Cara Evangeline Chandra Mohan
190539421

MACHINE LEARNING FOR VISUAL DATA ANALYSIS LAB - 1

1. Write your own code that assigns each descriptor in the training and test images to the nearest codeword cluster. (Hint: use the function min()) % Assume the assignment vector of training images is index_train and that of test images is index_test, the dimensions of which should be 1x270000 and 1x90000 respectively.

```
%--------------Write Your Own Code here that assigns all descriptors -------------
for i = 1:90000
   dis_test = TestMat(i,:);
   d = EuclideanDistance(dis_test, C);
   [minv,index] = min(d);
   index_test(i) = index;
end
for i = 1:270000
   dis_train = TrainMat(i,:);
   d = EuclideanDistance(dis_train, C);
   [minv,index] = min(d);
   index_train(i) = index;
end
save('data/global/assignd_discriptor','index_train','index_test');
```

2. Visualise some image patches that are assigned to the same codeword.

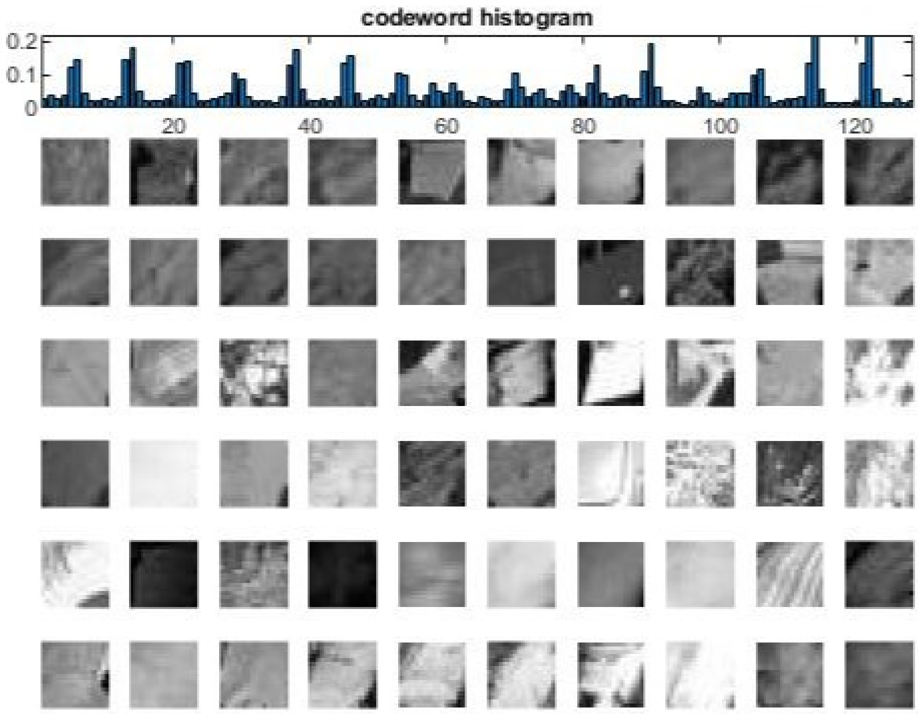


Fig 1: 30 train, 30 test image patches with codeid=78

Cara Evangeline Chandra Mohan
190539421

3. Represent each image in the training and the test dataset as a histogram of visual words (i.e. represent each image using the Bag of Words representation). Normalise the histograms by their L1 norm. For normalisation you may want to call the function do_normalize() that is in the file do_normalize.m in the software directory, with the appropriate arguments.

```
for ii = 1:nimages
   image_dir=sprintf('%s/%s/','data/local',num2string(ii,3)); % location where detector is saved
   inFName = fullfile(image_dir, sprintf('%s', 'sift_features'));
   load(inFName, 'features');
   %----------------------- write your own code here----------------------------------------
   histogram = zeros(1,500);
   for j = 1:900
     d = EuclideanDistance(features.data(j,:), C);%calculate the distance
     %calculate the histogram (that should be a vector with the same dimensionality as the number of codewords and normalize it
     [minv,index] = min(d);
     histogram(index) = histogram(index) +1;
   end
   histogram = do_normalize(histogram);
   BoW(ii,:) = histogram;
   if isshow == 1
    close all; figure;
    subplot(1,2,1),imshow(imread(strcat('image/',image_names{7})));
    subplot(1,2,2),bar(BoW(ii,:)),xlim([0 500]);
   end
end
save('data/global/bow','BoW')
```

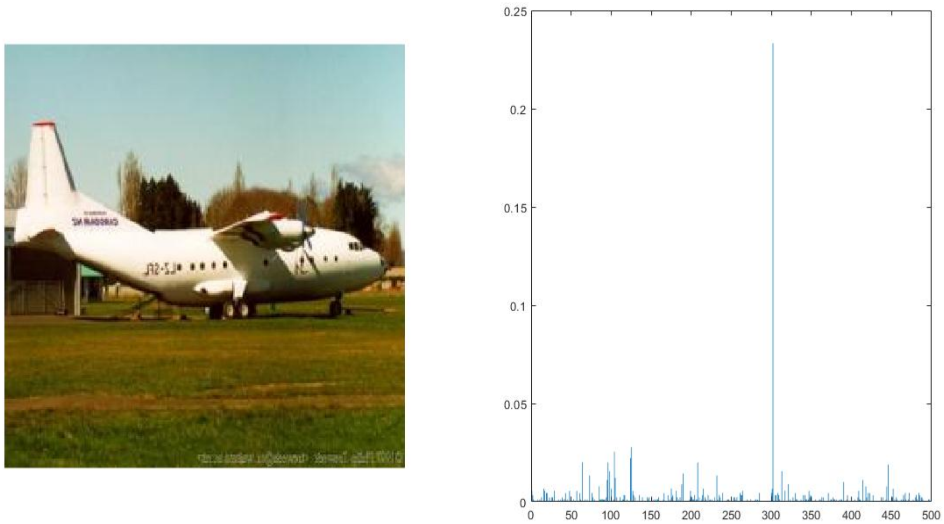


Fig 2: Image and it's corresponding histogram codewords

Cara Evangeline Chandra Mohan
190539421

KNN Algorithm with method = Euclidean Distance

Error and Confusion Matrix

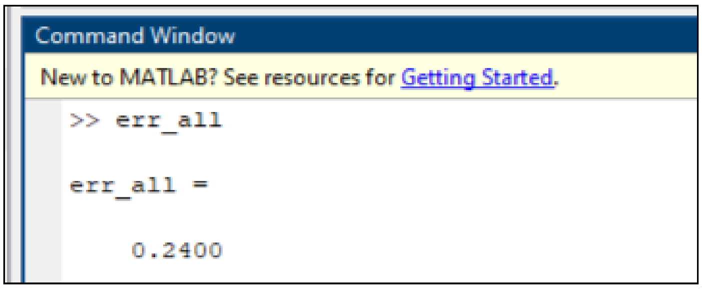


Fig 3: Error for Knn using Euclidean Distance

err_per_class = [0.10,   0.05,   0.20,   0.05,   0.80]

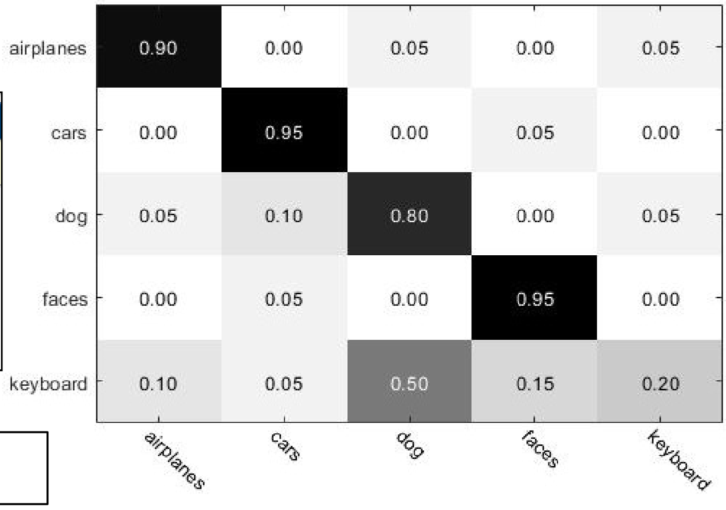


Fig 4: Confusion matrix for Knn using Euclidean distance

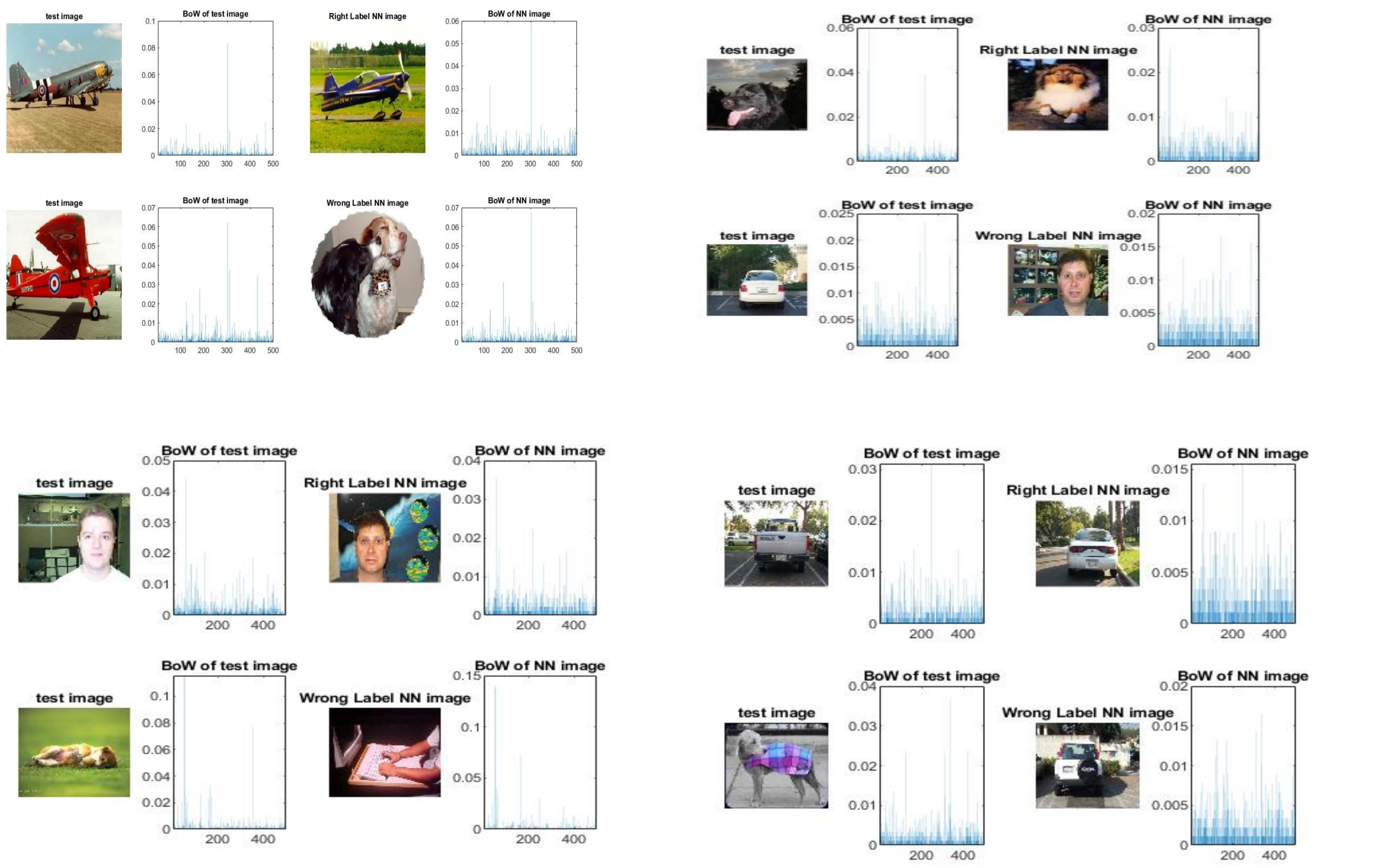


Fig 5: Correctly and Incorrectly Classified Instances

Reasons for failure:

1) Depends upon the value of k
2) Based on whether the classes are separable or non-separable
3) Depends upon the metric that we chose to calculate the distance.

Cara Evangeline Chandra Mohan
190539421

KNN Algorithm with method = histogram intersection

4. Write code for computing the histogram intersection between two histograms

```
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
   % -------------- write your own code here ---------------
    h1 = imhist(a);
    h2 = imhist(b);
    d = sum(min(h1,h2));
```

err_per_class = [1, 1, 0.90, 1, 0.50]

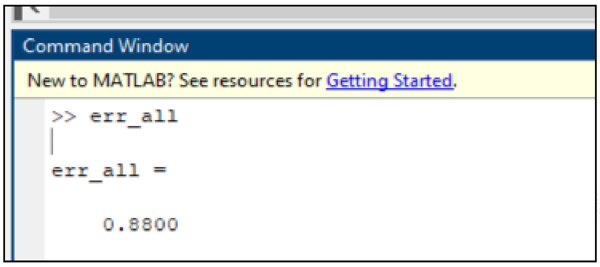


Fig 6: Error for Knn using Histogram Intersection

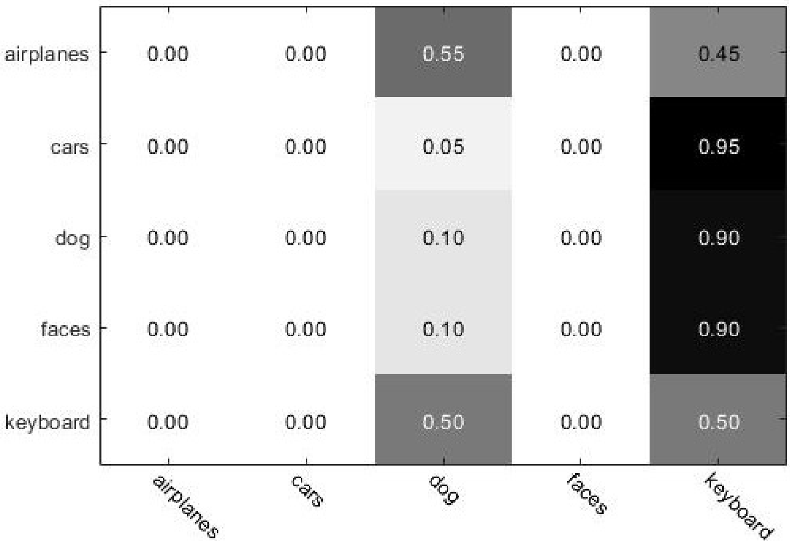


Fig 7: Confusion Matrix (Histogram Intersection)

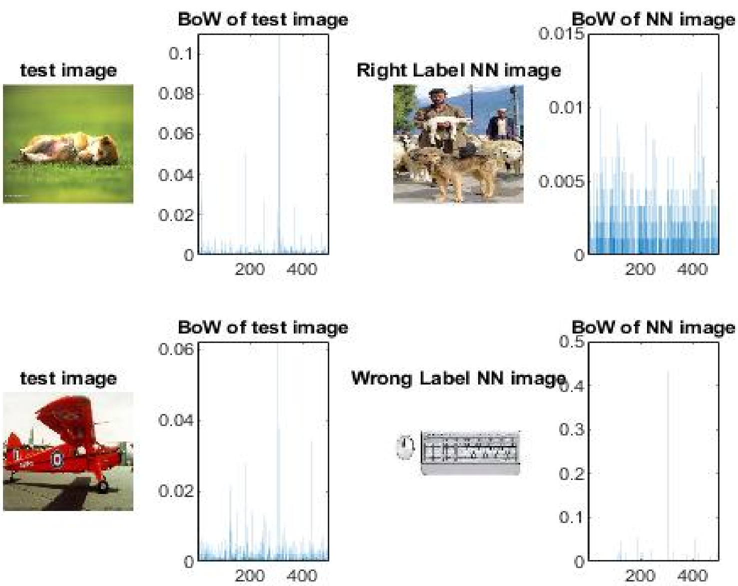


Fig 8: Correctly and Incorrectly Classified Instances

1) 'a' is the test sample
2) 'b' is the training sample
3) a and b are converted to corresponding pixel histogram
4) The intersection of both samples is obtained by using the min() function

Cara Evangeline Chandra Mohan
190539421

5. Perform the classification experiment using a very small dictionary and report the classification error and confusion matrices. Hint: Cluster the descriptors into a small number of clusters (e.g. 20).

Cluster Size = 20

Method = Euclidean Distance

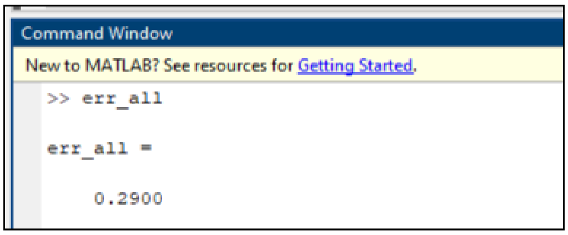
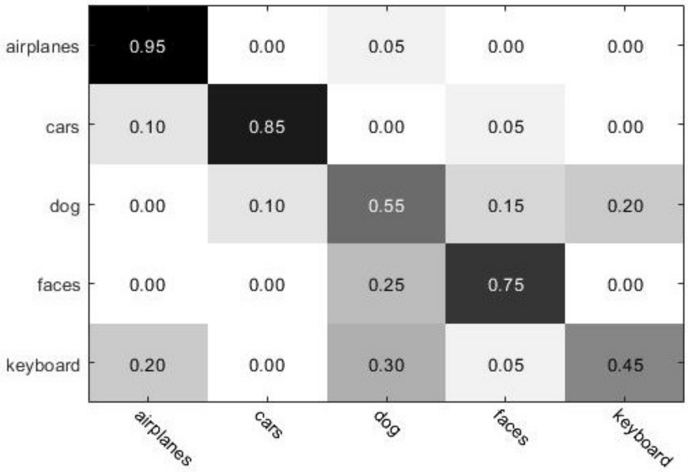


Fig 9: Error for Knn using Euclidean Distance



	airplanes	cars	dog	faces	keyboard
airplanes	0.95	0.00	0.05	0.00	0.00
cars	0.10	0.85	0.00	0.05	0.00
dog	0.00	0.10	0.55	0.15	0.20
faces	0.00	0.00	0.25	0.75	0.00
keyboard	0.20	0.00	0.30	0.05	0.45

Fig 10: Confusion matrix for Knn using Euclidean distance

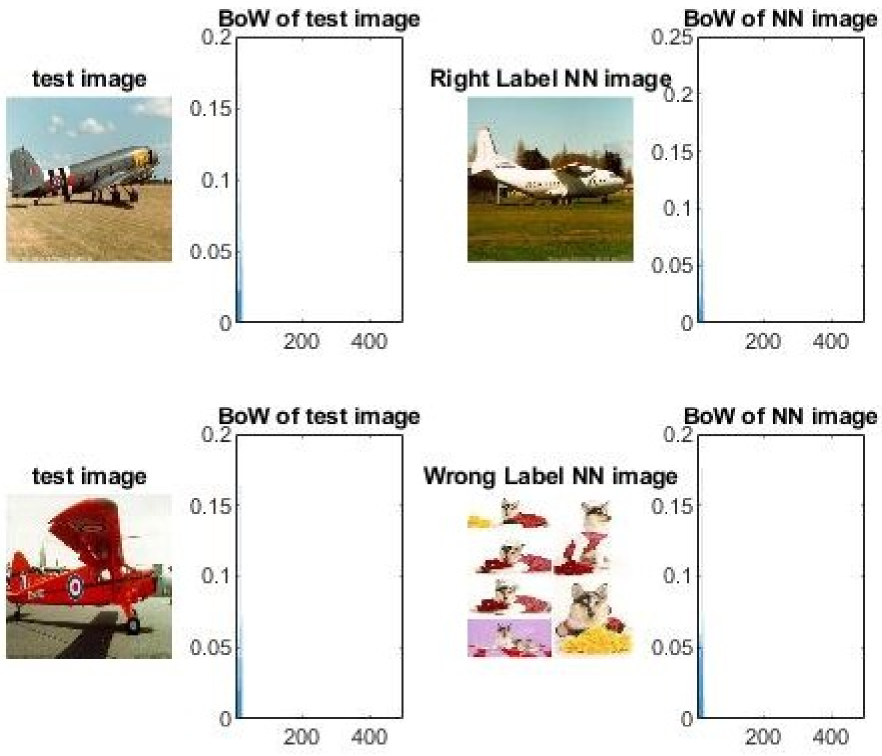


Fig 11: Correctly and Incorrectly Classified Instances

Cara Evangeline Chandra Mohan
190539421

Cluster Size = 20

Method = Histogram Intersection

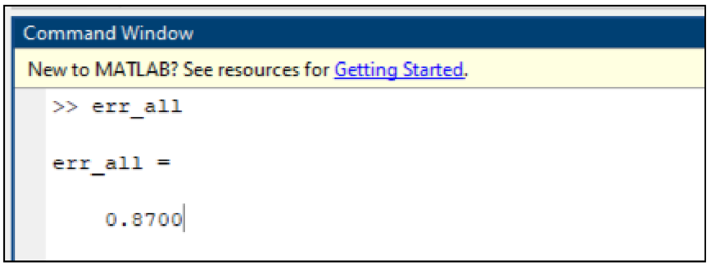
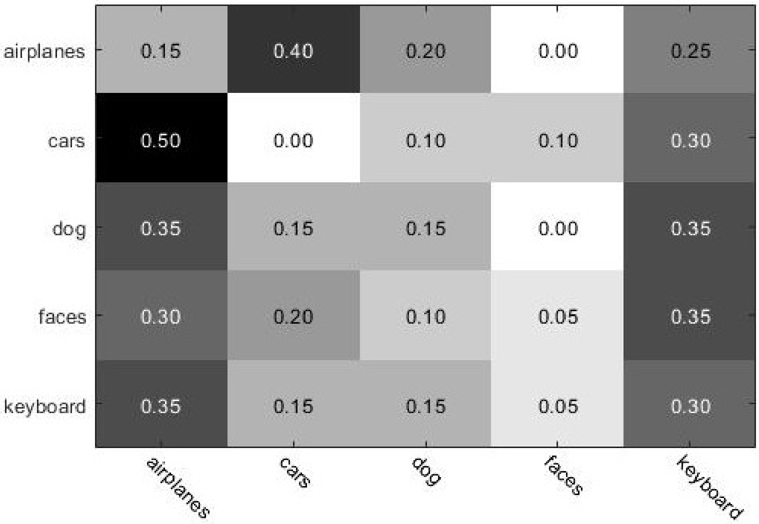


Fig 12: Error for Knn using Histogram Intersection



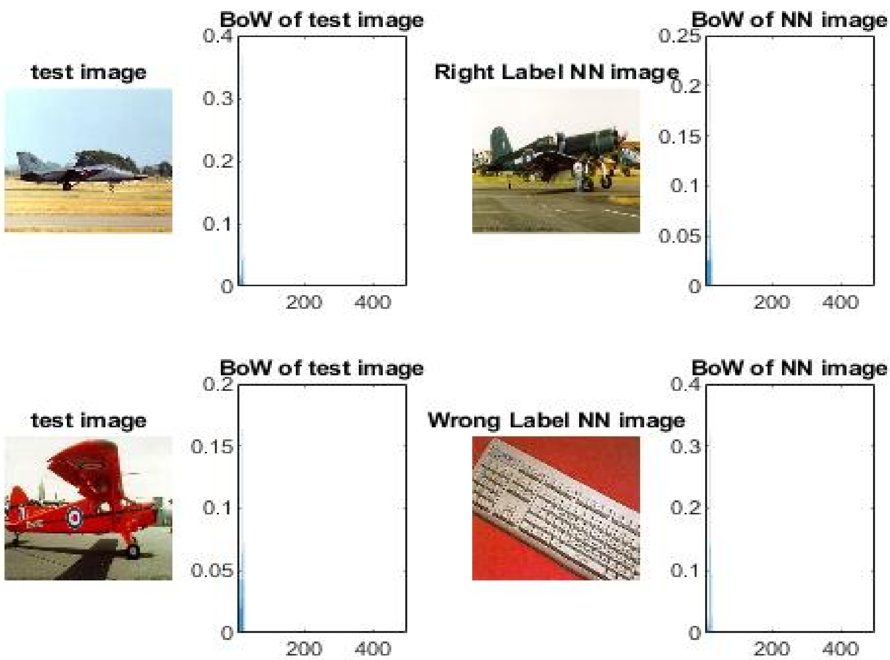


Fig 14: Correctly and Incorrectly Classified Instances

6. Explain the drop in the performance.

When the number of codewords is decreased to 20 from 500 there is a slight decrease in performance increasing the error rate in KNN classification. This is because the algorithm learns from very few instances (ie.codewords) and hence there is very less variability which leads to incorrect identification of images.

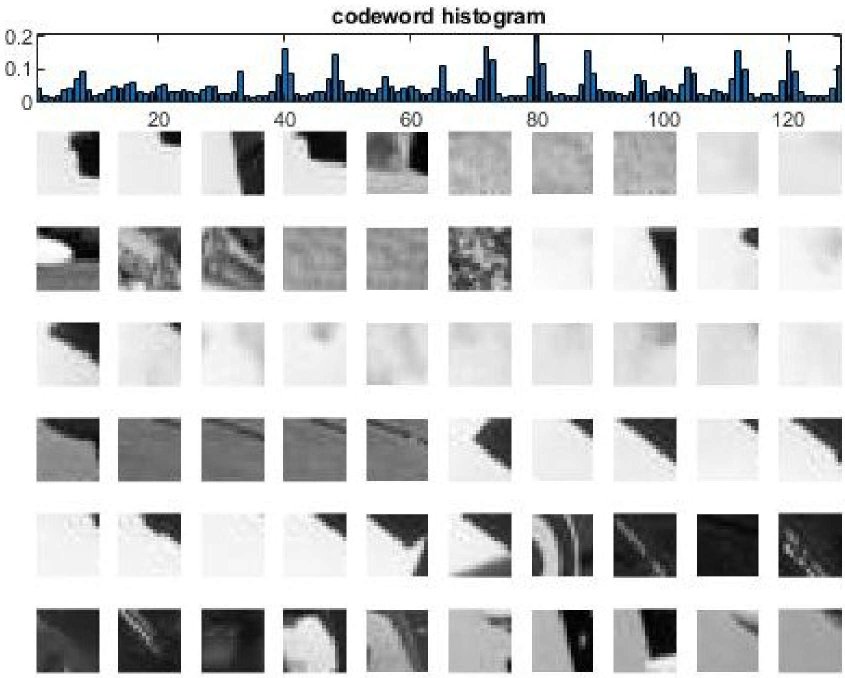


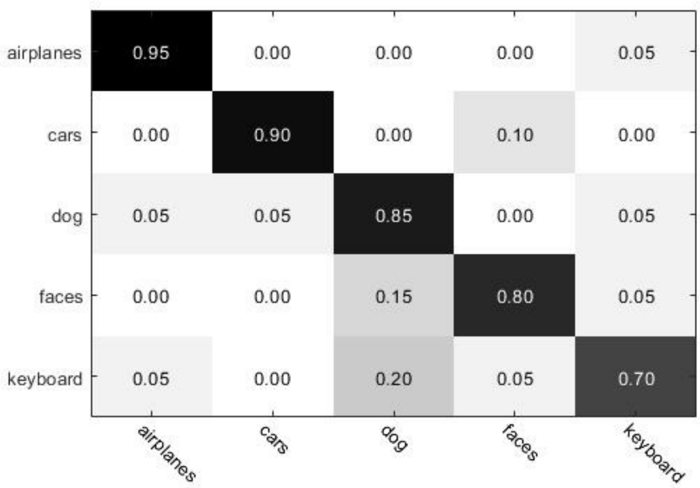
Fig 15: 30 train, 30 test image patches with codeid=18

<u>Support Vector Machine:</u>

Classification Errors and Confusion matrix

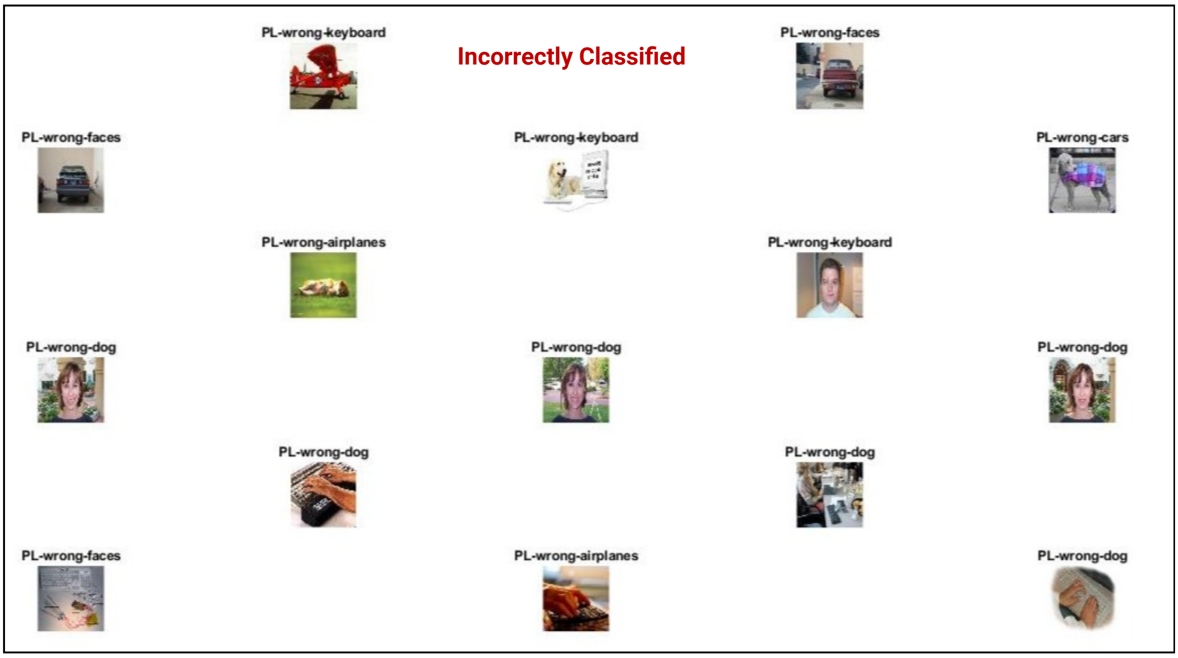
err_all = 0.1600 % Overall error rate
err = [0.0500, 0.1000, 0.1500, 0.2000, 0.3000] %Error per class

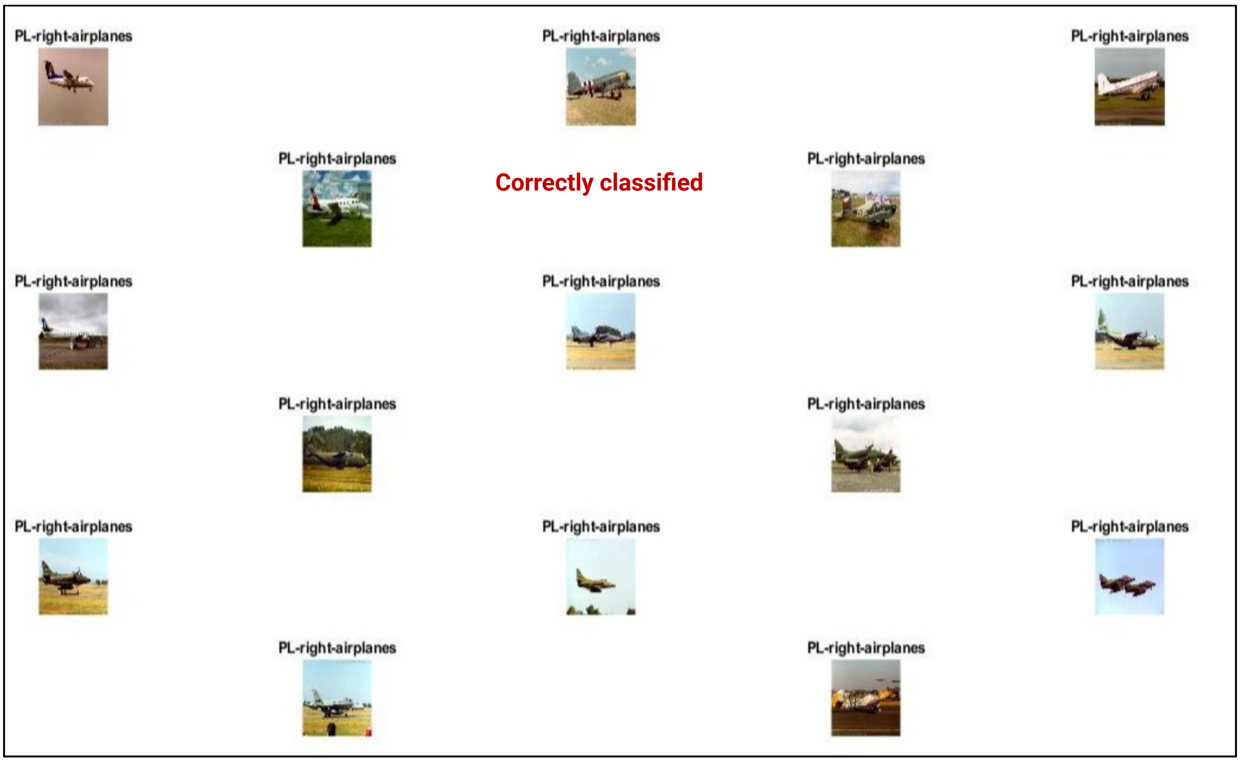
Classification using BOW rbf_svm
Accuracy = 84% (84/100)



Cara Evangeline Chandra Mohan
190539421

Correctly and incorrectly classified instances





<u>Reasons for failure:</u>

1) Requires Feature scaling
2) Requires Choosing of appropriate kernel function