Lab Project Part 1
Image Classification using Bag-of-Words

Qi Bi and Weijie Wei

Due 11:59pm, October 24, 2021 (Amsterdam time)

General Guideline

- **Aim:**
  - Able to understand the basic Image Classification/Recognition pipeline using traditional Bag of Words method.
  - Able to use to python packages for image classification:*matplotlib, cv2, sklearn etc.*
- **Prerequisite:**
  - Familiar with Python and relevant packages.
  - Know the basics of feature descriptors(SIFT, HoG) and machine learning tools (K-means, SVM and etc.).
- **Guidelines:** Students should work on the assignments in a group of **three person** for two weeks. Some minor additions and changes might happen with approval from the Senior TA. Students will be informed for these changes via Canvas. Any questions regarding the assignment content can be discussed on Canvas Discussions. Students are expected to do this assignment in Python and PyTorch. However students are free to choose other tools (like TensorFlow). Your source code and report must be handed in together in a zip file (`ID1_ID2_ID3.zip`) before the deadline. Make sure your report follows these guidelines:
  - The maximum number of pages is 10 (single-column, including tables and figures). Please express your thoughts concisely.
  - Follow the given script and answer all questions (in green boxes). Briefly describe what you implemented. Blue boxes are providing you hints to answer questions.
  - Analyze your results and discuss them, *e.g.*why algorithm A works better than algorithm B on a certain problem.
  - Tables and figures must be accompanied by a brief description. Do not forget to add a number, a title, and if applicable name and unit of variables in a table, names and units of axes and legends in a figure.

  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.

Instruction

1. Students are expected to prepare a report for this project. The report should include the analysis of the results for different settings.

   Do not just provide numbers, remember to follow the general guidelines and discuss different settings.

2. For qualitative evaluation, you are expected to visualize the top-5 and the bottom-5 ranked test images (based on the classifier confidence for the target class) per setup. That means you are supposed to provide a figure for each experimental setup, as discussed in Section 2.6.

3. A demo function which runs the whole system should be prepared and submitted with all other implemented functions.

Hint

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

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. You will perform 5-class ($\{1 : airplanes, 2 : birds, 3 : ships, 4 : horses, 5 : cars\}$) image classification based on bag-of-words approach[1] using SIFT features, respectively. STL-10 dataset[2] will be used for the task. For each class, test sub-directories contain 800 images, and training sub-directories contain 500 images. Images are represented as (RGB) 96x96 pixels.

Hint

In a real scenario, the public data you use often deviates from your task. You need to figure it out and re-arrange the label as required using *stl10_input.py* as a reference.

Download the dataset from `http://ai.stanford.edu/~acoates/stl10/stl10_binary.tar.gz`. There are five files: *test_X.bin*, *test_y.bin*, *train_X.bin*,*train_y.bin* and *unlabeled_X.bin*. For the project, you will just use the train and test partitions. Download the dataset and make yourself familiar with it by figuring out which images and labels you need for the aforementioned 5 classes. Note that you do not need *fold_indices* variable.

1.1 Training Phase

Training must be conducted over the training set. Keep in mind that using more samples in training will likely 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

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

[1] `http://www.robots.ox.ac.uk/~az/icvss08_az_bow.pdf`
[2] `https://cs.stanford.edu/~acoates/stl10/`

Hint

You are not allowed to use the test images for training purpose.

1.2 Testing Phase

You have to test your system using the specified subset of test images. All 800 test images should be used at once for testing to observe the full performance. Again, exclude them from training for fair comparison.

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. Quantify features using visual dictionary (encoding)
4. Representing images by frequencies of visual words
5. Train the classifier

We will consider each step in detail.

2.1 Feature Extraction and Description

SIFT descriptors can be extracted from either (1) densely sampled regions or (2) key points. You can use SIFT related functions in *OpenCV* for feature extraction.

Feature Extraction (10-*pts*)

Extract SIFT descriptor from training datasets. Show two image from each of the five class (draw the circles with size of keypoint).

Hint1

Check out the Docs of SIFT and related functions for further information in the following links: `https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html` and `https://docs.opencv.org/3.4.9/d5/d3c/classcv_1_1xfeatures2d_1_1SIFT.html`

Note

For copyright reason, the newest version of OpenCV does not contain SIFT related function. However you can install an old version (for example: opencv-python==3.4.2.17 and opencv-contrib-python==3.4.2.17).

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 (maximum half) 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. Then, take the rest of the training images to calculate visual dictionary. Nonetheless, you can also use less images, say 100 from each class (exclusive from the previous subset) if your computational resources are limited. Pre-defined cluster numbers will be the size of your vocabulary. Set it to different sizes (500, 1000 and 2000).

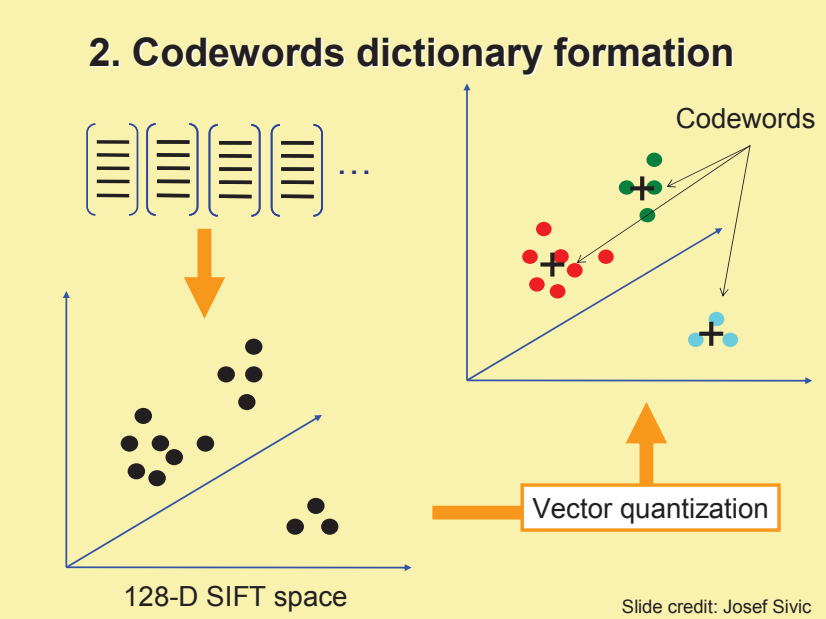

Figure 1: An illustration on learning visual dictionary. Note: (1) Code-words is another term for visual words. (2) The figure is from *Hosef Slvic*, with SIFT space used.

Building Visual Vocabulary (5-*pts*)

Create visual vocabulary by using K-means clustering. Remember to display the results in the size of 500, 1000 and 2000 respectively.

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. You can achieve K-means clustering using either *sklearn* package or *scipy* package.

2.3 Encoding 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 ***matplotlib***'s `hist` function. Since different images can have different numbers of features, histograms should be normalized.

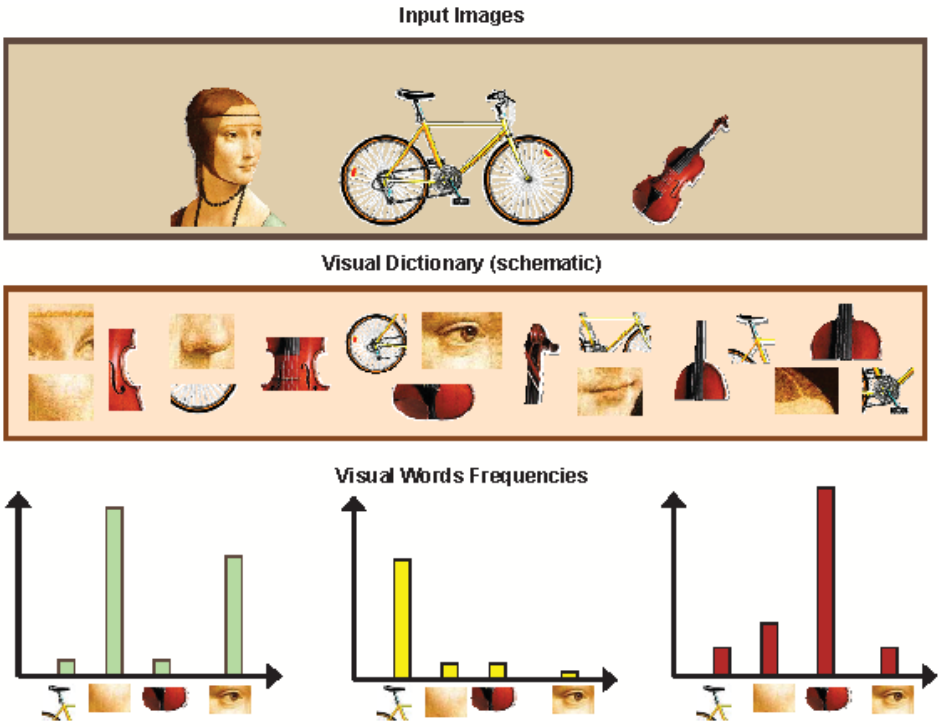

Figure 2: Schematic representation of Bag-Of-Words system.

Representing images by frequencies of visual words (5-*pts*)

Show the histogram of each class. Describe the similarities and differences.

2.5 Classification

We will train a classifier per each object class. Now, we take the Support Vector Machine (SVM) as an example. As a result, we will have 5 binary classifiers. Take images from the training set of the related class (should be 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 or more, but remember to debug your code first! If you use the default setting, you should have 50 histograms of size 500. 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 200 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 experiment with this cross-validation step).

Classification (5-*pts*)

Utilize SVM and finish classification training.

Hint

You can use *scikit-learn* software to conduct SVM classification. The relevant documents can be found at `https://scikit-learn.org/stable/modules/svm.html`

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 five 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 ($n = 50 \times 5 = 250$), 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: $[R, R, T, R, T, T, R, T]$, then $n = 8$, $m = 4$, and $AP(R, R, T, R, T, T, R) = \frac{1}{4}\left(\frac{1}{1} + \frac{2}{2} + \frac{0}{3} + \frac{3}{4} + \frac{0}{5} + \frac{0}{6} + \frac{4}{7} + \frac{0}{8}\right)$.

Evaluation and Discussion (35-*pts*)

Show the evaluation results and describe. For the qualitative evaluation, you are expected to visualize the top-5 and the bottom-5 ranked test images (based on the classifier confidence for the target class) per setup. The report should include the analysis of the results for different settings such as:

- mAP based on different vocabulary sizes (500, 1000, 2000).
- mAP based on SIFT descriptor and HoG descriptor.
- mAP based on classifiers of SVM.

Hint1

For SVM, hyper-parameters can be adjusted to find the optimal settings.

Hint2

Be sure to discuss the differences between different settings such as vocabulary sizes in your report.

Hint3

You can use *skimage.feature.hog* to extract HoG descriptor. The relevant documents can be found at `https://scikit-image.org/docs/dev/api/skimage.feature.html?highlight=hog#skimage.feature.hog`