Queen Mary
University of London

**School of Electronic Engineering and Computer Science**

**ECS797 Machine Learning for Visual Data Analysis**

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

# Introduction

The outcomes from the lab are to be handed in as a .zip file that contains a report and programs that show that you have completed the steps of the lab successfully. Details are given at the end of this sheet.

# 1. Getting Started

1. Create the subdirectories "ecs797" and "ecs797/lab1" in your workspace.
2. Download the file "ECS797Lab1.tar.gz" and extract the contents in the directory ecs797/lab1.
3. Check first there are three sub-folders and one main file:
    1. data- > contains the data in .mat files
    2. image -> contains 5 sub-directories each of which contains images from one of the following image classes: airplanes, cars, dog, faces and keyboard. In each class there are 80 images, the first 60 will be used for training and the rest will be used for testing.
    3. software-> contains dependency libs (libsvm-3.14, vlfeat-0.9.16) and Matlab codes.
    4. lab1.m -> is the main executable Matlab file
4. Start Matlab. Use "cd <directory>" to get into the directory "lab1" you have just created. In Matlab, load the "lab1.m" file in the matlab editor. You will need to edit the code in the file in order to complete the parts of the lab that are in red.
   *>> add software path to your workspace as the code of step 1.*

# 2. Dictionary creation – feature quantization

1. Executing commands in the file lab1.m load the pre-computed features for the training and the test images.
   %>> load('data/global/all_features');
   Check two variables, TrainMat (270000x128) and TestMat (90000x128), are loaded. The sift feature has 128 dimensions.
2. Create a dictionary by clustering a subset of the extracted descriptors. Use a dictionary of 500 words. Which parameter controls the size of the dictionary?%Run code in 2.2 and save the dictionary as coded:
   >>save('data/global/dictionary','C');
   The variable C , a 500x128 matrix will be saved in dictionary.mat .

3. Calculate the Euclidean distance between image descriptors and codewords (i.e. cluster centres)
   % The function is calculate in the file named *EuclideanDistance.m*
4. 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. The entry in the vector is the codeword ID. Save them like this: >>save('data/global/assigned_descriptor','index_train','index_test');
5. Visualise some image patches that are assigned to the same codeword.
   %The visualization requires the the assignment step (2.4) is already done, that is, that the index_train and index_test have been constructed (you may want to load them from the saved files). The function file is *visualize_patches.m*

# 3. Image representation using Bag of Words representation

1. 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.
   %Here we should read the feature data for each image which is stored in data/local/*ID*
   The example code is as follows:

```
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');
        d = EuclideanDistance(features.data, C);%calculate the distance
        %%%%calculate the histogram (that should be a vector with the
same dimensionality as the number of codewords and
normalize it
        %%%%BoW(ii,:) = histogram;
end
```

   To represent the images in training to the test dataset in the same matrix called BoW, of which the elements from 1 to 300 are representations for training images and elements from 301 to 400 are for test images. Therefore, the BoW should be 400x500 (if 500 words are used). Finally, save it as:
   >>save('data/global/bow','BoW')

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

1. Apply the steps of the NN classifier as described in the lab1.m script: calculate the L2 distances (i.e. Euclidean distances) between the test and the training images and assign to each test image the label of its nearest neighbour in the training test (i.e. assign each test image to the class of its nearest neighbour in the training set).
   %Try to use the function developed in 2.3. The main function file is *knnsearch.m.* The input arguments of this are :

test_data->TxD matrix , T test samples with D dimensional features
train_data->NxD matrix, N training samples with D dimensional features
k-> 'k' of KNN algorithm
method-> distance method option, 1-L2 distance, 2-histogram intersection
The output is:
NNresult->Txk matrix, the k-NNs for each of the T samples.

2. Compute and report the overall and the classification errors per class.
   %See code in 4.2.
3. Compute and show the confusion matrix.
   %See code in 4.3.
4. For each class show some images that are correctly classified and some images that are incorrectly classified. Can you explain some of the failures?
   %Set the number of which correctly/incorrectly images you are about to visualize.
5. Write code for computing the histogram intersection between two histograms.
   % Change the 'method' option in 4.1 and write the code for computing the histogram intersection in the file knnsearch.m
   Do the steps 1-4 above.

## 5. Dictionary size

1. 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).
2. Explain the drop in the performance. To support your argument you may want to perform step 2.5 in order to visualize descriptors that belong to some of the visual words.

## 6. Image classification using a Support Vector Machines classifier

1. For each of the image classes train a linear multiclass SVM classifier (using the libsvm library). Pay attention to the procedure that calculates the optimal values for the parameter C using cross validation.
2. Apply the linear SVM classifiers to the test images and classify them.
3. Compute and report the overall and the classification errors per class.
4. Compute and show the confusion matrix.
5. For each class show some images that are correctly classified and some images that are incorrectly classified. Can you explain some of the failures?

## 7. Handing In

Create a folder that will contain:

• A .pdf report that contains the answers to the exercises, the answers to specific questions, plots from experiments and program listings (including comments).
• The programs files

Create a .zip file and submit electronically.

**IMPORTANT:** Plagiarism (copying from other students, or copying the work of others without proper referencing) is cheating, and **will not be tolerated**.

**IF TWO "FOLDERS" ARE FOUND TO CONTAIN IDENTICAL MATERIAL, BOTH WILL BE GIVEN A MARK OF ZERO.**

## 8. Examination

You should be orally examined in one lab session after you have submitted your code and report. You should be prepared to explain the code that you wrote and critically evaluate the results that you obtained.