Deep Learning for Computer Vision
Homework 2

National Taiwan University

Due: April 11st Wed 01:00 AM Total: 110 points

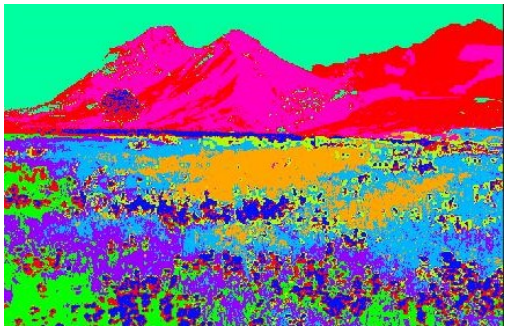
Homework policy

- Late policy: Up to 3 free late days in a semester. After that, late homework will be deducted 30% each day.
- Taking any unfair advantages over other class members (or letting anyone do so) is strictly prohibited. Violating university policy would result in F for this course.
- Students are encouraged to discuss the homework assignments, but you must complete the assignment by yourself. TA will compare the similarity of everyone's homework. Any form of cheating or plagiarism will not be tolerated, which will also result in F for students with such misconduct.

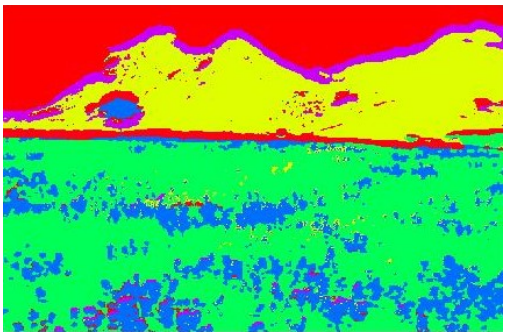
Download the file **HW2.zip (https://goo.gl/rzq8Lz)**. Please turn in your homework in PDF and submit it to Ceiba. Your file name should be **hw2_YourStudentID.pdf** (e.g., hw2_r06941111.pdf). Please **DO NOT** include any source code in your writing. We will **NOT** grade your homework if you don't follow the hand in format.

Problem 1: Kernel Trick (10%)

As noted in class, a (properly selected) kernel function can be expressed as an inner product between two data points which are projected onto a high-dimensional space, i.e., $K(\mathbf{x}, \mathbf{x}') = \Phi(\mathbf{x})^\top \Phi(\mathbf{x}')$, where K is the kernel function and $\Phi(\cdot)$ denotes feature mapping. Suppose that a mapping satisfies $\Phi(\mathbf{x}) : \mathbb{R}^2 \to \mathbb{R}^3$ with a corresponding kernel function $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^\top \mathbf{x}')^2$. Given $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^\top$, please derive the explicit form of $\Phi(\mathbf{x}) \in \mathbb{R}^3$ in terms of x_1, x_2.



(a) Color segmentation



(b) Texture segmentation

Figuur 1: Example results of (a) color segmentation and (b) texture segmentation.

Problem 2 : Color and Texture Segmentation (40%)

For this problem you will need to develop an unsupervised learning algorithm for image segmentation using either color or texture information.

Image data and filter banks

- **filterBank.mat**: The given .mat file contains a set of 38 filters (also known as filter bank). This filter bank is stored as a 49 x 49 x 38 matrix (i.e., each filter is of size 49 x 49 pixels).
- **Images**: **zebra.jpg** and **mountain.jpg**

Problem sets

(a) (20%) **Color segmentation**: Given an *RGB color image* of size $n = w \times h$ pixels, each pixel can be viewed as a three dimensional feature, each dimension describes the value in each color channel. For the task of image segmentation, please run the k-means algorithm* to cluster these n pixels (in terms of their 3D features) into k groups.
(i) Plot the segmentation results for both images based on your clustering results. For visualization purposes, pixels in the same group should be represented by the same color, while those in different groups are shown in distinct colors, as shown in Fig. 1a.
(ii) Convert both RGB images into Lab color space. Repeat the above clustering procedure and plot your segmentation results.
(*Please use $k = 10$ and maximum number of iterations = 1000 for both **zebra.jpg** and **mountain.jpg** when running the k-means algorithm.)

(b) (20%) **Texture segmentation**: We now consider the use of texture information for image segmentation. For simplicity, please convert the color images into *grayscale* ones, before extracting image textural features via the provided filter bank. As a result, you would produce $n = w \times h$ 38-dimensional features, each dimension corresponds to a particular filter response. Similarly, please perform k-means clustering* to cluster these n features into k different groups.
(i) Please plot the texture segmentation results for both images, as depicted in Fig. 1b.

(ii) Combine both color and texture features ($3 + 38 = 41$-dimensional features) for image segmentation. Repeat the above clustering procedure and plot your segmentation results.
(*Note that when calculating filter responses, please use symmetric padding to deal with pixels near image boundaries. Please use $k = 6$ and maximum number of iterations $= 1000$ for both **zebra.jpg** and **mountain.jpg**.)

Problem 3 : Recognition with Bag of Visual Words (60%)

For this problem you will implement a basic image-based bag-of-words (BoW) model for a scene image dataset with 5 categories. *You are encouraged* to use any related packages, libraries, and functions for your implementation. We will *not* grade on your source code, so please provide figure outputs with detailed discussions/explanations. You are also required to report the classification accuracy (no need to over-tune the parameters for training/testing).

Training/test data

The scene image database contain images of 5 categories: Coast, Forest, Highway, Mountain, and Suburb.

- **Train-10**: This dataset consists of 10 images $\times$ 5 categories = 50 images in total.
- **Train-100**: This dataset consists of 100 images $\times$ 5 categories = 500 images in total.
- **Test-100**: This dataset consists of 100 images $\times$ 5 categories = 500 images in total.

Problem sets

(a) (5%) Randomly pick an image from **Train-10**. Detect interest points and calculate their descriptors for this image using SURF. Plot your interest point detection results (e.g., image with the 30 most dominant interest points detected).

(b) (10%) Now you will learn a "dictionary" consisting of "visual words". Please extract the detected interest points from all of the 50 images in **Train-10**, and stack them into a $N \times d$ matrix, where N denotes the total number of interest points and d is the dimension of its descriptor. Use k-means algorithm to divide these interest points into C clusters (you may simply choose $C = 50$ and maximum number of iterations $= 5000$ for simplicity). The centroid of each cluster then indicates a visual word.
Construct the 3-dimensional PCA subspace from the above N interest points. Randomly select 6 clusters from the above results. Plot the visual words and the associated interest points in this PCA subspace. Please use the same color to denote projected visual words and interest points in your result.

(c) (20%) With the derived dictionary of visual words, you can now represent each training and test image as BoW features. When encoding the interest points into BoW, three different strategies will be considered: **Hard-Sum**, **Soft-Sum**, and **Soft-Max**, as we detail below.

Take an image with 4 interest points and a learned dictionary with 3 visual words ($C = 3$) for example. Table 1 lists the distance between each interest point f_i (with $i = 1, ..., 4$) and one of the centroids c_j (with $j = 1, 2, 3$). Each strategy for converting an image into a C-dimensional BoW feature is as follows:

Tabel 1: Distances between the interest points and each centroid. Note that 4 interest points and 3 centroids are considered.

	c_1	c_2	c_3
f_1	1	2	3
f_2	2	3	1
f_3	2	3	1
f_4	3	2	1

- **Hard-Sum**: For each interest point, a hard encoding is performed to obtain a C-dimensional BoW feature. For example, take f_1 in Table 1, its BoW feature would be [1 0 0]. Thus, a BoW of [1 0 3] would be observed.
- **Soft-Sum**: For each interest point, the reciprocal of its distance to each centroid would be normalized (i.e., the reciprocal of each entry in a row in Table 1 sums to 1). Then, a soft encoding is performed to obtain a C-dimensional BoW feature. For example, the BoW of the interest points would be [0.55 0.27 0.18] + [0.27 0.18 0.55] + [0.27 0.18 0.55] + [0.18 0.27 0.55] = [1.27 0.9 1.83] by this strategy.
- **Soft-Max**: For each interest point, the reciprocal of its distance to each centroid would be normalized (i.e., the reciprocal of each entry in a row in Table 1 sums to 1). Different from **Soft-Sum**, each attribute in the BoW is now determined by the maximum value of the soft-encoded features in that dimension (e.g., we obtain BoW as [0.55 0.27 0.55] in Table 1).

Note that you need to normalize your BoW features of **Hard-Sum** and **Soft-Sum** to avoid unequal numbers of descriptors in each image.

Now compute BoW of training images in **Train-10**, resulting in a $50 \times c$ matrix. Choose one image from each category and plot their **Hard-Sum**, **Soft-Sum**, and **Soft-Max**, respectively. Can you expect which BoW strategy results in better classification results and why?

(d) (25%) Finally, We adopt the k-nearest neighbors classifier (k-NN) to perform classification using the above BoW features.
(i) Use **Train-10** as the training data and **Test-100** for testing. Report the classification accuracy using **Hard-Sum**, **Soft-Sum**, and **Soft-Max**. Are the results as expected (based on your observation on different BoW features in (c))? If not, why?
(ii) Repeat (a) to (c) using **Train-100** as the training data. Do you observe improved classification results? Note that you might need to adjust parameters (e.g., the number C of clusters as visual words, and the number of k nearest neighbors in k-NN, etc.) for this experiment. Please report and explain your results.