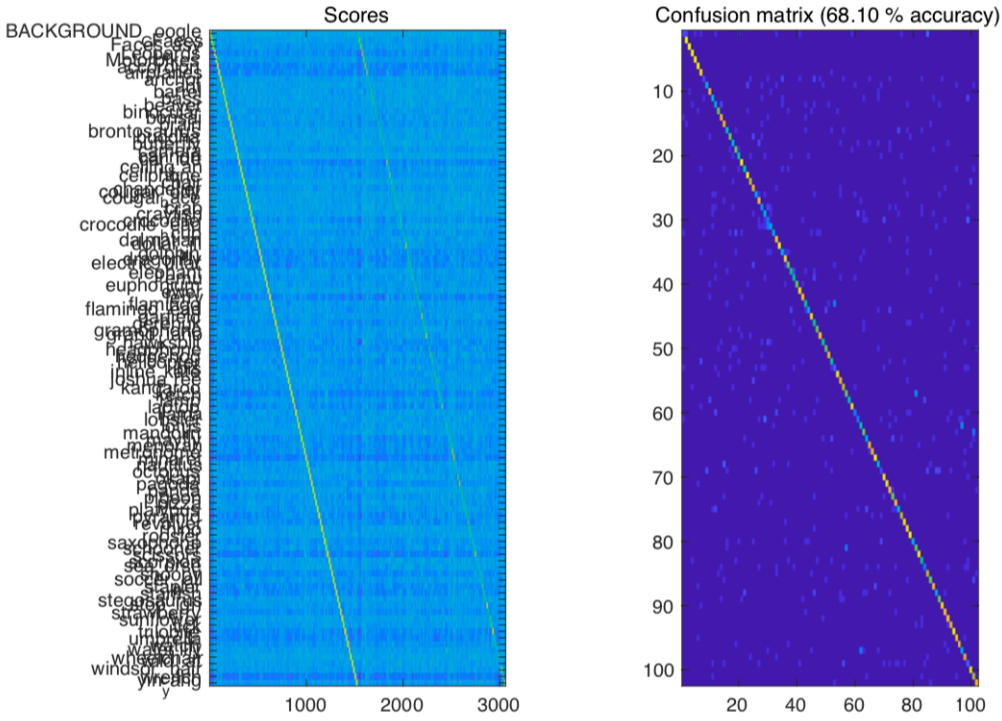
Computer Vision Final Project

Using Spatial pyramid Caltech101 test accuracy

소프트웨어학과 201723274
이서현

- **Result Summary**



<사진 1. 결과 Score와 Confusion matrix>

이번 Bag of Word project에서 test 정확성을 67.58%에서 68.10%로 upgrade했다. 사진 1의 confusion matrix를 보면 68.10% accuracy를 나타낸다. Confusion matrix는 간단하게 supervised training algorithm의 성능을 시각화 할 수 있는 표이다. 파란색에 가까울 수록 정확성이 높은 것이고 빨간색에 가까울 수록 정확성이 낮은 것이다.

성능을 높이기 위해 **Spatial Pyramid Matching**을 이용해서 input image을 n 등분하여 영역마다 histogram을 따로 계산하여 spatial information을 더하고자 했다. 또한 **k-means clustering**에서 k 의 값을 구하기 위해 다양한 k 값을 대입해보았다. Input image를 절반으로 나눠서 각자의 histogram을 독립적으로 계산해서 spatial information을 추가했다. K값은 30, 50, 100, 300을 대입해보았고 k=100에서 가장 높은 accuracy가 나타났다.

- **Table of content**

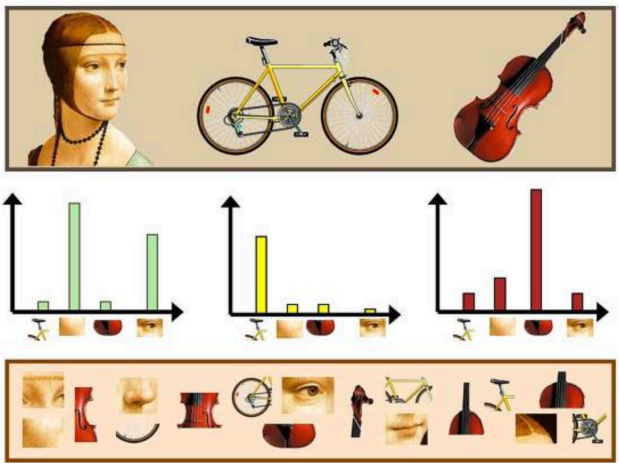
1. Bag of words에 대한 이론

1.1 Image categorization

Image를 category로 나누기 위해서는 training과 testing 파트로 나눠진다. Training에서는 training image가 입력되면 image의 feature를 추출한다. 추출된 image feature와 입력된 training labels를 이용해서 classifier training을 한다. 그 결과 trained된 classifier가 나오게 된다. Testing 파트는 image의 feature를 추출하고 trained된 classifier를 이용해서 prediction을 출력한다.

1.2 Bag of words

영상에서 feature을 추출한 뒤 feature를 representation할 수 있는 값인 code로 이루어진 codebook을 만든다. Codebook은 많은 image에서 추출한 feature들에 대해 k-means clustering을 수행하고 여기서 만들어진 각 cluster center인 대표 feature로 구성된다. Codebook으로 각각의 image를 representation할 수 있다. Codebook의 codeword로 image의 특성을 histogram으로 표현할 수 있다. Histogram에는 codeword가 image에 나타난 개수를 센 것이다. 요약하자면 Bag of words는 사진 2처럼 histogram이 동일한 image에서는 비슷하게 나타나고 다른 image에서는 다른 histogram이 나타난다.



<사진 2. Bag of words>

1.3 Spatial pyramid matching

Spatial pyramid matching은 Bag of words의 accuracy를 높이기 위한 방법이다. Histogram은 동일하나 입력된 영상은 다른 것을 나타내는 단점을 보완한다. Input

image에 level에 따라 image를 나눠서 나눈 영역에 대한 histogram을 각자 구해주는 것이다. 즉 서로 독립적인 spatial bin에 따라 histpgram을 구해주는 것이다. 영역을 나눠주어서 histogram에는 spatial information이 더해진다.

2. Source code 해석

2.1 phow_caltech101.m

```
%configure 을 정의한다.
conf.calDir = 'data/caltech-101' ; %image data가 있는 위치
conf.dataDir = 'data/' ;
conf.resultDir = 'result/';  %result data가 저장될 위치
conf.autoDownloadData = true ;

conf.numTrain = 15 ; % train image 수 = 15개
conf.numTest = 15 ; % test image 수 = 15개
conf.numClasses = 102 ; % 이미지를 가져올 폴더 수 102개 (카테고리)

conf.vocabPath = fullfile(conf.resultDir, [conf.prefix '-vocab.mat']) ;
% vocabulary data 저장 위치
conf.histPath = fullfile(conf.resultDir, [conf.prefix '-hists.mat']) ;
% histogram data 저장
conf.modelPath = fullfile(conf.resultDir, [conf.prefix '-model.mat']) ;
% model 정보 저장
conf.resultPath = fullfile(conf.resultDir, [conf.prefix '-result']) ;
% 결과 저장
% --------------------------------------------------------------------
%                                                 Download Caltech-101 data
% --------------------------------------------------------------------


% --------------------------------------------------------------------
%  디렉토리에서 이미지를 불러온다.                                  Setup data
% --------------------------------------------------------------------
 classes = dir(conf.calDir) ;
 classes = classes([classes.isdir]) ;
 classes = {classes(3:conf.numClasses+2).name} ;
 %이미지 디렉토리에서 첫 번째부터 세 번째까지 디렉토리를 classes에 넣는다.

 images = {} ; % 모든 이미지들의 인덱스가 들어간다.
 imageClass = {} ; % 모든 이미지의 카테고리가 들어간다
for ci = 1:length(classes)
   ims = dir(fullfile(conf.calDir, classes{ci}, '*.jpg'))' ;
   ims = vl_colsubset(ims, conf.numTrain + conf.numTest) ;
   ims = cellfun(@(x)fullfile(classes{ci},x),{ims.name},'UniformOutput',false) ;
   images = {images{:}, ims{:}} ;
   imageClass{end+1} = ci * ones(1,length(ims)) ;
end
 selTrain = find(mod(0:length(images)-1, conf.numTrain+conf.numTest) < conf.numTrain) ;
 % 16~30...selTest 인덱스를 제외한 15*102개의 이미지의 인덱스가 들어간다.
 selTest = setdiff(1:length(images), selTrain) ;
 % 16~30 인덱스 15개의 테스트 이미지의 인덱스가 들어간다.
% --------------------------------------------------------------------
%  vocab 만드는 과정                                          Train vocabulary
% --------------------------------------------------------------------
```

```
% vl_phow() 함수를 사용해서 descre를 추출했다. 특징점을 BoVW를 만들기 위해 kmeans 클러스터링을 해야한다.
% k-means 클러스터링이란 k개의 센터를 정하고 센터에서 가까운 데이터들을 하나의 집합으로 만드는 것

if ~exist(conf.vocabPath) || conf.clobber

  % Get some PHOW descriptors to train the dictionary
  selTrainFeats = vl_colsubset(selTrain, 100) ;
  % x(selfTrain)가 n(30)의 개수보다 작으면 랜덤이 아닌 컬럼을 그대로 반영하지만
  % x의 컬럼이 더 많다면 랜덤으로 n의 개수로 랜덤 반영한다.
  % 15개 중 30개를 특징점 train image로 뽑았다.
  descrs = {} ;
  % 128 x K matrix의 descriptors를 리턴 받을 장소

  %for ii = 1:length(selTrainFeats)
  parfor ii = 1:length(selTrainFeats)
    im = imread(fullfile(conf.calDir, images{selTrainFeats(ii)})) ;
    im = standarizeImage(im) ;
    [drop, descrs{ii}] = vl_phow(im, model.phowOpts{:}) ;
  end
  % 30개의 이미지 인덱스를 사용해 im에 이미지를 하나하나 불러온 후 standarizeImage()함수로
  % 이미지를 같은 타입으로 만든다. (vfleat는 single form의 이미지를 요구함)
  % 그 후 vl_phow 함수를 사용해서 model.phowOpts{:} 옵션으로 im에 있는 이미지의
  % features(frames로 구성-frames(1:2,:)는 특징점의 x,y좌표 (센터 값)) 와 descriptor를 구함

  descrs = vl_colsubset(cat(2, descrs{:}), 10e4) ;
  descrs = single(descrs) ;

  % Quantize the descriptors to get the visual words
  vocab = vl_kmeans(descrs, conf.numWords, 'verbose', 'algorithm', 'elkan', 'MaxNumIterations', 50) ;
  % vl_kmeans 함수에 입력데이터 descre를 넣고, conf.numWords으로 클러스터링하고 elkan알고리즘을 사용해서
  % vocab에 저장
  save(conf.vocabPath, 'vocab') ;
  % 코드북(visual words) 만들기 위해 quantization한다.
  % vl_kmeans 함수를 사용해 센터값을 구한다.
if strcmp(model.quantizer, 'kdtree')
  model.kdtree = vl_kdtreebuild(vocab) ;
end
  % vl_kdtreebuild(x) x 데이터의 인덱스로 kd-tree forest를 만든다.

% --------------------------------------------------------------------
%  histogram을 만드는 과정      Compute spatial histograms
% --------------------------------------------------------------------
 if ~exist(conf.histPath) || conf.clobber
   hists = {} ; % histogram을 저장할 곳

   parfor ii = 1:length(images) % 모든 카테고리 images
   % for ii = 1:length(images)
     fprintf('Processing %s (%.2f %%)\n', images{ii}, 100 * ii / length(images)) ;
     im = imread(fullfile(conf.calDir, images{ii})) ;
     hists{ii} = getImageDescriptor(model, im);
   end
   hists = cat(2, hists{:}) ;

   save(conf.histPath, 'hists') ;
 else
   load(conf.histPath) ;
 end
% --------------------------------------------------------------------
% hists로 커널 맵 구성 (linear SVM solver PEGASOS를 train하기 위해)   Compute feature map
% --------------------------------------------------------------------

psix = vl_homkermap(hists, 1, 'kchi2', 'gamma', .5) ;
```

2.1 getImageDescriptor.m

```
% get PHOW features
 [frames, descrs] = vl_phow( im, model.phowOpts{:}) ;
% 이미지 im의 f,d 키포인트의 위치와 속성을 나타낸다. decres는 4x4x18 방향을 나타낸다.
```

```
% i의 범위를 image를 반으로 자른 길이로 설정하여
% image의 histogram을 나눈 영역별로 따로 계산한다
% spatial pyramid를 구현하였다.
for i = 1:length(model.numSpatialX)/2
   binsx = vl_binsearch(linspace(1,width,model.numSpatialX(i)+1), frames(1,:)) ;
   binsy = vl_binsearch(linspace(1,height,model.numSpatialY(i)+1), frames(2,:)) ;

   % combined quantization
   bins = sub2ind([model.numSpatialY(i), model.numSpatialX(i), numWords], ...
                  binsy,binsx,binsa) ;
   hist = zeros(model.numSpatialY(i) * model.numSpatialX(i) * numWords, 1) ;
   hist = vl_binsum(hist, ones(size(bins)), bins) ;
   hists{i} = single(hist / sum(hist)) ;
end

for i = length(model.numSpatialX)/2:length(model.numSpatialX)
   binsx = vl_binsearch(linspace(1,width,model.numSpatialX(i)+1), frames(1,:)) ;
   binsy = vl_binsearch(linspace(1,height,model.numSpatialY(i)+1), frames(2,:)) ;

   % combined quantization
   bins = sub2ind([model.numSpatialY(i), model.numSpatialX(i), numWords], ...
                  binsy,binsx,binsa) ;
   hist = zeros(model.numSpatialY(i) * model.numSpatialX(i) * numWords, 1) ;
   hist = vl_binsum(hist, ones(size(bins)), bins) ;
   hists{i} = single(hist / sum(hist)) ;
end
hist = cat(1,hists{:}) ;
hist = hist / sum(hist) ;
```

2.2 standarizeImage.m

```
% 이미지 type을 single로 변경
% 이미지 double을 single로 만든다.
% 이미지의 크기가 480을 넘으면 resize 한다.
```

3. Result

```
Caltech101 test accuracy = 68.10%
경과 시간은 462.831071초입니다.
```

<사진 3. 출력값>

SIFT algorithm을 이용한 결과값은 67.58%로 출력되었다. SIFT algorithm보다 accuracy를 높이기 위해서 spatial pyramid matching 방법을 구현하는 것에 중점을 두었다. Spatial image는 하나의 input image가 histogram을 계산했을 때 다른 image와 histogram와 똑 같을 문제을 보완한다. 각 영역을 나눠서 각자의 bin에 따른 histogram을 계산하는 것이다. Image를 나누기 위해 getImageDiscriptor.m에 코드를 추가해보았다. phow_caltect101.m에서 image를 받아오면 하나의 image의 전체적인 histogram을 계산하는데 이 때는 spatial information이 없기 때문에 정확성이 낮다.

```
% i의 범위를 image를 반으로 자른 길이로 설정하여
% image의 histogram을 나눈 영역별로 따로 계산한다
% spatial pyramid를 구현하였다.
for i = 1:length(model.numSpatialX)/2
  binsx = vl_binsearch(linspace(1,width,model.numSpatialX(i)+1), frames(1,:)) ;
  binsy = vl_binsearch(linspace(1,height,model.numSpatialY(i)+1), frames(2,:)) ;

  % combined quantization
  bins = sub2ind([model.numSpatialY(i), model.numSpatialX(i), numWords], ...
                 binsy,binsx,binsa) ;
  hist = zeros(model.numSpatialY(i) * model.numSpatialX(i) * numWords, 1) ;
  hist = vl_binsum(hist, ones(size(bins)), bins) ;
  hists{i} = single(hist / sum(hist)) ;
end

for i = length(model.numSpatialX)/2:length(model.numSpatialX)
  binsx = vl_binsearch(linspace(1,width,model.numSpatialX(i)+1), frames(1,:)) ;
  binsy = vl_binsearch(linspace(1,height,model.numSpatialY(i)+1), frames(2,:)) ;

  % combined quantization
  bins = sub2ind([model.numSpatialY(i), model.numSpatialX(i), numWords], ...
                 binsy,binsx,binsa) ;
  hist = zeros(model.numSpatialY(i) * model.numSpatialX(i) * numWords, 1) ;
  hist = vl_binsum(hist, ones(size(bins)), bins) ;
  hists{i} = single(hist / sum(hist)) ;
end
hist = cat(1,hists{:}) ;
hist = hist / sum(hist) ;
```

<사진 4. getImageDiscrptor.m의 일부분 >

Spatial pyramind를 구현하기 위해 Image의 size를 반으로 나눠서 각각 i의 범위를 설정해주었다. Image를 반으로 나눠 feature를 추출하고 각각 영역에 따른 독립적인 histogram을 계산한다. 즉, 각자 spatial bin에 따른 histogram을 계산하는 것이다. 이와 같이 spatial information을 주었더니 accuracy가 상승하는 것을 관찰할 수 있었다.

`selTrainFeats = vl_colsubset(selTrain, 100) ;`

위 코드는 `phow_caltech101.m`의 `Train vocabulary` 영역이다. selTrain의 개수가 100보다 작으면 랜덤이 아닌 column을 그대로 반영하지만 selTrain의 개수가 더 많다면 랜덤으로 수를 뽑아서 특징점의 개수를 설정한다. 즉 k-means algorithm의 k의 값을 설정하는 것이다. 30, 50, 100, 300을 넣어봤지만 100이 가장 정확도가 좋았다.

4. 보완할 점

```
다음 사용 중 오류가 발생함: getImageDescriptor (line 38)
배열 인덱스는 양의 정수이거나 논리값이어야 합니다.

오류 발생: phow_caltech101 (line 175)
  parfor ii = 1:length(images) % 모든 카테고리 images

오류 발생: run_caltech (line 6)
acc = phow_caltech101();
```

<사진 5. getImageDiscriptor.m 오류>

Spatial Pyramid에서 level1인 image를 4등분하는 코드를 구현해보고 싶었지만, 사진 5와 같은 오류가 계속 발생하였다. 4등분으로 정확도를 크게 높여보고 싶었지만 배열 인덱스의 적절한 값을 넣지 못해 발생하는 오류를 잡아내지 못해서 아쉬웠다. Image를 어떻게 분리하여 spatial pyramid의 성능을 극대화시킬 수 있는지 고민해보고 성능이 좋아지는 다른 방법도 생각해볼 것이다.

5. 참고

- Computer Vision: Algorithms and Applications <Richard Szeliski>
- Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories <Svetlana Lazebnik1, Cordelia Schmid2, Jean Ponce1,3>
- Bag of Words 기법 : https://darkpgmr.tistory.com/125
- k-meanns algorithm: https://ratsgo.github.io/machine%20learning/2017/04/19/KC/