190573735 - Neeraj Vashistha

ECS797 Machine Learning for Visual Data Analysis
Lab 1: Image Classification using a Bag-of-Words model

Section 2: Dictionary creation – feature quantization

1. Executing the given code section successfully creates and load variables TrainMat(270000 X 128) and TestMat (90000X128).

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

2. The code given in section 2.2 executes successfully and creates a dictionary with 500 words. The number of clusters determines the size of the dictionary. This dictionary is stored in C(500x128) is saved in dictionary.mat

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

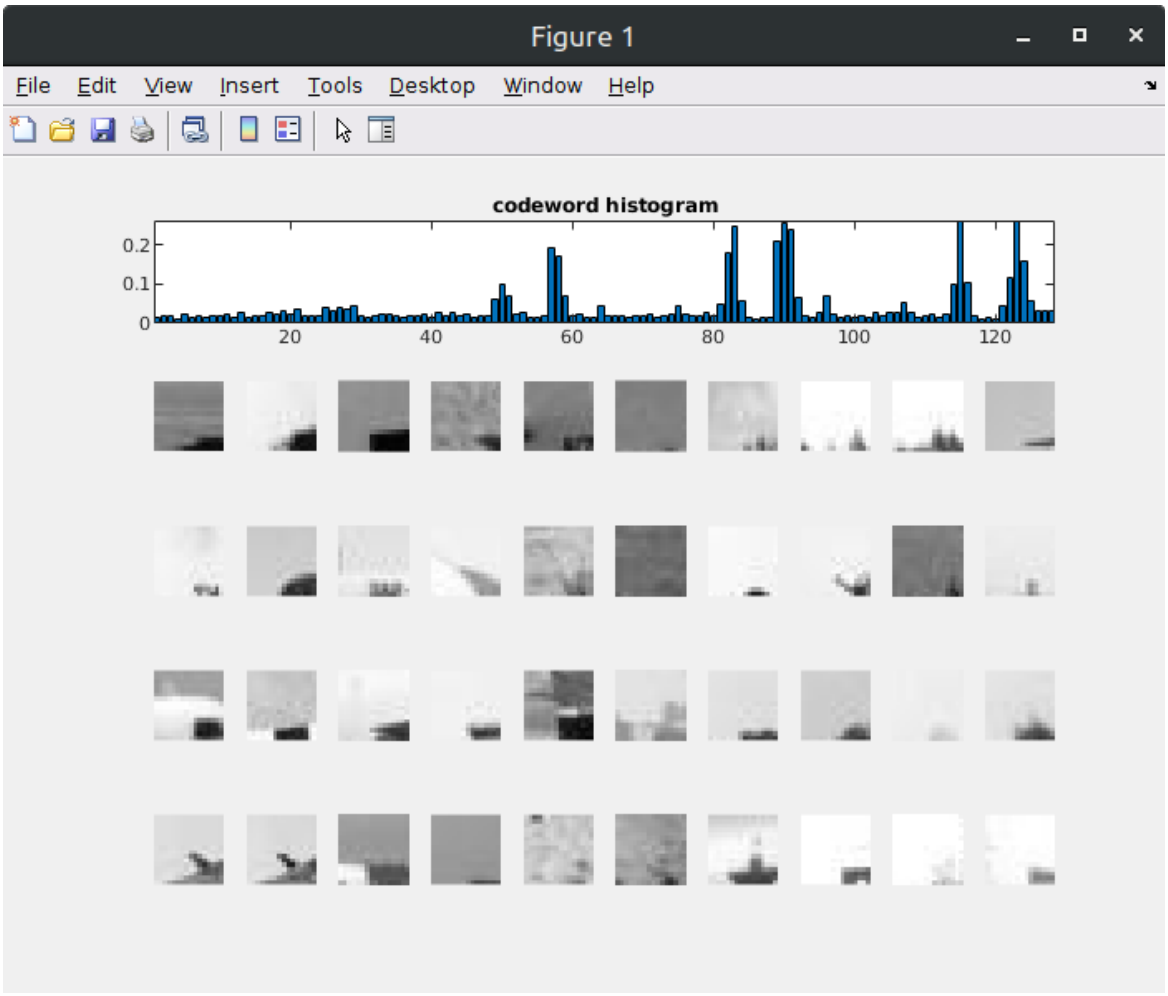
3. The next section of codes uses the code definition in file EuclideanDistance.m to calculate the distance between image descriptors and codewords.
4. The code used to assign each descriptor in the dataset to the nearest cluster or codeword is displayed below. And as a result index_train(1x270000) and index_test(1x90000) are created that contain the indices of the assigned codewords.

```
%-----------------------------Write Your Own Code here that assigns all descriptors ------------
index_train = zeros(size(TrainMat,1),1); % initalise with size of training samples
index_test = zeros(size(TestMat,1),1); % initalise with size of test samples

for k=1:size(TrainMat,1) % iterate over and compute for each training sample
    [minv,index] = min(EuclideanDistance(TrainMat(k,:),C)); % compute codeword for each training sample
    index_train(k,1) = index; % store the index only for the nearest code word
end

for k=1:size(TestMat,1) % iterate over and compute for each test sample
    [minv,index] = min(EuclideanDistance(TestMat(k,:),C)); % compute codeword for each test sample
    index_test(k,1) = index;
end
%-----------------------------Write Your Own Code here that assigns all descriptors ------------
```

5. The image patches belonging to the same codeword are displayed below.



Section 3: Image representation using Bag of Words representation

1. The representation of each image as a histogram of Bag of Words has been displayed. It was normalised using the predefined do_normalise() function. In the dataset 1 to 300 are the training images and whereas 301 to 400 are the testing images. The code used to generate this output and the output is as follows.

190573735 - Neeraj Vashistha

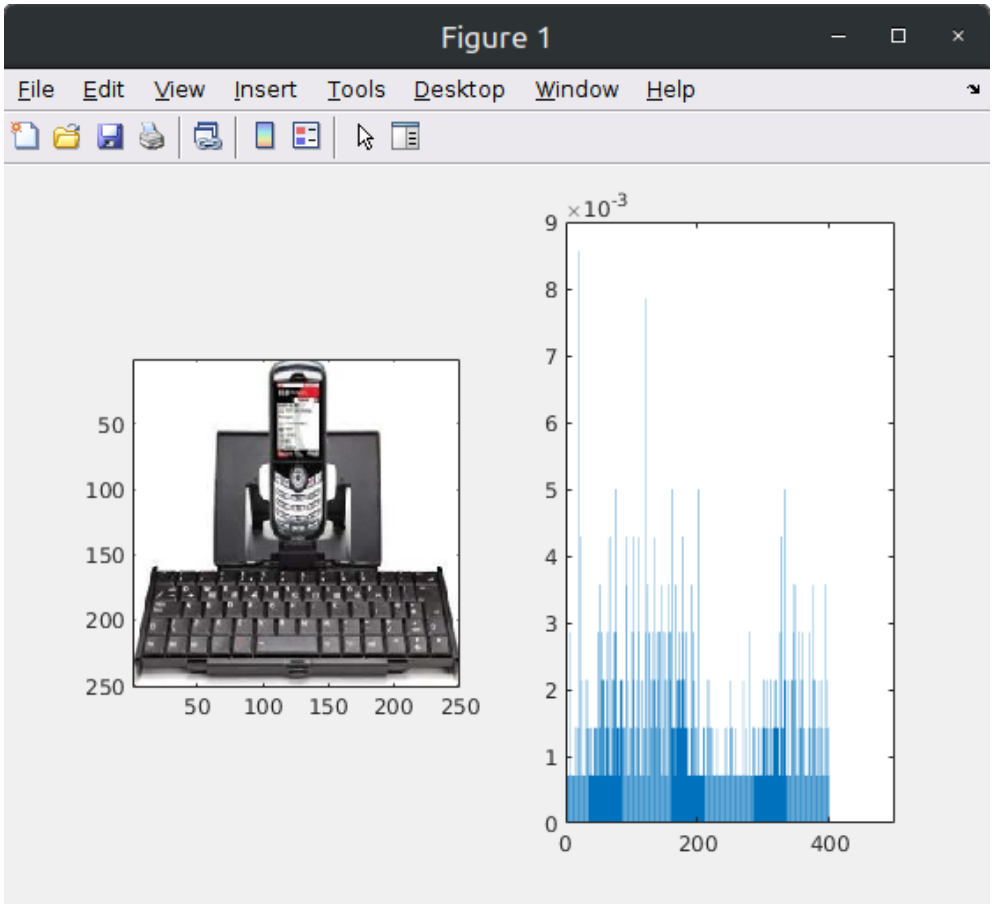
```
for ii = 1:nimages
      image_dir=sprintf('%s/%s/','data/local',num2string(ii,3));
      inFName = fullfile(image_dir, sprintf('%s', 'sift_features'));
      load(inFName, 'features');
      %------------------------ write your own code here------------------------
      % for each image there are 900 features
      % then we get 900 indices
      % by counting occurences you construct 500 columns feature vector BoW

      % 1 2 1 3 1 (900) ->   3 1 1(500)

      d = EuclideanDistance(features.data, C);
      Row = size(d,1);
      WordCluster = zeros(1,vocbsize);
      for i =1:Row
        [minv,index] = min(d(i,:));
        WordCluster(1,index) = WordCluster(1,index)+1;
      end
      BoW(ii,:) = do_normalize(WordCluster);

      %------------------------ write your own code here------------------------

      if isshow == 1
        close all; figure;
        subplot(1,2,1),subimage(imread(strcat('image/',image_names{ii})));
        subplot(1,2,2),bar(BoW(ii,:)),xlim([0 500]);
      end
end
```



190573735 - Neeraj Vashistha

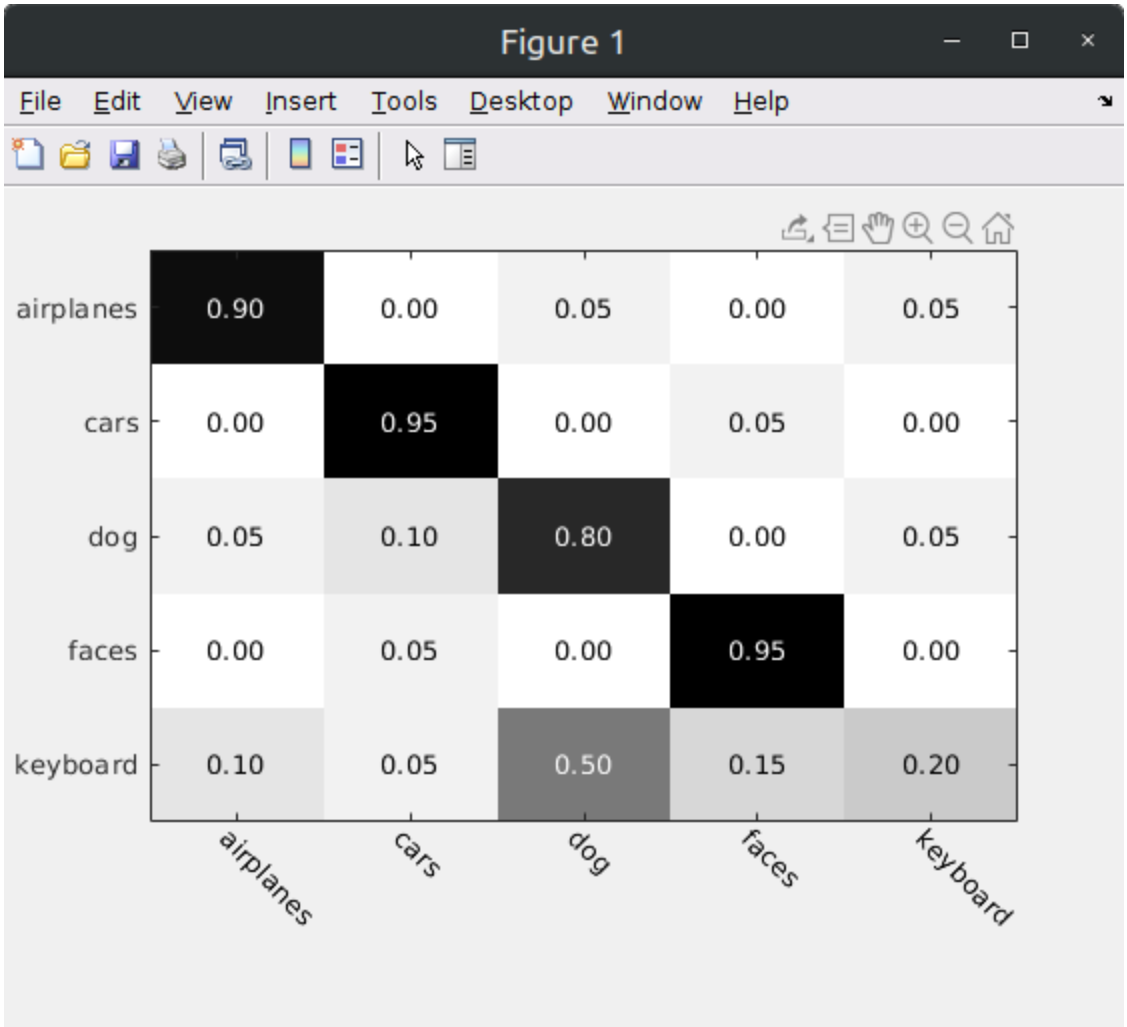
Section 4: Image Classification using a Nearest Neighbor(NN) Classifier

1. This section calculates the L2(Euclidean distance) between the test and training images and classifies each image as the label of the nearest neighbor in the training set. This is done using the provided code and the desired output is achieved.
2. The classification error per class and the overall error are reported here:

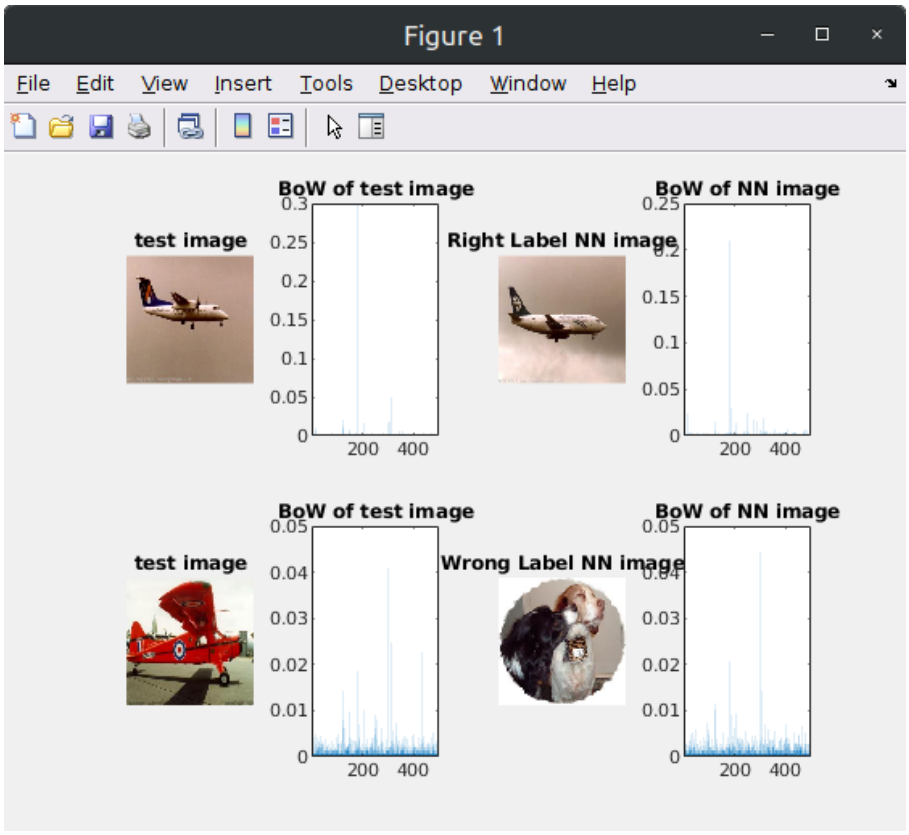
Overall Error
0.24

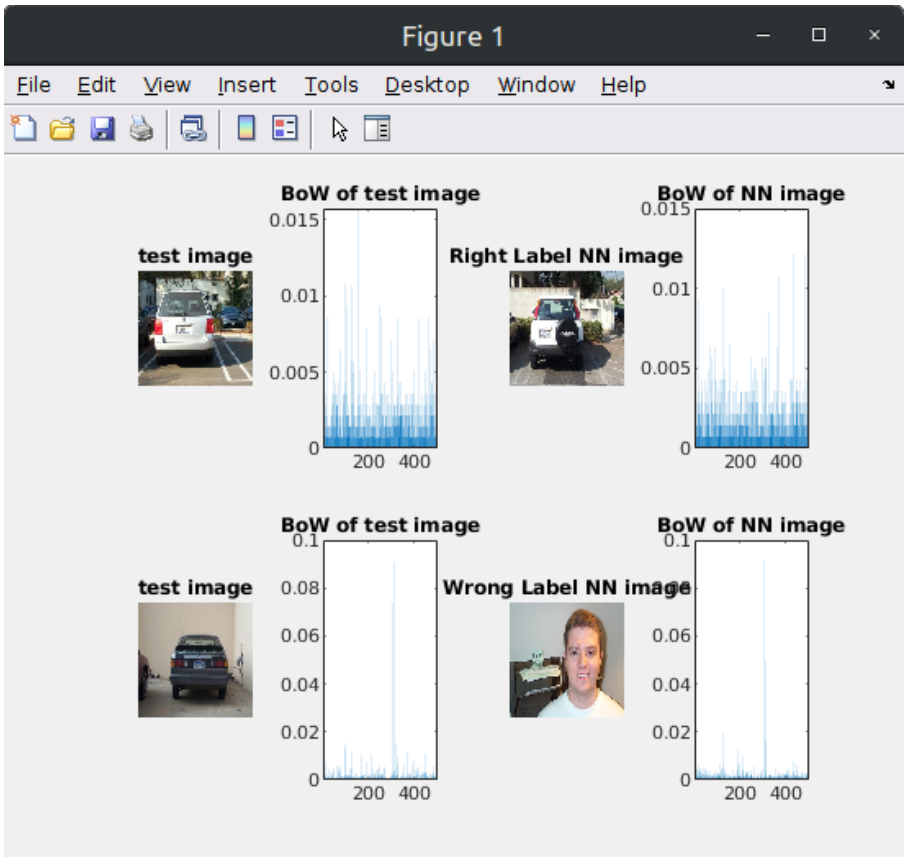
Per Class Errors	
Airplanes	0.1
Cars	0.05
Dogs	0.2
Faces	0.05
Keyboard	0.8

3. The confusion matrix achieved by executing the given code is:

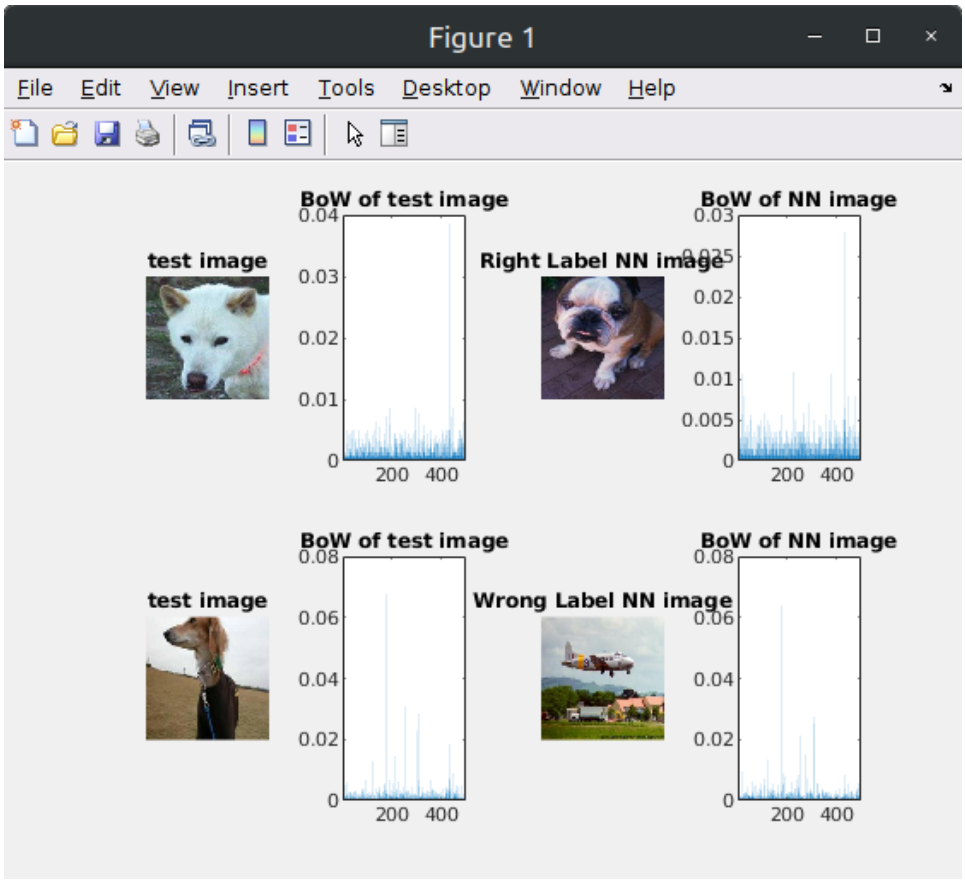


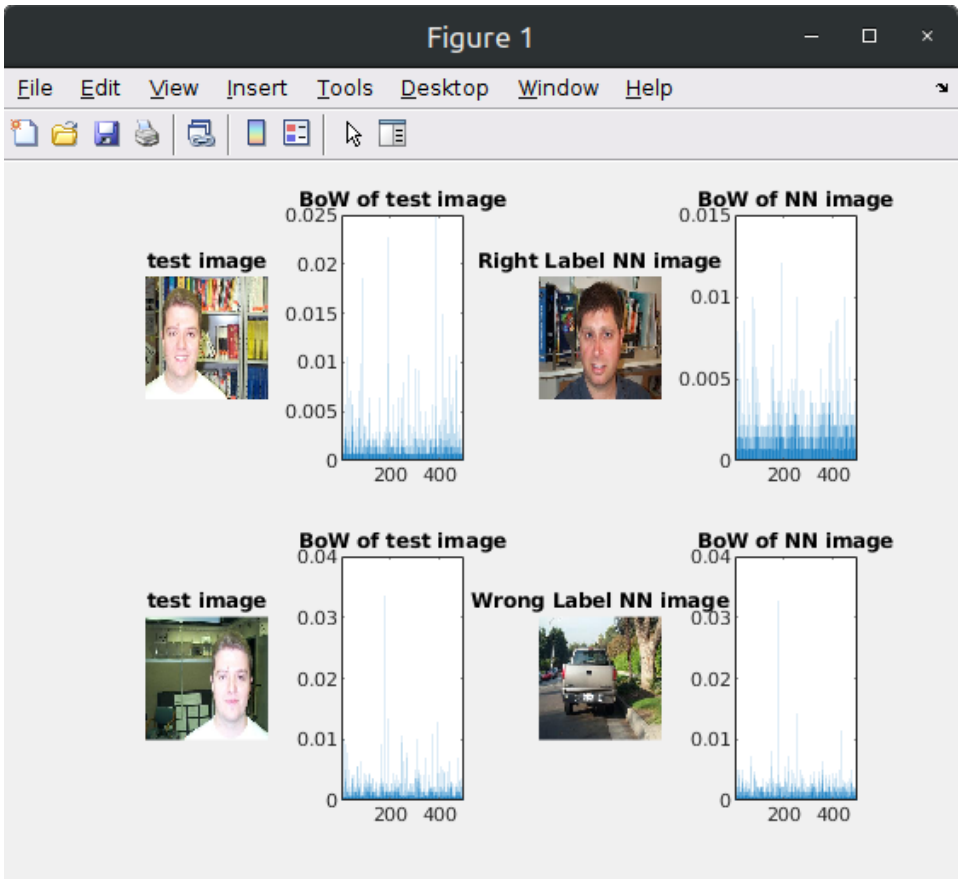
4. Some images that were correctly and incorrectly classified for each class:



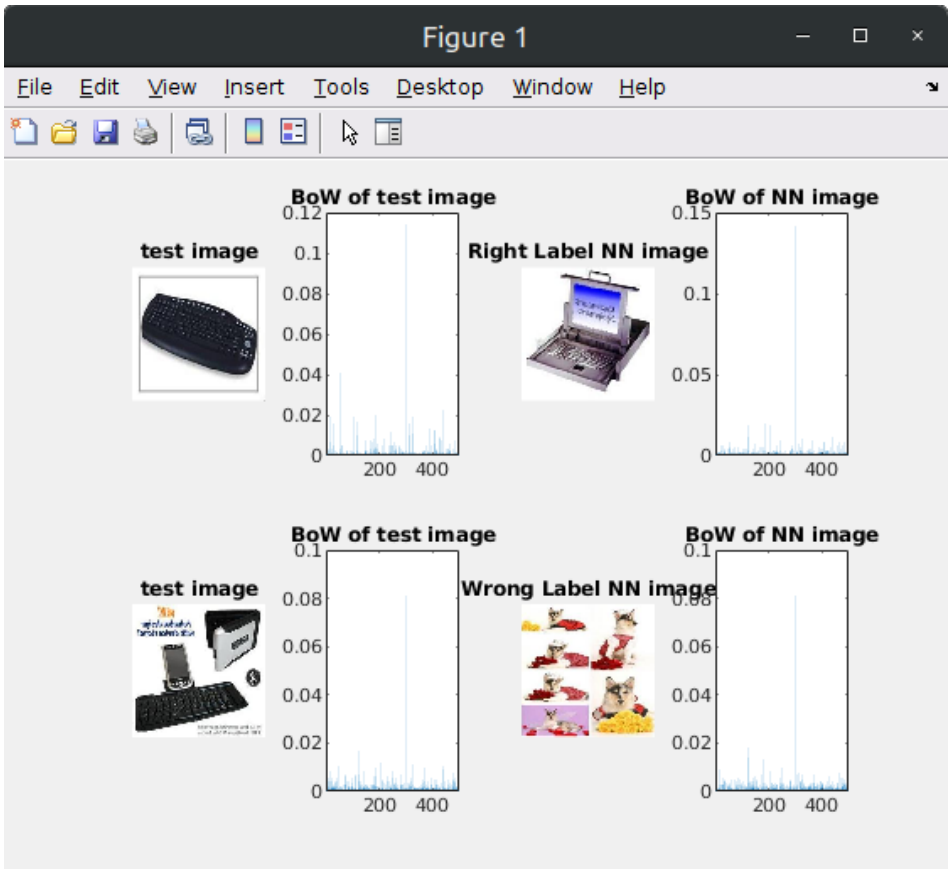


190573735 - Neeraj Vashistha





190573735 - Neeraj Vashistha



5. kNN classification was done with Histogram intersection method by writing the following code and setting method=2. Generated outputs are:

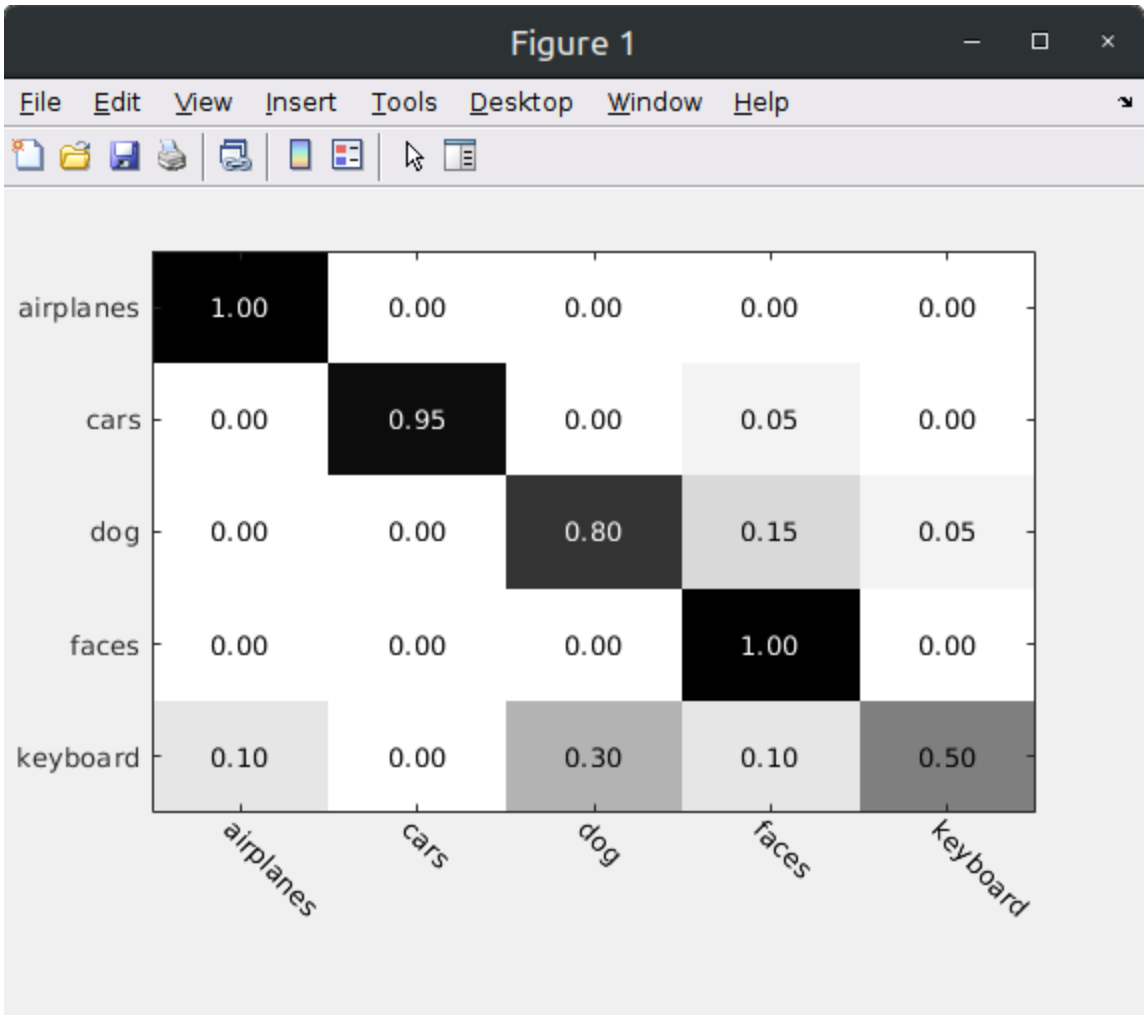
```
% -------------- write your own code here ---------------
d = 1;

for i = 1:p
  d = d - min(a(1,index),b(1,index));
end
% -------------- write your own code here ---------------
```

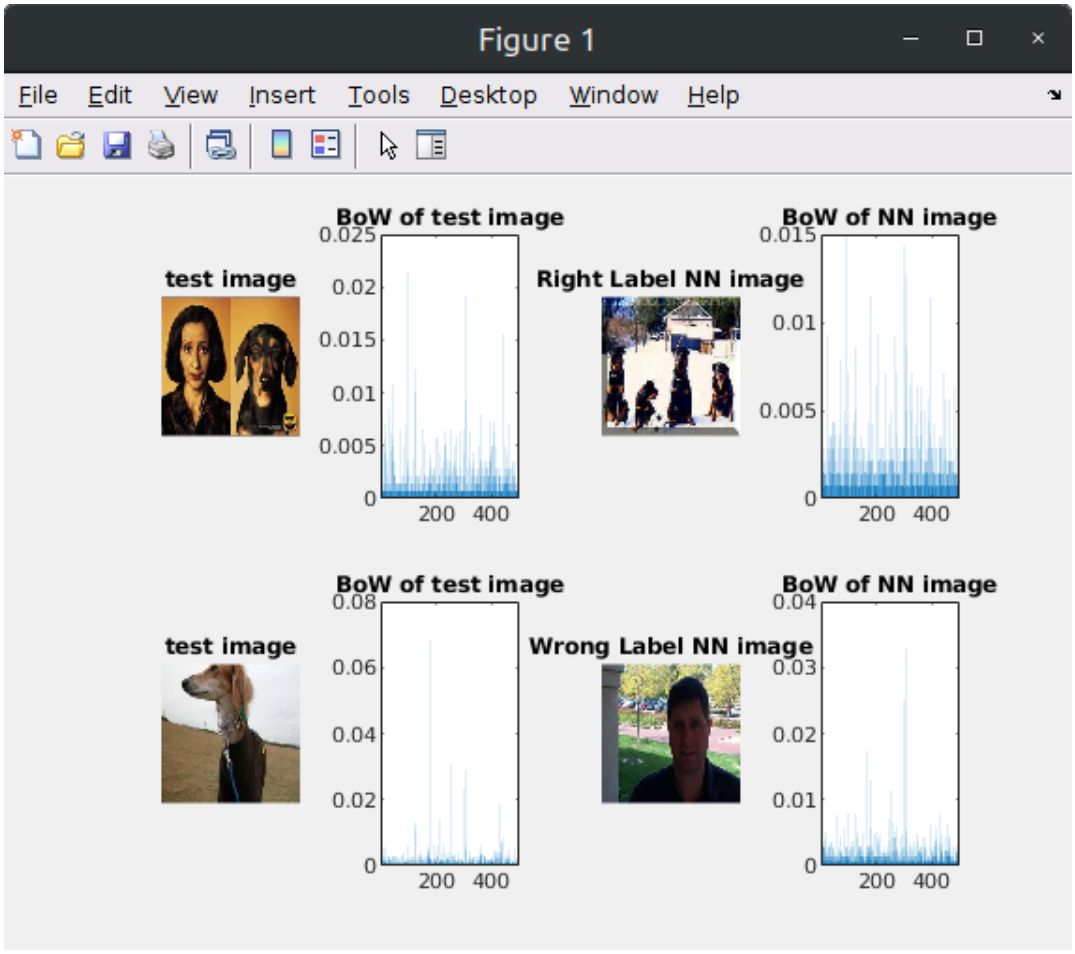
Overall Error
0.15

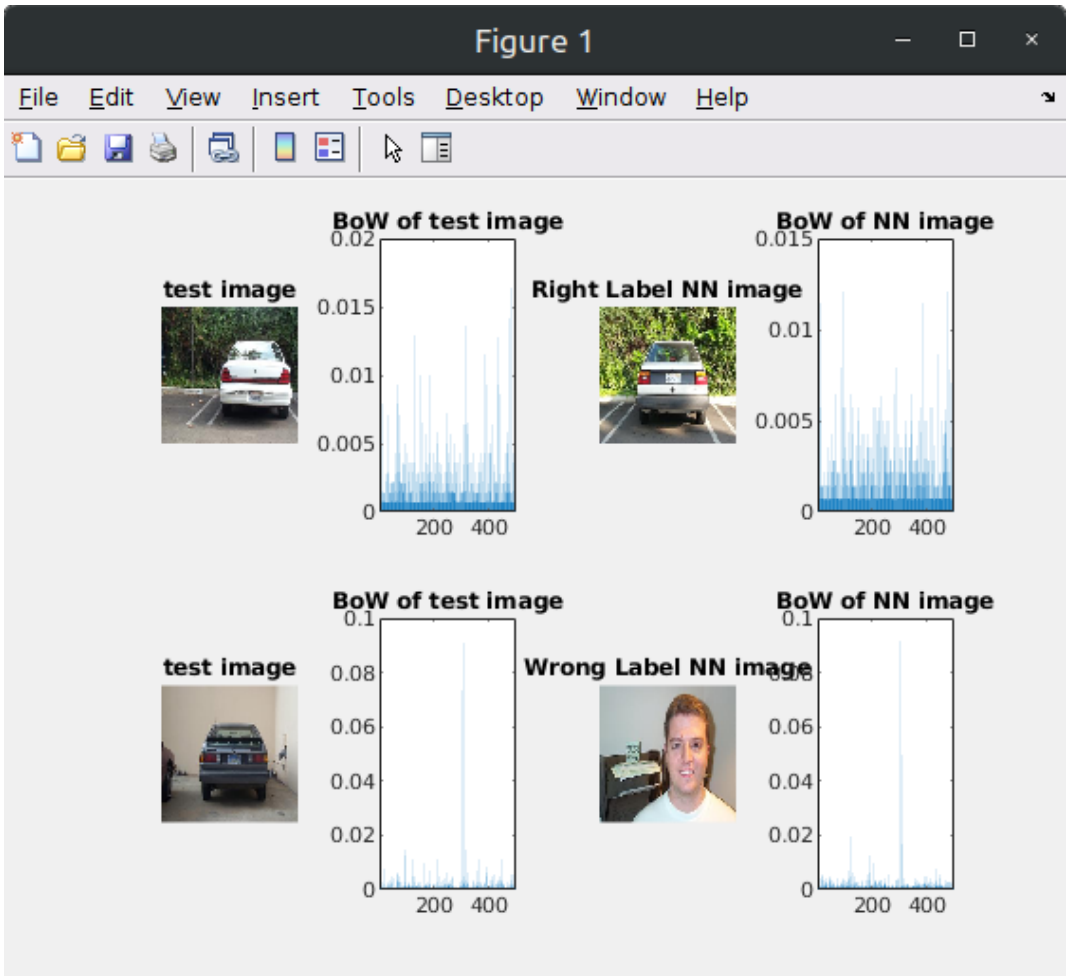
Per Class Errors	
Airplanes	0
Cars	0.05
Dogs	0.2
Faces	0
Keyboard	0.5

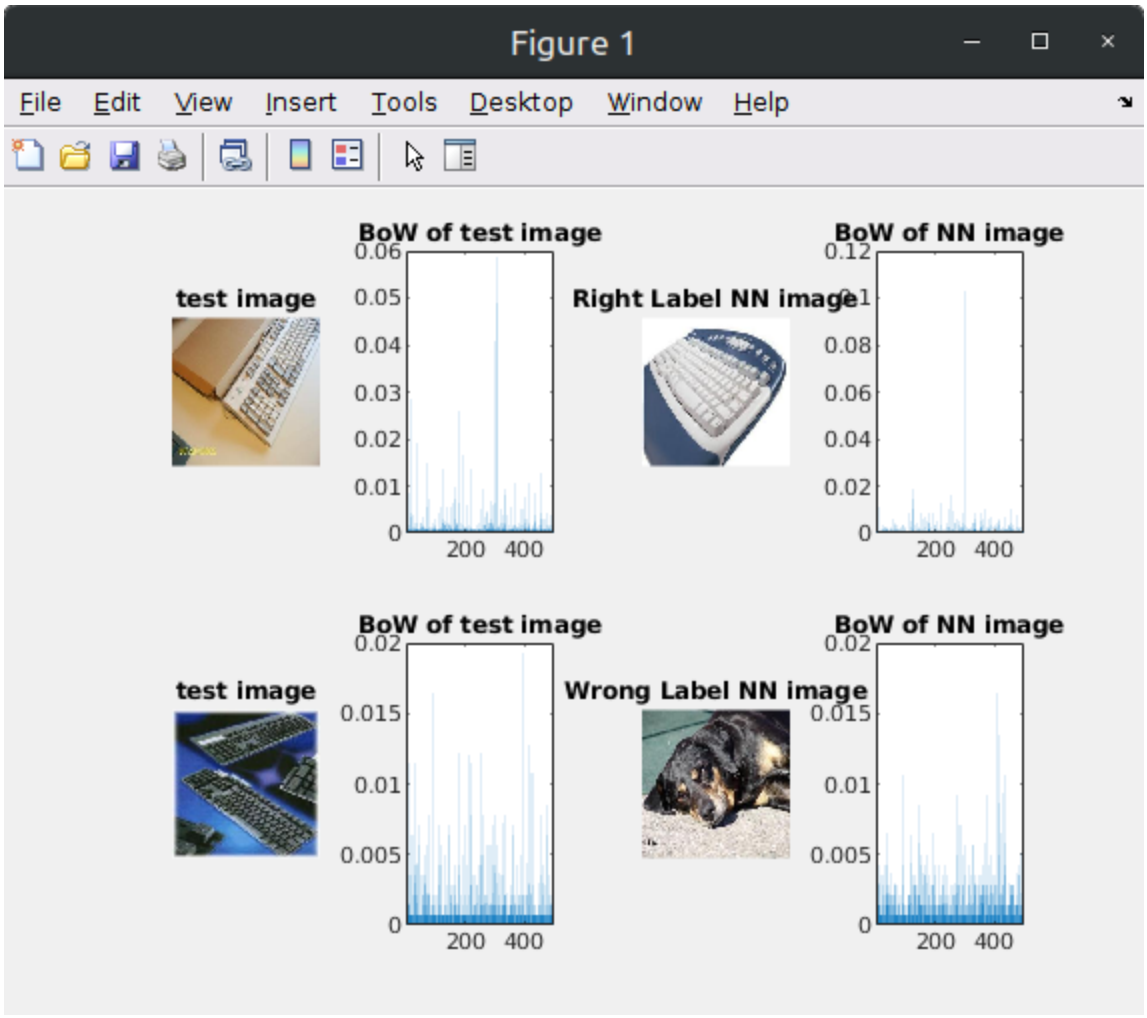
190573735 - Neeraj Vashistha



190573735 - Neeraj Vashistha







Section 5: Dictionary Size

1. Performing the classification with a very small dictionary size results in reduced accuracy and increased errors. The BoW matrix is now 400x20. Corresponding results are as follows.

Method 1:

Overall Error
0.3

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

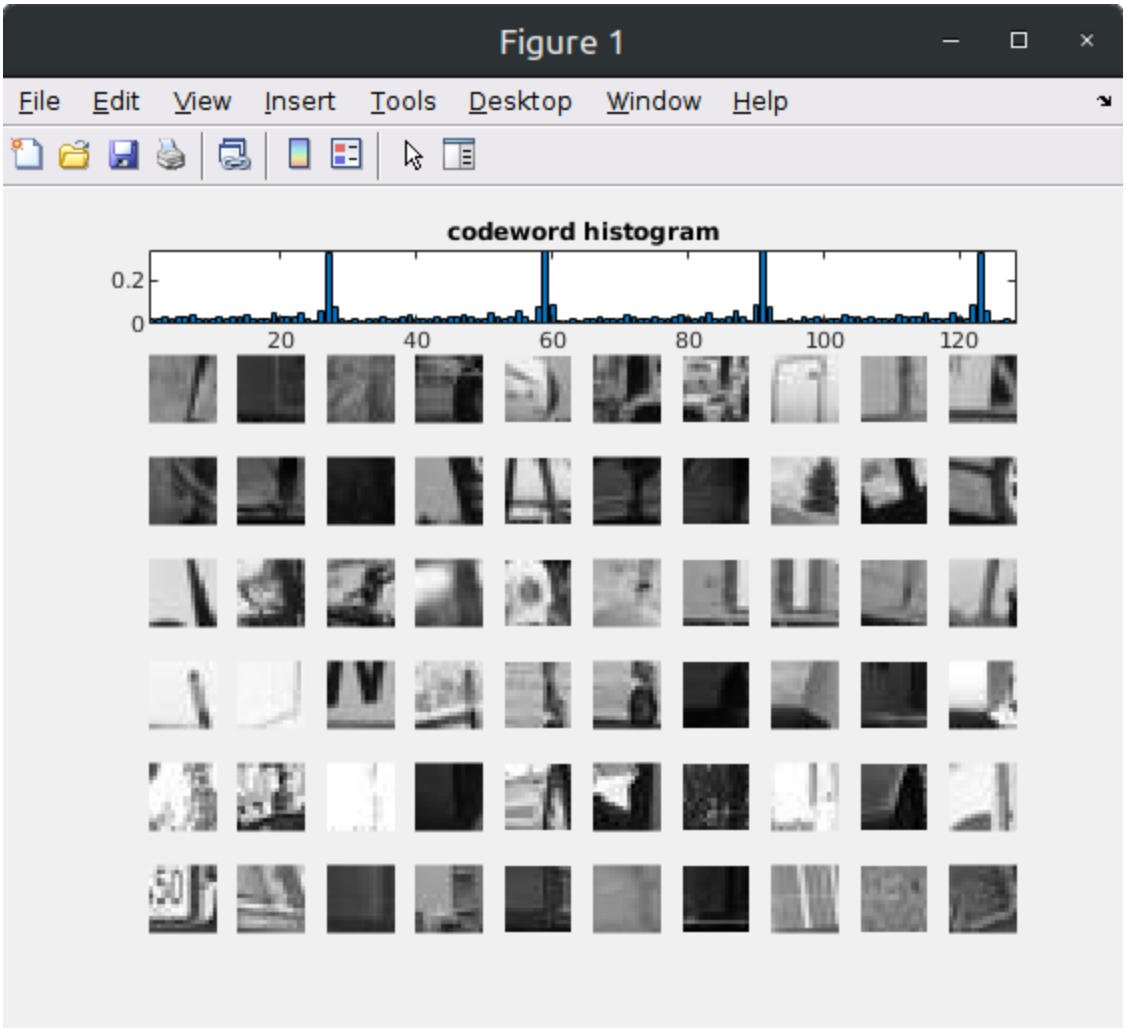
190573735 - Neeraj Vashistha

Method 2:

Overall Error
0.33

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

2. After executing the classification steps again it was observed that there is a drop in performance. This is due to the size of the dictionary. It is too small for the train and test dataset that was presented. As a result, it creates a classic machine learning problem of underfitting. It is also seen in the histogram which says that there are multiple repetitive patches of images of the same kind in the whole codewords set.



190573735 - Neeraj Vashistha

Section 6: Image Classification using a Support Vector Machines Classifier

1. To train the model with a linear multiclass SVM algorithm for each class of image, predefined code provided in 6.1 was executed and the results were observed. Cross validation is being used to calculate the optimal values for C using the commented code. The accuracy improves with each iteration and a total of 168 iterations were performed.

```
Cross Validation Accuracy = 75.6667%
10 1.5 75.6667 (best c=32, g=2.63902, rate=81.3333)
```

2. After classifying all the test images the overall accuracy was 87%.

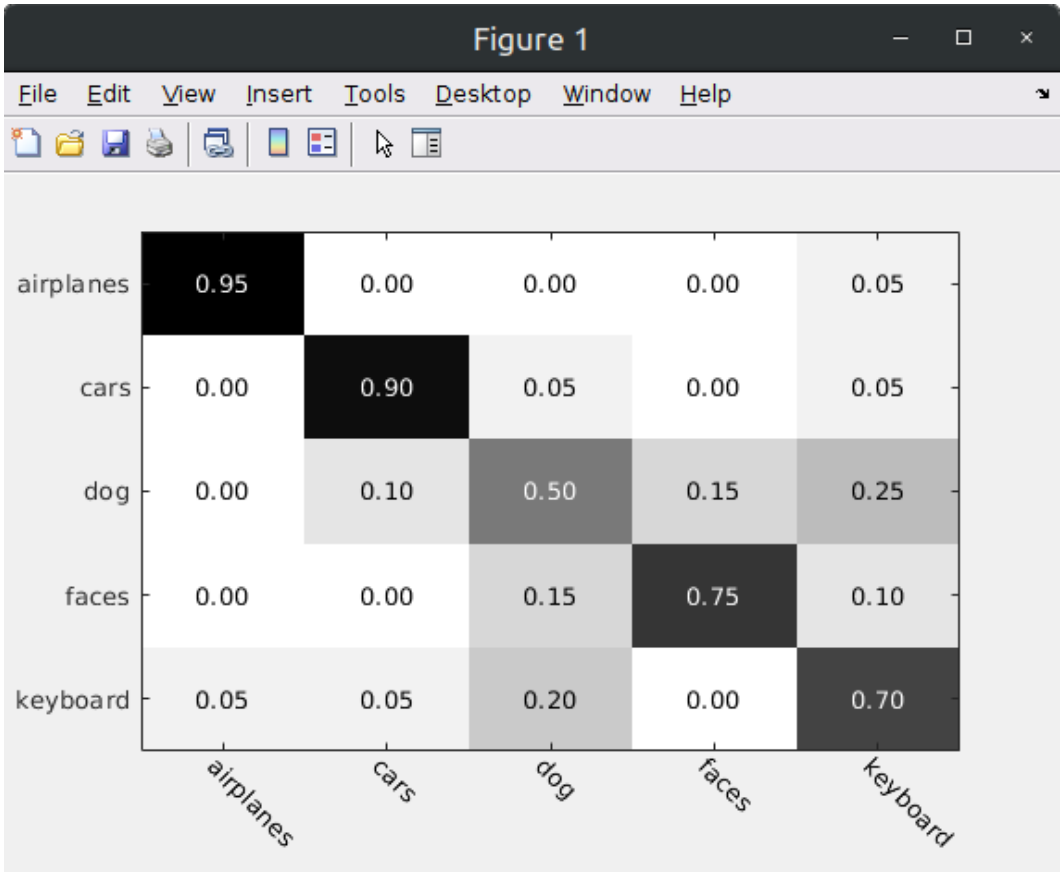
```
Accuracy = 76% (76/100) (classification)
```

3. Per class errors and the overall error is reported here.

Overall Error
0.24

Per Class Errors	
1	0.05
2	0.1
3	0.5
4	0.25
5	0.3

4. Confusion matrix:



5. SVM works very well when the data points are well separated. But performs poorly otherwise, which is in the case of overlapping classes. The situation is very similar here, as there are many images in the dataset that have a similar structure, as can be seen from the histogram. Hence, SVM cannot create a proper boundary between the datasets which results in incorrect classifications.