CS 484, Fall 2018
Homework Assignment 2: Local Features

Due: November 26, 2018

1 Introduction

Grouping images into semantically meaningful categories is an important problem in computer vision. This problem has been traditionally tackled by using holistically computed low-level visual features such as color or texture histograms. Another popular approach is the bag-of-words (aka. bag-of-features, bag-of-keypoints) model for image representation. The goal of this homework assignment is to organize and visualize the content of an image archive by using such a representation based on local gradient and color features.

2 Background

The bag-of-words model for image representation will be discussed during the lecture but you should read the following papers to learn more about the model (the papers can be found on the course home page):

- G. Csurka, C. R. Dance, L. Fan, J. Willamowski, C. Bray, "Visual Categorization with Bags of Keypoints," European Conference on Computer Vision, 2004.
- L. Fei-Fei, P. Perona, "A Bayesian Hierarchical Model for Learning Natural Scene Categories," IEEE Conference on Computer Vision and Pattern Recognition, 2:524–531, June 20–25, 2005.

Once the bag-of-words representations are obtained for all images, we will use the t-SNE method to visualize the image archive. You should read the following paper to learn more about the visualization technique:

- L. J. P. van der Maaten and G. E. Hinton, "Visualizing High-Dimensional Data Using t-SNE," Journal of Machine Learning Research, 9(11):2579-2605, 2008.

3 Approach

The approach can be summarized in terms of the following steps.

3.1 Preparing data

Download the scene category data set for this assignment from the course home page. It includes a subset of the following data set:

- J. Xiao, K. Ehinger, J. Hays, A. Torralba, A. Oliva, "SUN Database: Exploring a Large Collection of Scene Categories," International Journal of Computer Vision, 119(1):3-22, August 2016.

The subset we will use contains 10 scene categories and 50 images in each category for a total of 500 images. You can see example images in Figure 1. Together with the image data, we provide three files:

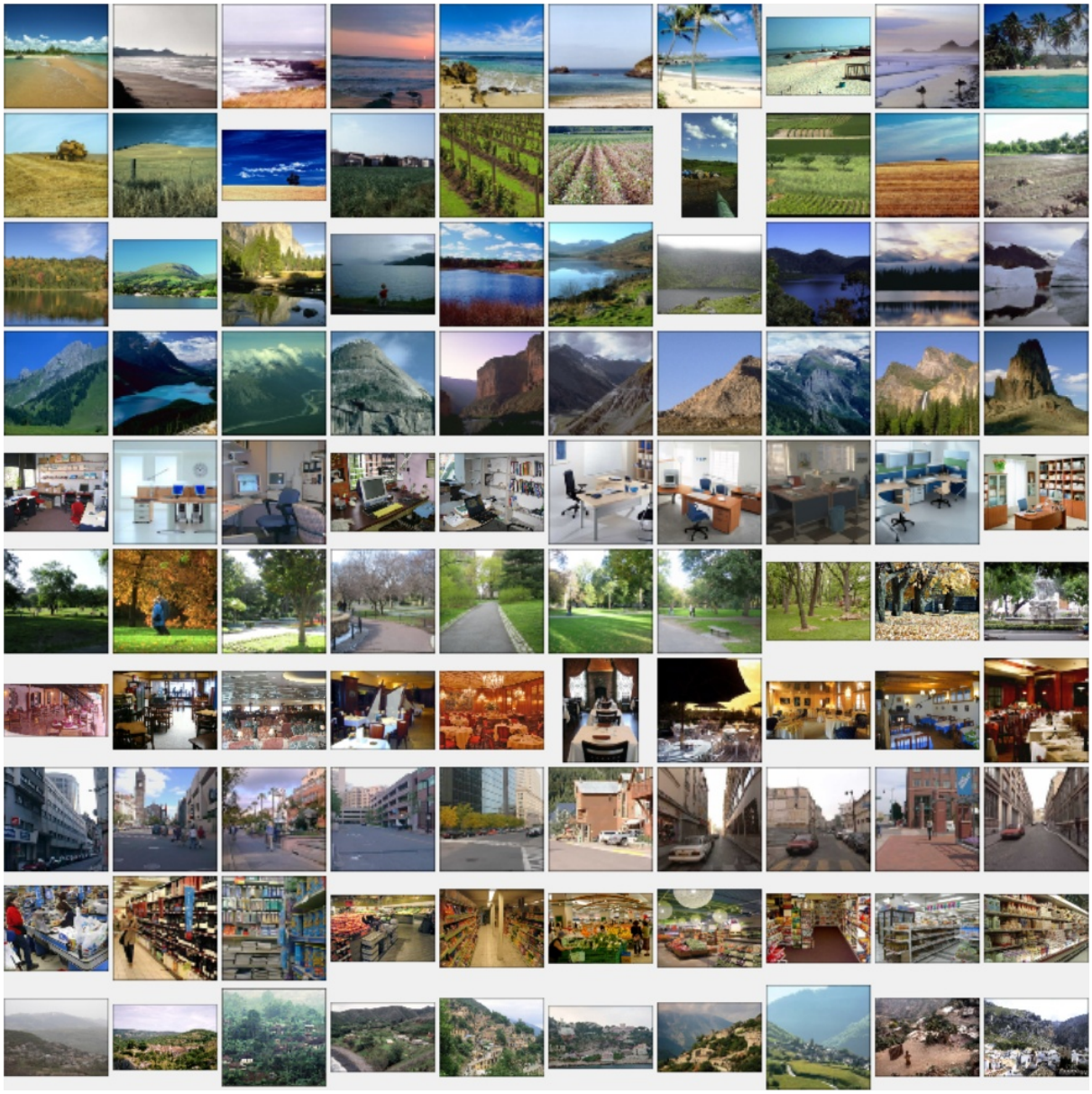

Figure 1: Examples from the scene category data set. Each row shows 10 images from a particular category. From top to bottom: beach, field, lake, mountain, office, park, restaurant, street, supermarket, and village.

- `files.txt`: Text file with 500 lines where each line contains the file name of one image.
- `classes.txt`: Text file with 500 lines where each line contains the category id of the corresponding image in `files.txt`. The categories are: 1 - beach, 2 - field, 3 - lake, 4 - mountain, 5 - office, 6 - park, 7 - restaurant, 8 - street, 9 - supermarket, and 10 - village.
- `classNames.txt`: Text file that contains the categories listed above.

3.2 Detecting local features

The first processing step is to run the detector of the Scale-Invariant Feature Transform (SIFT) on each image to obtain a set of interest points. The output of this step is, for each image, a set of keypoints represented with four numbers: x and y location, scale (for which the point is a local maximum), and orientation (the dominant orientation in the neighborhood of the point). Each image can contain a different number of interest points.

Note: You can use the VLFeat open source library (`http://www.vlfeat.org/`) for an implementation of the SIFT detector.

3.3 Describing local features

We will use two separate descriptors for each point:

- Gradient-based descriptors: compute the gradient-based descriptor of each local feature by using the descriptor step of the SIFT algorithm. Each point will have a SIFT descriptor of length 128.
- Color-based descriptors: compute a color histogram descriptor for each local feature. You should use the location, scale and orientation values computed in the previous step to form a scaled and rotated square around each point. Then, the color descriptor will be computed as a three-dimensional histogram of the RGB values of the pixels within the square neighborhood. You can use a $4 \times 4 \times 4$ histogram in the RGB space (i.e., each color channel is divided into 4 bins). The final one-dimensional descriptor can be obtained by *flattening* the three-dimensional histogram. Each point will have a color descriptor of length 64.

Note: You can use the VLFeat open source library (`http://www.vlfeat.org/`) for an implementation of the SIFT descriptor but the color-based descriptor step MUST BE your own implementation.

3.4 Finding visual words

The next step is the construction of the codebook of visual words. Here you can use the k-means clustering algorithm for quantizing the local descriptors. Typically, a large number of clusters (e.g, 500, 1000, 2000) has been used in the literature. You need to experiment with the number of clusters with respect to the number of local features obtained in the data.

In this step, we will compute three different codebooks:

- Using only the gradient-based descriptors (length of 128)
- Using only the color-based descriptors (length of 64)
- Using concatenated gradient-based and color-based descriptors (length of 192)

You must build two separate codebooks corresponding to two different sizes (i.e., two different values for number of clusters k) for each of the three options listed above. That is, you must have a total of six codebooks at the end of this step.

Note: Again, VLFeat and built-in MATLAB functions are good software resources for this step. If the set of all local descriptors that you extract does not fit into the memory of your computer (or if k-means is too slow), you may subsample them to run the k-means algorithm. Once you find the cluster centers, you can assign each local descriptor to the closest cluster center.

3.5 Building bag-of-words model

The next step is to compute the bag-of-words representation (i.e., histogram of visual words) for each image by using each of the six codebooks and the corresponding descriptors. At the end of this step, you will have six different bag-of-words representations for each image.

Optionally, you can ℓ_2 normalize each histogram by dividing the histogram by the norm of the histogram vector, in order to make image descriptors more robust against changes in the number of visual words (and some other factors). That is, for a bag-of-words histogram h, its ℓ_2-normalized version is given by $h/(\|h\| + 0.0001)$.

Note: This step MUST BE your own implementation.

3.6 Visualizing the data set

The final step is to use the t-SNE algorithm to map each image (by using its bag-of-words representation) to a point in a two-dimensional map for visualizing the whole data set. This algorithm is very popular for extracting the structure of the data in high dimensions. Once the images are mapped to the two-dimensional space, the resulting map can be visualized by coloring the points according to the class id of the corresponding image (like Figure 2(a) in the van der Maaten and Hinton paper). You can experiment with the parameters of the t-SNE algorithm to obtain good mappings. The result of this step contains six different visualizations (i.e., six separate figures), each one corresponding to one of the bag-of-words representations described above. Note that the category ids provided in the data set are only used for coloring the resulting mapping.

Note: You can use one of the t-SNE implementations given at `https://lvdmaaten.github.io/tsne/`.

3.7 Discussion

You must discuss your implementation choices at each step and how these choices affect the final result in a report. This discussion must include example outputs for each step, such as local features detected for at least one image per category, example gradient-based and color-based descriptors (shown as bar plots) for several points from these images, example bag-of-words representations (also shown as bar plots) for these images, and the final t-SNE visualizations (six different colored maps). You must report how different descriptors and codebooks affect the final maps, and discuss which categories are easier or more difficult to identify in the resulting maps.

3.8 Software

You must provide well-documented code that implements all steps described above. For the steps that use external libraries, you must provide a README file that explains which libraries are needed and how they can be obtained and installed.

You must have a specific entry point (e.g., a function or script) to your solution to this homework assignment with the inputs listed below:

- A text file that lists a set of file names (like `files.txt` as provided in the assignment data set),
- A text file that provides the category id of each image in this list (like `classes.txt` as provided in the assignment data set),
- An option for choosing a particular local feature descriptor (one of three options),
- An option for choosing the codebook size (one of two options for each descriptor).

The code for this entry point must be clearly described and documented in your solution. You are free to use any data structures and programming languages in your implementation. Note that we are going to test your code by using a separate image set that belongs to the same 10 categories.

Submit:

1. **A report (pdf file)** that contains the information requested above. You are also expected to provide a discussion of the results (e.g., which steps were easy and which were more difficult, what was possible and what was not).
2. **Well-documented code** for all steps that you implemented yourself. The specific entry point must be clearly indicated and documented.
3. Citations to all external resources that you used in your solution.

Make sure that all files you submit include your name and student ID.

Notes:
This assignment is due by midnight on Monday, November 26, 2018. You should upload your solutions as a **single archive file** that contains your **code** and **report** (a pdf file that contains the resulting plots, descriptions of how you obtained them, and discussion of the results) using the online submission form on the course web page before the deadline. Please see the course syllabus for a discussion of the late homework policy as well as academic integrity. If you have any questions about what is allowed and what is not allowed in a solution, please check the course syllabus on the course web page.