Homework 4 Writeup

Instructions

- Describe any interesting decisions you made to write your algorithm.
- Show and discuss the results of your algorithm.
- Feel free to include code snippets, images, and equations.
- Use as many pages as you need, but err on the short side If you feel you only need to write a short amount to meet the brief, th
- **Please make this document anonymous.**

Files

1. get_tiny_image.m

   input : image_path
   output : image_feats

2. nearest_neighbor_classify.m

   input : train_image_feats, train_labels, test_image_feats
   output : predicted_categories

   1. and 2. are the first section.

3. bulid_vocabulary.m

   input : image_paths, vocab_size
   output : vocab

4. get_bags_of_words.m

   input : image_paths
   output : image_feats

   2. and 3.&4. are the second section.

5. svm_classify.m

   input : train_image_feats, train_labels, test_image_feats
   output : predicted_categories

   5. and 3.&4. are the last section.

Code

1. get_tiny_image.m

```
N = size(image_paths,1);
image_feats = zeros(N,256);
for i = 1:N
    resize_im = imresize(imread(image_paths{i}),[16 16]);
    image_feats(i,:) = reshape(resize_im, 1,256);
    image_feats(i,:) = image_feats(i,:) - mean(
       image_feats(i,:));
    image_feats(i,:) = image_feats(i,:)./norm(image_feats
       (i,:));
end
```

This function resizes the size of the image. Afterwards, the image is corrected by dividing it into norm.

2. nearest_neighbor_classify.m

```
N1 = size(test_image_feats,1);
D = pdist2(train_image_feats,test_image_feats);
k = 15;
label = unique(train_labels);
size_l = size(label,1);
[~, index] = sort(D,1);
count = zeros(size_l,N1);
for i = 1:N1
    for j = 1:size_l
        m = train_labels(index(1:k,i));
        count(j,i) = sum(strcmp(label(j),m));
    end
end
[~, index_l] = max(count,[],1);
predicted_categories = label(index_l);
end
```

It is first method to categorize the images. Nearest neighbor classifier.

3. bulid_vocabulary.m

```
N1 = size(image_paths,1);
feat = [];
for i = 1:N1
    I1 = im2single(imread(image_paths{i}));
    SURF = detectSURFFeatures(I1);
    strongest = SURF.selectStrongest(200);
    [features,v] = extractHOGFeatures(I1,strongest,'
       CellSize',[48 48]);
    size_f = size(features,1);
    rand1 = randperm(size_f);
    sample = rand1(1:round(size_f/5));
    other = features(sample,:);
    feat = vertcat(feat,other);
end
[idx,C] = kmeans((feat),vocab_size);
vocab = C;
```

Catch the feature with the fuction extractHOGFeatures. It makes vocab.mat to find feature at get_bags_of_words.m.

4. get_bags_of_words.m

```
load('vocab.mat')
size_v = size(vocab, 1);
N = size(image_paths, 1);
image_feats = zeros(N, size_v);
for i=1:N
    I1 = im2single(imread(image_paths{i}));
    SURF = detectSURFFeatures(I1);
    strongest = SURF.selectStrongest(200);
    [features,v] = extractHOGFeatures(I1,strongest,'
       CellSize',[48 48]);
    temp = zeros(size_v,1);
    index = knnsearch(vocab,features,'K',23);
    size_i=size(index,1);
    for j=1:size_i
        for k=1:10
            temp(index(j,k)) = temp(index(j,k)) +1;
        end
    end
    hist = temp./norm(temp);
    image_feats(i,:) = hist;
end
```

Categorize the images with knn algorithm.
I find K value with experiment. $(K = 23)$

5. svm_classify.m

```
categories = unique(train_labels);
num_categories = length(categories);

scores = [];
for i=1:num_categories
    match = strcmp(categories{i},train_labels);

    % fitcl =fitclinear(train_image_feats,match');
    fitcl = fitcsvm(train_image_feats,match');
    W = fitcl.Beta';
    B = fitcl.Bias;
    predict = W*test_image_feats'+B;
    scores = [scores; predict];
end

[M,I] = max(scores);
predicted_categories = categories(I);
end
```

Categorize the images with svm algorithm.

Result

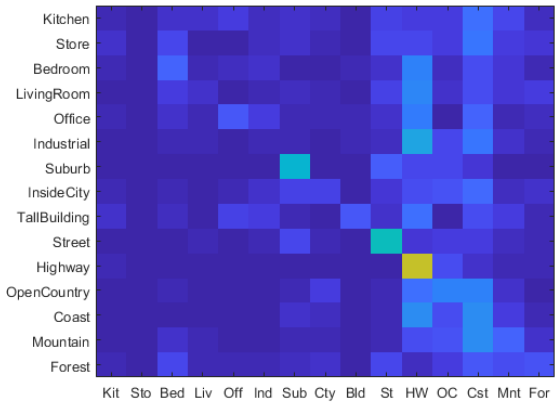

Figure 1: tiny image&nearest neighbor

section 1.
FEATURE = 'tiny image'
CLASSIFIER = 'nearest neighbor'
Accuracy = 0.221

section 2.

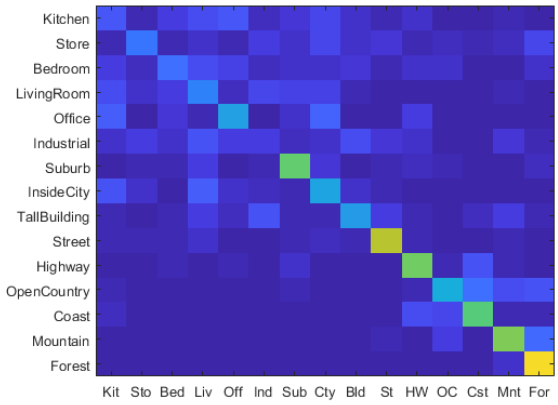

Figure 2: bag of words&nearest neighbor

FEATURE = 'bag of words'
CLASSIFIER = 'nearest neighbor'
Accuracy = 0.448

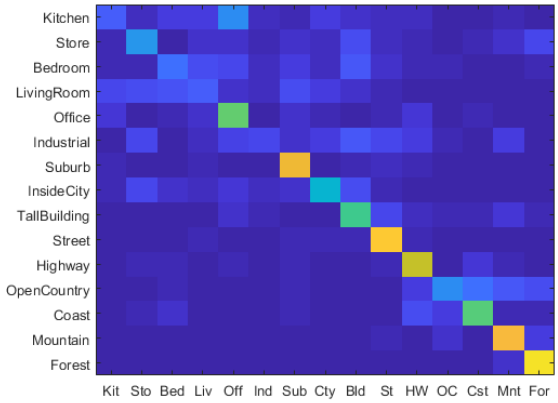

Figure 3: bag of words&support vecter machine

section 3.
FEATURE = 'bag of words'
CLASSIFIER = 'support vecter machine'
Accuracy = 0.518