Final Project Part 1
Bag-of-Words based Image Classification

Computer Vision 1
University of Amsterdam

Due 11:59pm, March 30, 2018

General guidelines

All your code and report must be handed in together in a zip file before the deadline (in Amsterdam time) by sending to **computervision1.uva(at)gmail.com**. For full credit, make sure your report follows these guidelines:

- Answer all given questions: briefly describe what you have done and include your code's outputs as figures in the report.
- Analyze the results and include your comments in the report, e.g. why algorithm A worked better than algorithm B in a certain problem. Reports comprised of sole results will decimate your grade.
- Try to understand the problem as much as you can. Give evidences (experimental results, references to papers, etc.) to support your arguments.
- All illustrations (tables, plots, images, etc.) must be accompanied by analysis and explanation. All tables and plots must have variables' names and units, axes' names and legends. All results that are not understandable will be rejected.
- Please express your thoughts in a compact fashion. After all, it is not how many words you use that tells how well you understood the concepts.

Late submissions are not allowed. Assignments that are submitted after the strict deadline will not be graded. In case of submission conflicts, TAs' system clock is taken as reference. We strongly recommend submitting well in advance, to avoid last minute system failure issues.
Plagiarism note: Keep in mind that plagiarism (submitted materials which are not your work) is a serious crime and any misconduct shall be punished with the university regulations.

1 Introduction

The goal of the assignment is to implement a system for image classification; in other words, this system should tell if there is an object of given class in an image. We will perform four-class (airplanes, motorbikes, faces, and cars) image classification based on bag-of-words approach. The provided data file contains training and test sub-directories for each category (all images are taken from Caltech Vision Group). For each class, test sub-directories contain 50 images, and training sub-directories contain up to 500 images.

1.1 Training Phase

Training must be conducted over a (training) dataset in which there exist at least 50 training images per class (take either the first 50 ones or a random subset of 50). Keep in mind that using more samples in training will (almost certainly) result in better performance. However, if your computational resources are limited and/or your system is slow, it's OK to use less number of training data to save time.

Hint

In order to debug your code use a small amount of input images/descriptors. Once you make sure everything works properly, you can run your code for the experiment using all the data points.

1.2 Testing Phase

You have to test your system using the specified subset of test images which will be supplemented separately. The instructions regarding how to use this set will be released together with the meta-data. (That is to ensure that we can verify your results while grading).

2 Bag-of-Words based Image Classification

Bag-of-Words based Image Classification system contains the following steps:

1. Feature extraction and description
2. Building a visual vocabulary
3. Quantize features using visual dictionary (encoding)
4. Representing images by frequencies of visual words
5. Classification

We will consider each step in detail.

2.1 Feature Extraction and Description

SIFT descriptors can be extracted from key points. As an alternative to key points, SIFT descriptors can be extracted densely. You can use VLFeat functions for dense SIFT (e.g. `vl_dsift`) and key points SIFT descriptor extraction (e.g. `vl_sift`). Moreover, it is expected that you implement all the steps also for RGBSIFT, rgbSIFT and opponentSIFT.

Hint

Check out the MATLAB API of VLFeat for further information in the following link: `http://www.vlfeat.org/matlab/matlab.html`

Use VLFeat functions in order to extract SIFT descriptors while working on RGBSIFT and etc.

2.2 Building Visual Vocabulary

Here, we will obtain visual words by clustering feature descriptors, so each cluster center is a visual word, as shown in Figure 1. Take a subset of all training images (this subset should contain images from ALL categories), extract SIFT descriptors from all of these images, and run k-means clustering (you can use your favourite k-means implementation) on these SIFT descriptors to build visual vocabulary. Take about half of the training images from each class to calculate visual dictionary (around 1000 images). Nonetheless, you can also use less images, say 100 from each class. Pre-defined cluster numbers will be the size of your vocabulary. Set the vocabulary size to 400, and experiment with several different sizes (800, 1600, 2000 and 4000).

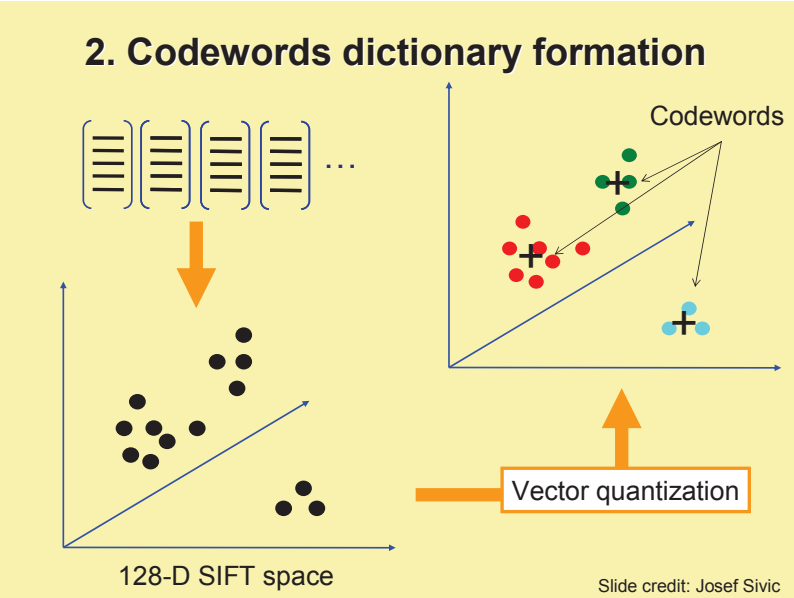

Figure 1: Learning Visual Dictionary. Code-words is another term for visual words.

Hint

Remember first to debug all the code with a small amount of input images and only when you are sure that code functions correctly run it for training over the larger data.

2.3 Quantize Features Using Visual Vocabulary

Once we have a visual vocabulary, we can represent each image as a collection of visual words. For this purpose, we need to extract feature descriptors (SIFT) and then assign each descriptor to the closest visual word from the vocabulary.

2.4 Representing images by frequencies of visual words

The next step is the quantization, the idea is to represent each image by a histogram of its visual words, see Figure 2 for overview. Check out MATLAB's `hist` function. Since different images can have different numbers of features, histograms should be normalized.

2.5 Classification

We will train a Support Vector Machine (SVM) classifier per each object class. As a result, we will have four binary classifiers. Take images from the training set of the related class (but the ones which you did not use for dictionary calculation). Represent them with histograms of visual words as discussed in the previous section. Use at least 50 training images per class (take either the first 50 ones or a random

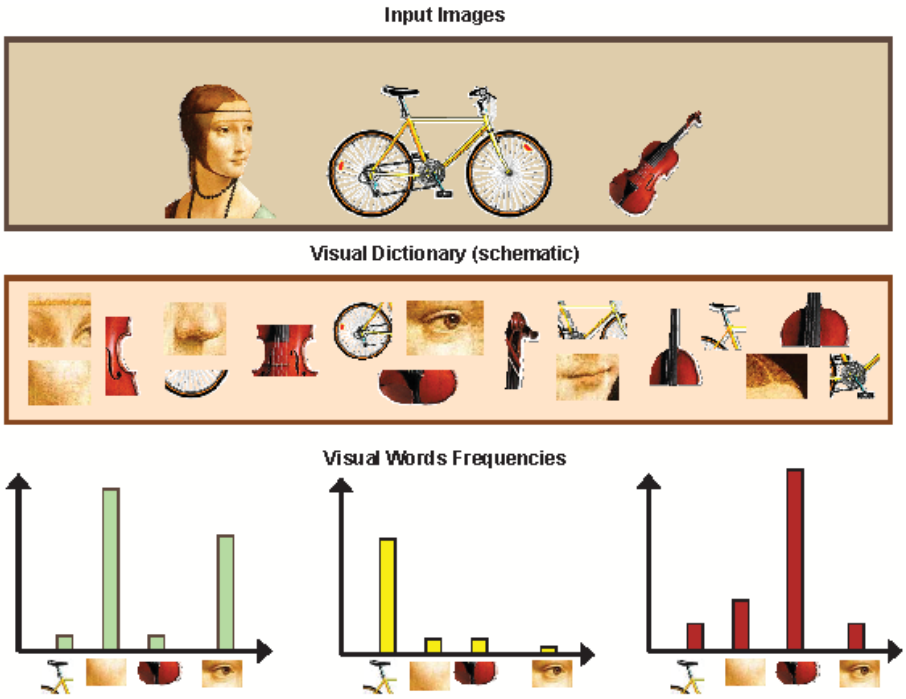

Figure 2: Schematic representation of Bag-Of-Words system.

subset of 50) or more, but remember to debug your code first! So, if you use the default setting, you should have 50 histograms of size 400. These will be your positive examples. Then, you will obtain histograms of visual words for images from other classes, again about 50 images per class, as negative examples. Therefore, you will have 150 negative examples. Now, you are ready to train a classifier. You should repeat it for each class. To classify a new image, you should calculate its visual words histogram as described in Section 2.4 and use the trained SVM classifier to assign it to the most probable object class. (Note that for proper SVM scores you need to use cross-validation to get a proper estimate of the SVM parameters. In this assignment, you do not have to do this cross-validation).

Hint

For the second part of the assignment we will use LIBLINEAR library `https://www.csie.ntu.edu.tw/~cjlin/liblinear/` to use support vector machines. So, it is better if you also use it for this part to get familiar with the library. Otherwise, you can use any other SVM library you like.

2.6 Evaluation

To evaluate your system, you should take all the test images from all classes and rank them based on each binary classifier. In other words, you should classify each test image with each classifier and then sort them based on the classification score. As a result, you will have four lists of test images. Ideally, you would have images

with airplanes on the top of your list which is created based on your airplane classifier, and images with cars on the top of your list which is created based on your car classifier, and so on.

In addition to the qualitative analysis, you should measure the performance of the system quantitatively with the Mean Average Precision over all classes. The Average Precision for a single class c is defines as

$$\frac{1}{m_c} \sum_{i=1}^{n} \frac{f_c(x_i)}{i} \ , \tag{1}$$

where n is the number of images ($n = 50 \times 4 = 200$), m is the number of images of class c ($m_c = 50$), x_i is the i^{th} image in the ranked list $X = \{x_1, x_2, \ldots, x_n\}$, and finally, f_c is a function which returns the number of images of class c in the first i images if x_i is of class c, and 0 otherwise. To illustrate, if we want to retrieve R and we get the following sequence: $[R, R, T, R, T, T, R]$, then $n = 7$, $m = 4$, and $AP(R, R, T, R, T, T, R) = \frac{1}{4}\left(\frac{1}{1} + \frac{2}{2} + \frac{0}{3} + \frac{3}{4} + \frac{0}{5} + \frac{0}{6} + \frac{4}{7}\right)$.

3 Deliverables (60 pts.)

1. Students are expected to prepare a report for this project. The report should include the following:

   - Analysis of the results for different settings such as key points vs dense sampling
   - mAP based on vocabulary size
   - mAP based on SIFT descriptor used (e.g., RGB-SIFT, rgb-SIFT and etc.)
   - mAP based on number of training samples used
   - mAP based on kernel choice for SVM (e.g. linear, RBF and etc.).

   Do not just provide numbers, remember to follow the general guidelines.

2. A demo function which runs the whole system should be prepared and submitted with all other implemented functions (excluding VLFeat and LIBSVM functions).

3. You are supposed to fill in the template HTML file, which is provided, with four ranked lists of test images as discussed in Section 2.6. This document should contain all your settings (e.g. size of visual vocabulary, number of positive and negative samples, and so on), Average Precision per class, and Mean Average Precision. You should submit a separate HTML for each descriptor type.

Hint

Having visual elements such as charts, graphs and plots are always useful for everyone. Keep this in mind while writing your reports.

4 Bonus (Max. 10 pts.)

We will award groups that prove to put extra efforts into their project. By *extra* we mean one or more of the following:

- **[2 pts.]** Using another classification approach (such as k-Nearest Neighbor). You can simply use the features you extracted in this project.
- **[4 pts.]** Using another type of features other than SIFT.
- **[Up to 10 pts.]** Any other thing that you can think of or find from the literature that can boost the results.

In order to get the bonus, your submission should first satisfy the previous steps. You need to implement and explain the method and justify your reasoning. You can get at most 10 points from bonus part.