

SCENE CLASSIFICATION USING BAG OF WORDS

INTRODUCTION:

Scene recognition is one of the interesting problems in computer vision. This project uses different types of feature representation and classification to identify the scene type. A training set of 1500 labelled scenes are used as a sample set to identify the scene type of test images.

The feature representation techniques used are :

- 1 Bag of words
- 2 Gist descriptors
- 3 Sift descriptors
- 4 Gist along with Sift descriptors
- 5 Tiny images
- 6 Spatial Pyramid sift
- 7 Gist along with fisher encoding
- 8 Spatial Pyramid with sift, gist and fisher encoding

The classification techniques used are :

- 1 Support Vector Machine
- 2 Nearest Neighbor classifier

In addition, I experimented using Fisher encoding instead of the bag of SIFT representation and found that Fisher encoding can significantly improve the performance of the classifier.

Bag of sifts and K-NN Classifier

Step 1: Represent each image with the appropriate feature: BAGS OF SIFTS

CODE: *get bags of sift.m*

METHOD 1 : Using vl_dsift

```
num_of_images = size(image_paths, 1);
image_feats = zeros(num_of_images, vocab_size);
vocab_inv = vocab';
k = 10;
parfor i = 1 : num_of_images
    img = imread(image_paths{i});
    [~, features] = vl_dsift(img, 'Step', 10);
    features = single(features);
    distance_table = vl_alldist2(vocab_inv, features);
    [~, index] = sort(distance_table);
    feat_count = zeros(vocab_size, 1);
```

```

    for j = 1 : vocab_size
        feat_count(index(1 : k, j)) = feat_count(index(1 : k, j)) + 1;
    end
    feat_count = feat_count / norm(feat_count);
    feat_count = feat_count';

    image_feats(i, :) = feat_count;
end

```

METHOD 2: Using KD trees - to increase the speed

```

%% using KD - trees

% % size of histogram or num_features
d = vocab_size;
n = length(image_paths);

image_feats = ones(n, d);

% build kdtree for distance querying
kdtree = vl_kdtreebuild(vocab', 'NumTrees', 1);

for i = 1:n
    image = single(imread(image_paths{i}));

    % Get the locations and descriptors for the image
    % with a fast dense sift with fixed size and step
    % descriptors = 128 x num_features_found
    % locations = 2 x num_features_found
    [~, descriptors] = vl_dsift(image, 'step', 8, 'size', 4, 'fast');

    % Find the closest vocab given this image's descriptors
    % each descriptor is found closest to the distance in the vocab
    % matrix
    % min finds the closes of the descriptor to the vocab
    % min_indices = 1 x num_features_found with values from
    % 1..vocab_size

    rand_descriptors = descriptors;
    [min_indices, ~] = vl_kdtreequery(kdtree, vocab', single(rand_descriptors));

    % Build histogram from min_indices
    count_histogram = hist(single(min_indices), d);

    image_feats(i, :) = count_histogram ./ size(rand_descriptors, 2);
end

```

The bag of sifts implementation has two stages:

- 1) building of the vocabulary
- 2) obtaining feature vector for each image.

1.1- In the Build vocabulary stage: each training image is densely sampled to obtain a set of feature vectors. This set is clustered to form vocabulary of N 'words'. The performance of the bag of sift model, depends on the vocabulary size, nSample, nSampleStep and nSupportSize

1.1 - Results: The following result was obtained by varying the 3 parameters:

1.1.a - Varying nSampleStep :

vocab_size	nSample	nSampleStep	nSupportSize	Accuracy
400	100	20	8	0.385
400	100	15	8	0.361
400	100	10	8	0.450
400	100	5	8	0.360
400	100	3	8	0.376

1.1.b - Varying the nSupportSize :

vocab_size	nSample	nSampleStep	nSupportSize	Accuracy
400	100	10	4	0.375
400	100	10	8	0.450
400	100	10	12	0.424

1.1.c - Varying the nSample:

vocab_size	nSample	nSampleStep	nSupportSize	Accuracy
400	200	10	8	0.357
400	100	10	8	0.450
400	50	10	8	0.386
400	20	10	8	0.313

1.2 After the vocabulary has been built, each training and testing image is densely sampled and a histogram map of 'words' in the given image are constructed. These features of length equal to vocab_size are then fed to the K-NN block to calculate the class of a test image.

In addition, the performance of scene recognition is affected by the size of the vocabulary size. In general, as the vocabulary size increases, the accuracy increases correspondingly.

1.2. Results: Varying vocab size

vocab_size	nSample	nSampleStep	nSupportSize	Accuracy
400	20	5	8	0.473
200	20	5	8	0.467
100	20	5	8	0.389

Step 2: Classify each test image by training and using the appropriate classifier : K-NN

Nearest Neighbor

The nearest neighbor is simple implementation, where pairwise distances are calculated, sorted and the class of the K nearest neighbors considered and test image is placed in class that the majority of neighbors belong to.

CODE: nearest_neighbor_classify.m

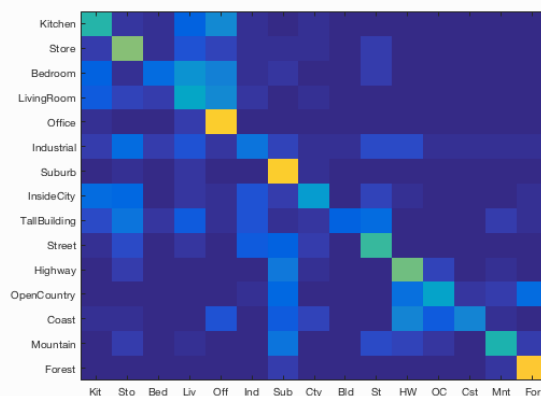
```
M = size(test_image_feats,1);
k = 20;
distances = vl_alldist2(train_image_feats', test_image_feats');
all_labels = unique(train_labels);
n_labels = size(all_labels, 1);
[~, indices] = sort(distances, 1);
count_labels = zeros(n_labels, M);
for i = 1:M
    for j = 1:n_labels
        top_k_labels = train_labels(indices(1:k, i));
        count_labels(j,i) = sum(strcmp(all_labels(j), top_k_labels));
    end
end
[~, label_indices] = max(count_labels,[],1);
predicted_categories = all_labels(label_indices);
```

Results: Varying k values

K	Accuracy
4	0.387
8	0.443
20	0.450



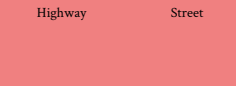




















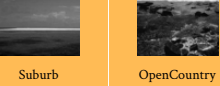
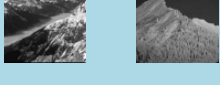

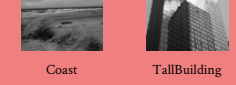

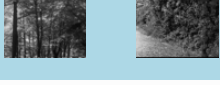

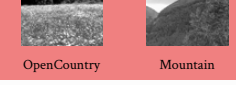

2.Results: Accuracy :0.473 with k = 20 , vocab size: 200

project 6 results visualization



Accuracy (mean of diagonal of confusion matrix) is 0.473

Category name	Accuracy	Sample training images	Sample true positives	False positives with true label	False negatives with wrong predicted label
Kitchen	0.470				
Store	0.620				
Bedroom	0.160				
LivingRoom	0.390				
Office	0.890				
Industrial	0.190				
Suburb	0.880				
InsideCity	0.330				

					
TallBuilding	0.130				
Street	0.500				
Highway	0.580				
OpenCountry	0.360				
Coast	0.260				
Mountain	0.460				
Forest	0.870				
Category name	Accuracy	Sample training images	Sample true positives	False positives with true label	False negatives with wrong predicted label

EXTRA CREDIT :**1) Tiny Images and K-NN Classifier:**

Each of the training and test images are resized to a 16*16 image, and reshaped into a 256 dimensional vector. Each test image is then compared to K nearest neighbors from the training set, and placed in the category of the majority of the closest neighbors. This is simple to understand in that, it compares the test image with each of the training images and find the most similar images. This is expected to be not particularly effective, because it is based only on similarity of images and not on the underlying structure of the image.

Tiny Images

The implementation crops the image to the nearest multiple of 16*16 and then resizes it to a 16*16 image rather than directly resizing it so as to not warp the images. A moderate improvement is seen by cropping over direct resizing. Mean Centering and Normalizing also modestly improves the performance

Accuracy Obtained : 0.219

2) SVM + Bags Of Sift :

For SVM, we basically have to make it ONE vs ALL, for all the labels and compute the score for an image based on its highest score across all of these ONE vs ALL models. Using Nearest Neighbor, I was able to achieve around 47.3% percent accuracy and with SVM, I achieved about 56.9 % accuracy. For SVM , regularization is an important factor and idea was to find a subtle point, where the accuracy was stable and independent of data modifications. I tried different data permutations , to check how it varies with hyper-parameter and LAMBDA = 0.000001 was robust and stable to the model.

Accuracy Obtained : 0.569

3) Spatial Pyramid representation + SVM:

- (i) Bag-of-features methods, which represent an image as an orderless collection of local features, have recently demonstrated impressive levels of performance . However, because these methods disregard all information about the spatial layout of the features, they have severely limited descriptive ability. In particular, they are incapable of capturing shape or of segmenting an object from its background.
- (ii) It is important to contrast our proposed approach with multi-resolution histograms [8], which involve repeatedly subsampling an image and computing a global histogram of pixel values at each new level. In other words, a multi-resolution histogram varies the resolution at which the features (intensity values) are computed, but the histogram resolution (intensity scale) stays fixed. We take the opposite approach of fixing the resolution at which the features are computed, but

varying the spatial resolution at which they are aggregated. This results in a higher-dimensional representation that preserves more information.

- (iii) In the figure toy example is provided which provides the intuition of how pyramid matching is applied and calculated. Using Spatial information seems to have a good overall impact and about 5 percent increase in performance from SVM. I got around 61.4 percent accuracy in this model. As, we will see later by concatenating this, Bag of SIFT and GIST features and Fisher, we get around 78.7 percent accuracy.

Accuracy Obtained : 0.614

4) GIST features + SVM classifier

Although SIFT features are quite descriptive, there are many other feature representations that can be useful in order to classify a scene. I experimented using GIST features using the implementation from (<http://people.csail.mit.edu/torralba/code/spatialenvelope/>). The GIST feature vector extracted from each image is normalized and used as the feature vector in place of the SIFT histogram. Using the GIST features results in an accuracy of 57.6%.

Accuracy Obtained : 0.576

5) Fisher encoding for SIFT features + SVM classifier

Using histograms to encode SIFT features performs well in order to classify scenes. However, more sophisticated feature encoding schemes such as Fisher encoding can potentially retain more information that can be used by the classifier to classify the scene. To create the vocab needed for Fisher encoding, I use the `vl_gmm()` function with 50 clusters. When extracting image features from test images, I use the `vl_fisher()` function to encode the extracted SIFT features and return the normalized feature vector. Although the extracted SIFT descriptors used are the same, Fisher encoding performs better than the Bag of SIFT representation. The accuracy rate achieved by Fisher encoding is 71.9%.

Accuracy Obtained : 0.719

6) GIST features + Fisher encoding of SIFT descriptors + SVM classifier

Both SIFT and GIST feature representations describe different features in their target images. By combining the feature vectors obtained using GIST and SIFT descriptors, I wanted to see if these complementary features could be used together to increase scene classification accuracy. I combined the GIST feature vectors with the Fisher encoding of SIFT descriptors by appending the feature vectors together. The final accuracy rate I achieve by combining the feature vectors is 79.1%, which is greater than using solely SIFT or Fisher descriptors.

Accuracy Obtained : 0.791

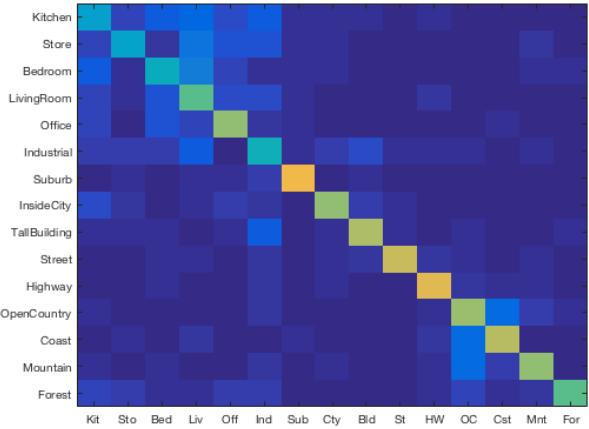
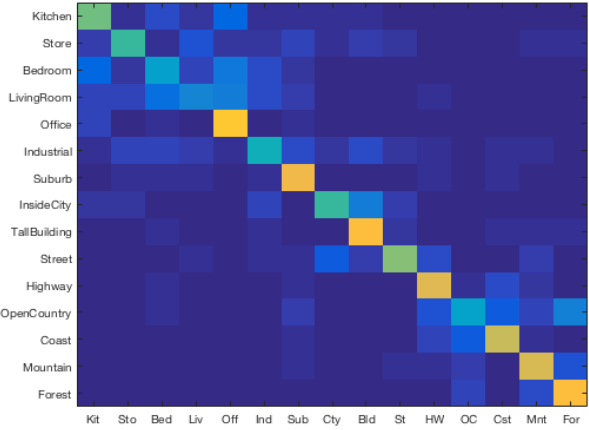
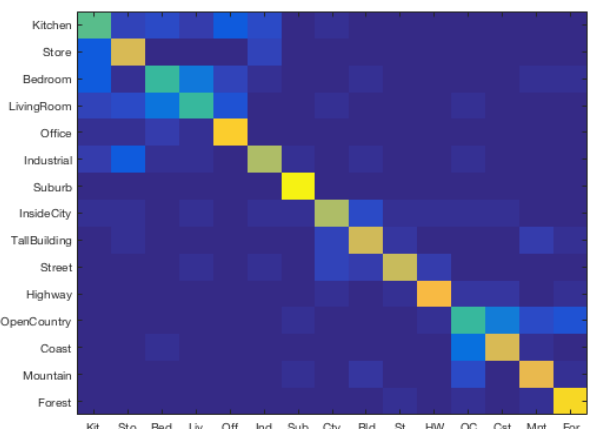
7) Spatial Pyramid + Fisher + GIST + SVM :

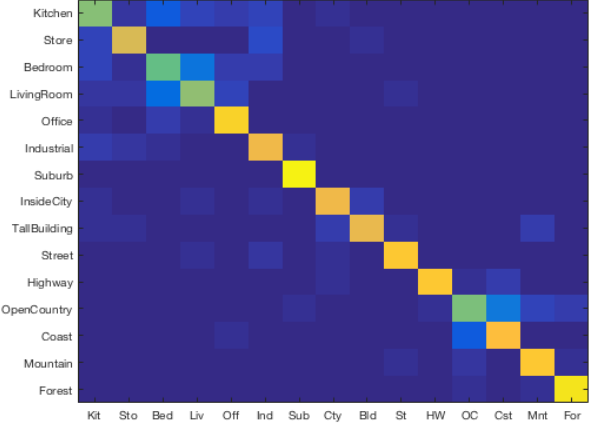
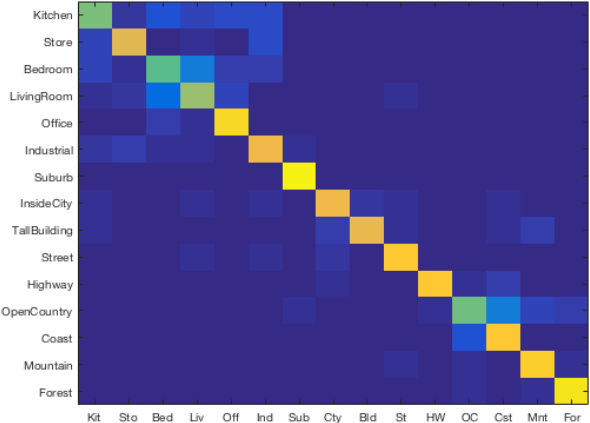
Accuracy Obtained : 0.787

RESULTS FROM VARIOUS MODELS : CONFUSION MATRIX

MODEL	CONFUSION MATRIX	ACCURACY
tiny images + SVM		0.135
tiny images + K -NN		0.219

MODEL	CONFUSION MATRIX	ACCURACY
bags of sift + K-NN		0.473
bags of sift + SVM		0.569
GIST + SVM		0.576

MODEL	CONFUSION MATRIX	ACCURACY
GIST + SIFT + SVM		0.593
Spatial Representation + SIFT + SVM		0.614
SIFT with Fisher + SVM		0.719

MODEL	CONFUSION MATRIX	ACCURACY
Spatial Representation + SIFT + Fisher + GIST + SVM		0.787
GIST + Fisher + SVM		0.791

CONCLUSION :

- 1) Thus from the above results table, the best model was obtained by combining GIST with Fisher encoding and using SVM classifier. I was able to obtain the accuracy of 79.1 %.
- 2) Overall, Using SVM classifier improves the performance results in comparison to nearest neighbor method.
- 3) Considering the nearest neighbor method, higher the K value better the performance obtained.
- 4) Using Fisher encoding along with Sift improved the performance tremendously by over 10% performance gain.