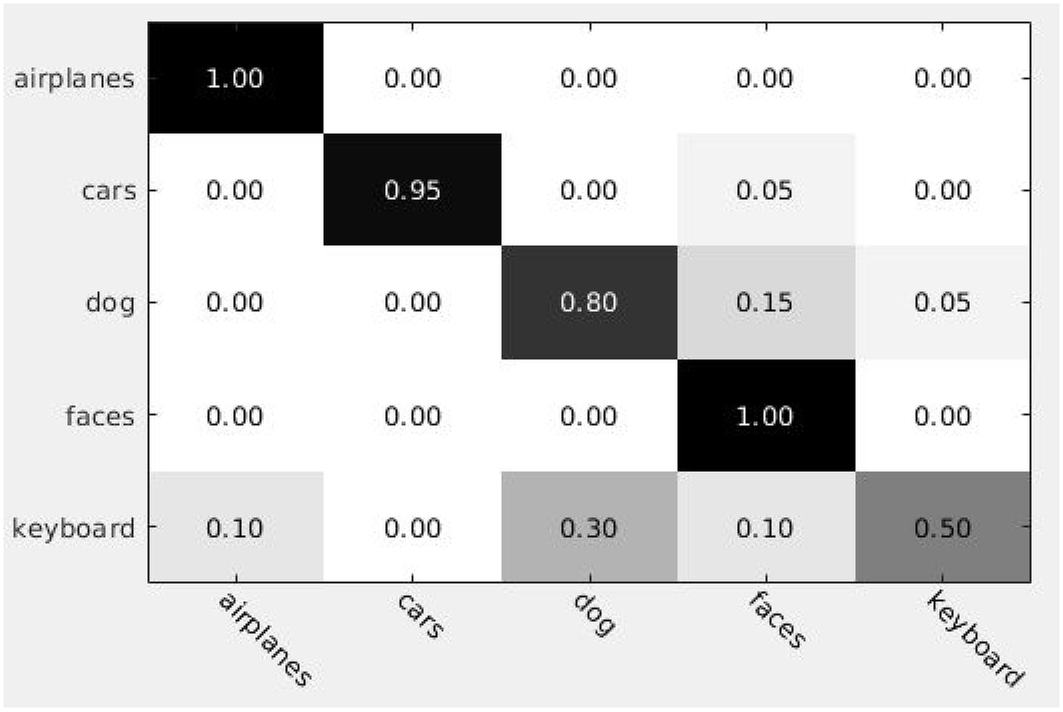
One of the reasons for the misclassification is the limitation of the BoW model to not detect the spatial relationships between the objects, i.e. it cannot specify how an object is located in space in relation to given reference object. In some of the misclassified images, the structure or the depth of the image is very similar to the misclassified image which results in a similar kind of BoW histogram.

5. The same process has been done again with the second method (method=2) which is histogram intersection using the following code. The corresponding results are displayed.

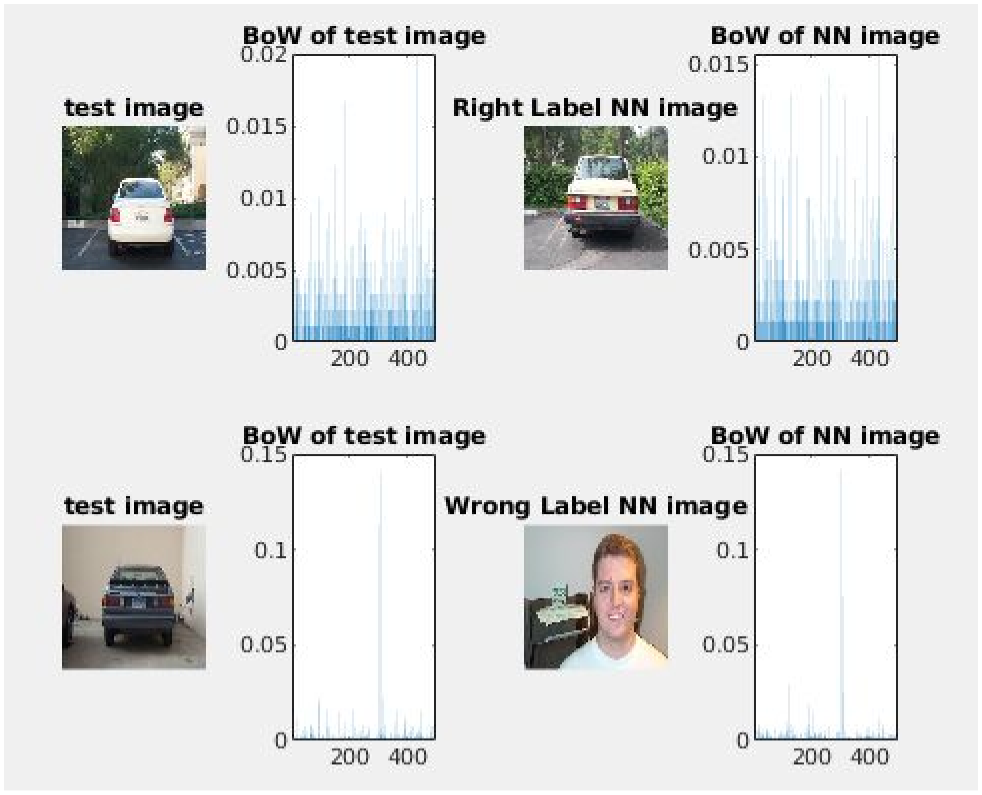
```
function d=histogram_intersection(a,b)

  p=size(a,2); % dimension of samples

  assert(p == size(b,2)); % equal dimensions
  assert(size(a,1) == 1); % a needs to be a single sample
  assert(size(b,1) == 1); % b needs to be a single sample

  %d=zeros(m,1); % initialize output array

  d = 0;
  %-----------------------------------------
  for index = 1:p
      d = d - min(a(1, index), b(1, index));
  end
  %-----------------------------------------
 end
```

Per Class Errors	
Airplanes	0
Cars	0.05
Dogs	0.2
Faces	0
Keyboard	0.5

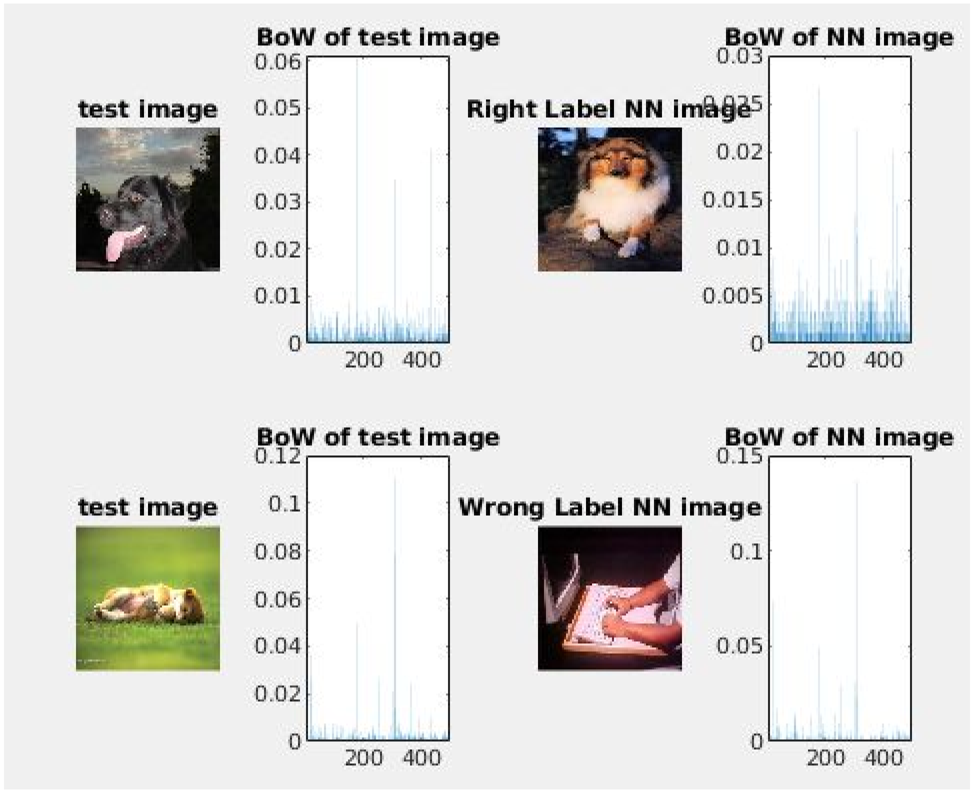
Overall Error
0.15



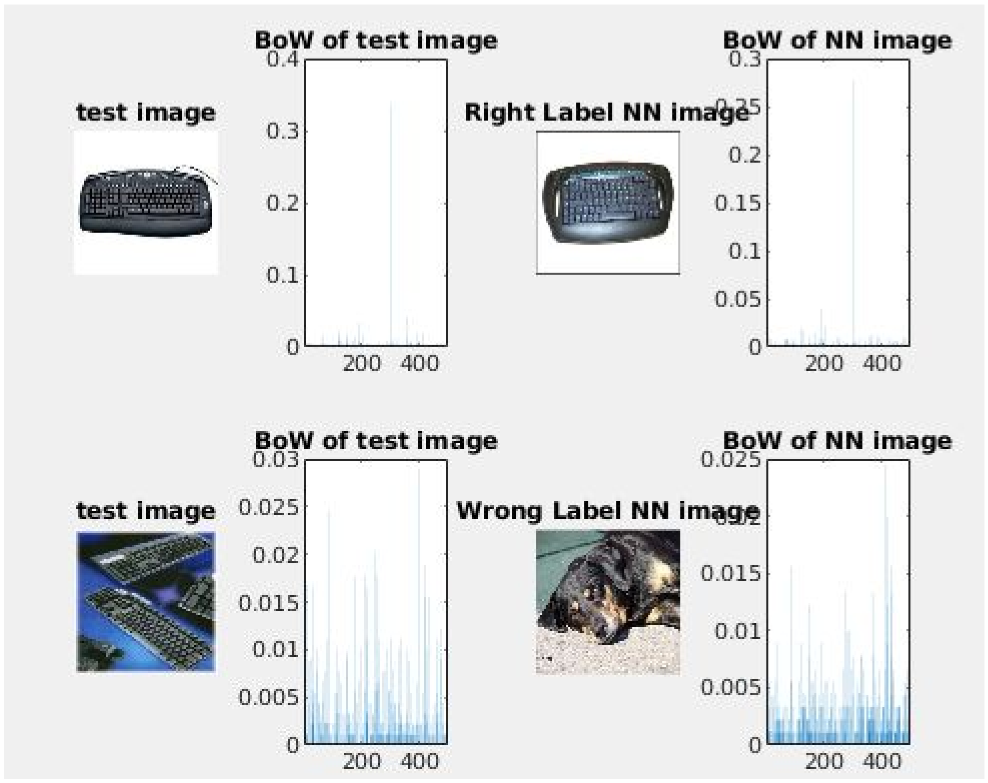
Car:


test image
BoW of test image
Right Label NN image
BoW of NN image
test image
BoW of test image
Wrong Label NN image
BoW of NN image

Dog:


test image
BoW of test image
Right Label NN image
BoW of NN image
test image
BoW of test image
Wrong Label NN image
BoW of NN image

Keyboards:



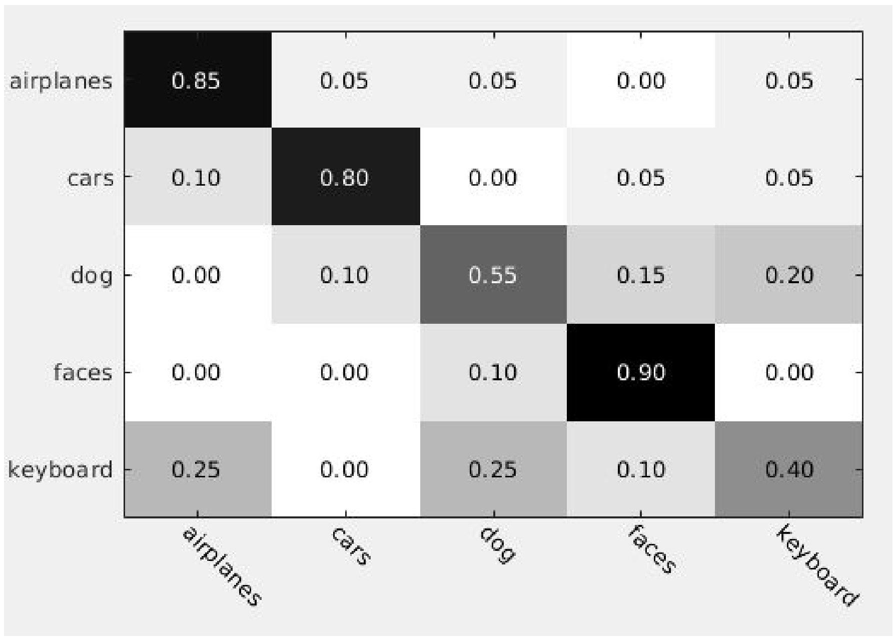
5. Dictionary size

1. After performing the classification experiment with a very small dictionary(DictionarySize=20), it is observed that accuracy decreases and errors increase. The BoW matrix is now 400X20. The resulting confusion matrix and errors have been reported.

Method 1:

Per Class Errors	
Airplane	0.15
Car	0.2
Dog	0.45
Faces	0.1
Keyboards	0.6

Overall Error
0.3

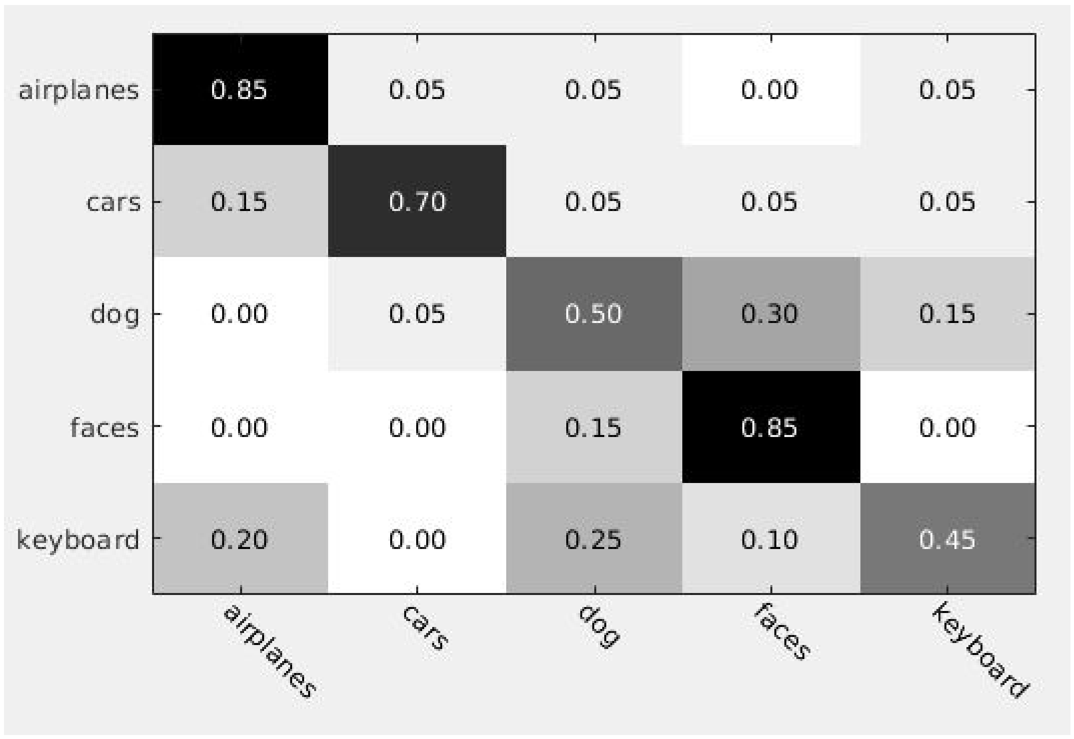


	airplanes	cars	dog	faces	keyboard
airplanes	0.85	0.05	0.05	0.00	0.05
cars	0.10	0.80	0.00	0.05	0.05
dog	0.00	0.10	0.55	0.15	0.20
faces	0.00	0.00	0.10	0.90	0.00
keyboard	0.25	0.00	0.25	0.10	0.40

Method 2:

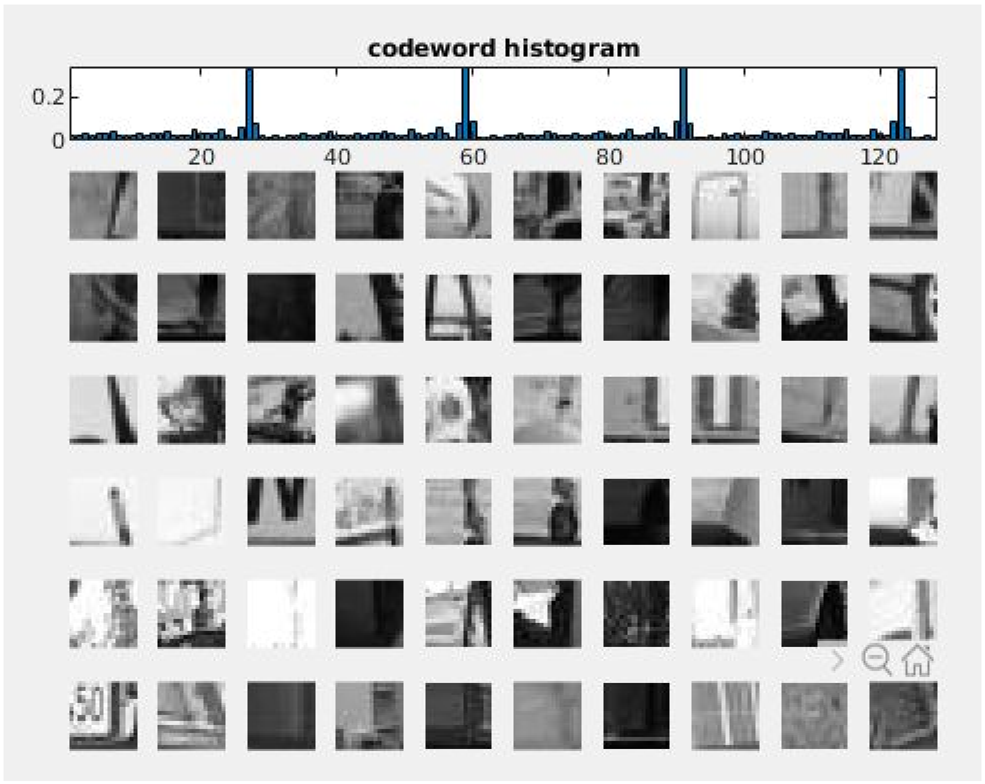
Per Class Errors	
Airplane	0.15
Car	0.3
Dog	0.5
Faces	0.15
Keyboards	0.55

Overall Error
0.33



	airplanes	cars	dog	faces	keyboard
airplanes	0.85	0.05	0.05	0.00	0.05
cars	0.15	0.70	0.05	0.05	0.05
dog	0.00	0.05	0.50	0.30	0.15
faces	0.00	0.00	0.15	0.85	0.00
keyboard	0.20	0.00	0.25	0.10	0.45

2. From the histogram in the figure it is evident that patches of the same kind have been repeated many times in the whole codebook. This is the result of the dictionary size being too small(20). As a result, this dictionary is not enough to understand the train and test data, resulting in underfitting. Hence, there is a drop in performance as it is difficult to get good accuracy with less codeword vectors for a large dataset.



6. Image classification using a Support Vector Machines classifier

1. The given code in Section 6.1 was executed to train the model with a linear multiclass SVM algorithm for each image class. The code was uncommented to observe the procedure that calculates the optimal values for C with the help of cross validation technique. It was evident that the accuracy was improving with the number of iterations.

```
Cross Validation Accuracy = 87.3333%
10 1.5 87.3333 (best c=512, g=0.812252, rate=87.6667)
```

2. The trained SVM model was applied to the test images using the given code and all the images were classified. Accuracy is reported here.

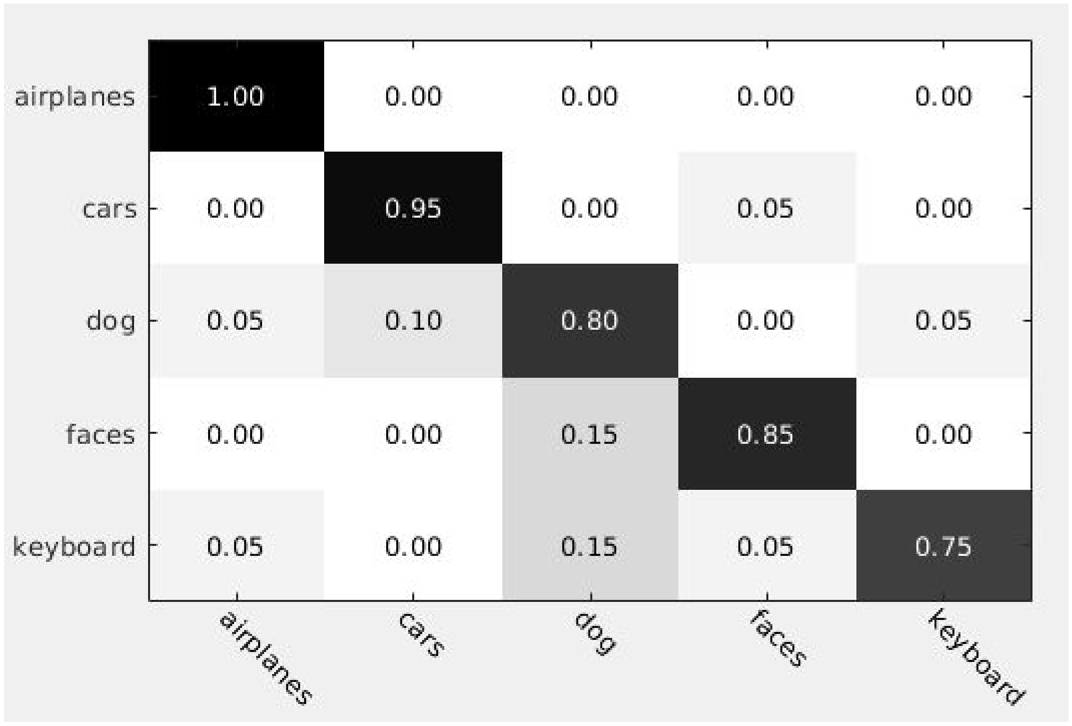
```
Accuracy = 87% (87/100) (classification)
```

3. The overall error and error per class were observed as follows:

Per Class Errors	
1	0
2	0.05
3	0.2
4	0.15
5	0.25

Overall Error
0.13

4. The confusion matrix for the SVM model:



	airplanes	cars	dog	faces	keyboard
airplanes	1.00	0.00	0.00	0.00	0.00
cars	0.00	0.95	0.00	0.05	0.00
dog	0.05	0.10	0.80	0.00	0.05
faces	0.00	0.00	0.15	0.85	0.00
keyboard	0.05	0.00	0.15	0.05	0.75

5. Some of the correctly classified images and some of the incorrectly classified images are reported here. SVM does not perform very well in the case of overlapping classes, i.e when the data points are not well separated. As SVM is not a probabilistic model, it becomes difficult to explain the classification in terms of probability. In the images provided here, many are very similar to each other and yet belong to a different class. SVM does a poor job in defining a boundary for such instances and as a result we get some incorrect classifications.