# CS172 Computer Vision I:
# Project2: Image Classification based on BoF, SPM and SVM

Ziyun Zeng
2019533086
zengzy1@shanghaitech.edu.cn

## Abstract

In this paper, Bags of Features (BoF) is implemented for feature extraction and Support Vector Machine (SVM) is used for training is classification part. Moreover, Spatial Pyramids Matching (SPM) is implemented for better feature extraction. All the codes and experiments are based on Caltech 256 Dataset.

## 1. Problem Formulation

Object recognition is an important area in computer vision, and image classification is a task of it. Traditional recognition pipeline includes hand-designed feature extraction and trainable classifier, which extract information from image pixels and generate object class finally. In this paper, Bags of Features (BoF) is implemented for feature extraction and Support Vector Machine (SVM) is used for training is classification part. Moreover, Spatial Pyramids Matching (SPM) is implemented for better feature extraction.

## 2. Algorithm

### 2.1. Bag of Features

Bag of Features represents images as disorder collections of local features, referring the method of Bag of Words model. BoF has four steps
(1) Extract local features.
(2) Learn 'visual vocabulary' via K-means algorithm.
(3) Quantize local features using visual vocabulary via SPM.
(4) Represent images by frequencies of 'visual words'.

#### 2.1.1 Detection and Description: SIFT

BoF is based on features, so SIFT is a good algorithm helping local feature detection and description, which can extract different features invariant to image scale and rotation. There are four steps in generating the set of image features.
(1) Scale-space extrema detection.
(2) Keypoint localization.
(3) Orientation assignment.
(4) Keypoint descriptor.

#### 2.1.2 Codebook Generation

After getting the descriptors of keypoints of the images, BoF generates codebook of the visual words of size K by clustering those N descriptors. This is implemented using K-Means, a clustering algorithm that partition N vectors into K clusters using random initialization of K-center points, assigning each point to its nearest center, replacing the K centers with the mean of each center iteratively.

#### 2.1.3 Feature Quantization

BoF represents each image by the frequency of the appearance of visual words. Specifically, for each descriptor, it can be clustered int oone of the K key words. For each image, it can be quatized int othe distibution of the K keywords appearances, which form a histogram in implementation.

### 2.2. Spatial Pyramids Matching

Spatial Pyramids Matching (Spm) is based on BoF, with improvement on utilizing multi-levels spatial information of a singe image. This technique works by partitioning the image into increasingly fine subregions and computing histograms of local features found inside each sub-region.

### 2.3. Support Vector Machine

Support Vector Classifier is based on Support Vector Machine, a binary classification technique. Support Vector Machine uses a hyperplane that maximizes the margin between the positive and negative examples to classify data. SVC can be achieved in one-vs-one or one-vs-all SVMs, and the latter one has a better performance after doing experiment on a 32-classes sample of Caltech 256 Dataset.

## 3. Implementation

### 3.1. Code Repository

- data.py: generate train.txt and test.txt according to CLASS NUM and TRAIN SIZE. For a specific train,test. I will call it train 15.txt, if its TRAIN NUM is 15.
- codebook.py: Input the path images, write the model.txt and predict.txt
- classify.py: Use libsvm to train model.txt as a model and test predict.txt via the model
- train.txt: To record the path of images used for train.
- test.txt: To record the path of images used for test.
- model.txt: To record the features (BoF) for training the predict model
- predict.txt: To record the features (BoF) for testing the predict model
- 256 ObjectCategories: a repository containing the Caltech 256

### 3.2. How to run

- Download caltech 256.
- Set the CLASS NUM and TRAIN NUM in data.py and codebook.py respectively.
- Run data.py codebook.py classify.py in order.

### 3.3. Experiments: Problem encountered and Result

#### 3.3.1 BoF + Linear SVM on Caltech 256

The main part of BoF + Linear SVM is bagsoffeatures in codebook.py which have 6 args

- trainpath = train.txt
- modelpath = model.txt
- codebook: a list get from scipy.cluster.vq.kmeans
- length: the number of the quantity of images
- indexs: every images' label
- images: every images' list

I choose libsvm.train and libsvm.predict to train and predict the classification. When training, I change two default parameters. The first one is -t which means which kernel is used to exert libsvm.train, I choose -t 0 which means Linear SVM. The second one is -c which means the magnitude of the cost function C, I choose -c 10000 to keep the stability.

Setting codebook size K as 200, I got results below:

| Train Size | 15 | 30 | 45 | 60 |
|---|---|---|---|---|
| Accuracy | 16.9% | 18.4% | 19.2% | 20.1% |

#### 3.3.2 BoF + SPM + Linear SVM on Caltech 256

The main part of BoF + Linear SVM is bagsoffeatures in codebook.py which have 6 args

- trainpath = train.txt
- modelpath = model.txt
- codebook: a list get from scipy.cluster.vq.kmeans
- length: the number of the quantity of images
- indexs: every images' label
- images: every images' list

I choose libsvm.train and libsvm.predict to train and predict the classification. When training, I change two default parameters. The first one is -t which means which kernel is used to exert libsvm.train, I choose -t 0 which means Linear SVM. The second one is -c which means the magnitude of the cost function C, I choose -c 10000 to keep the stability.

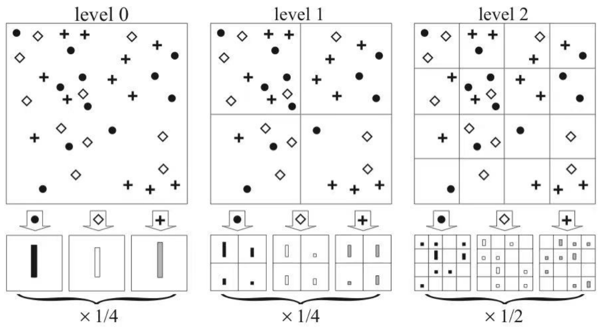

Figure 1. Discription of SPM

According to the SPM method, the BoF is 21 times larger than the BoF without using SPM method. So the time for calculating may be much longer than without using SPM. However, due to the pros of SPM, the

result shows that the accuracy rise about 30 % compared to the former method.[1]
Setting codebook size K as 200, I got results below:

| Train Size | 15 | 30 | 45 | 60 |
|---|---|---|---|---|
| Accuracy | 21.9% | 23.4% | 24.9% | 26.3% |

## 4. Problems and Solutions

### 4.1. Minibatch and K-Means++

Due to the large dataset, MiniBatchK-Means can be used in implementation to reduce the temporal and spatial cost. Because it uses small random batches of data of a fixed size to improve spatial performance and takes small randomly-chosen batches of the dataset in each iteration to improve temporal performance.

In some cases, it is to slow for a dataset to converge, so to reduce the temporal and spatial cost, K-Means++ can also be considered to use to converge faster.

### 4.2. Small pixel

In my implementation, I had tried two kinds of pixel 32*32 and 48*48. However,32*32 resolution ratio is too low that will make a large loss of features, so the accuracy is much lower than that of 48*48 resolution ratio. Moreover, through my rough estimate, rise the resolution ratio from 32*32 to 48*48 cost one more times time to calculate K-Means, for example it takes me 65 minutes to calculate all the data for K=200,CLASSNUM=256,TRAINNUM=60. If rising resolution ratio much higher, it may cost me several hours for a test.

In conclusion, rising resolution ratio can be a really efficient way to improve the accuracy.

### 4.3. Small K

As the former problem, they are all because of the insufficiency of my conputer performance. If choose a larger K, the BoF for a image set will be more detailed than before, so it can be important part to lead to a high accuracy.

## References

[1] Svetlana lazebnik, cordelia schmid, and jean ponce"beyond bags of features: Spatial pyramid matching for recognizing natural scene categories"in: 2 (2006),pp. 2169–2178.