Scene Recognition with Bag of Words

22221290 Jiarun Liu

Environment

- Python 3.7+ required

```
cd code
pip install -r requirements.txt
```

Tiny Images Representation

Running

```
cd code
python tiny_images_representation.py
```

Implementation

1. Tiny Image Feature

   Simply follow the instruction and scale the image size to 16x16. Moreover, a normalization is applied to each tiny image. Checkout function `get_tiny_images()` in `tiny_iamges_representation.py`

2. KNN

   To calculate L2 distance, I wrote a function `euclidean_distances(x,y)` to compute L2 distance between each row vector in matrix x and y.

   ```
   def euclidean_distances(x, y):
       x_square = np.sum(x*x, axis=1, keepdims=True)
       if x is y:
           y_square = x_square.T
       else:
           y_square = np.sum(y*y, axis=1, keepdims=True).T
       distances = np.dot(x, y.T)
       distances *= -2
       distances += x_square
       distances += y_square
       np.maximum(distances, 0, distances)
       if x is y:
           distances.flat[::distances.shape[0] + 1] = 0.0
       np.sqrt(distances, distances)
       return distances
   ```

   After that, the operation is simple. Checkout function `knn_classifier()` in `utils.py`. Notice that the parameter k is changeable. The default value is 1.

Results

Tiny image representation method gets 24% accuracy on the 15 scene database, which is subject to 15% to 25% accuracy.

```
Average accuracy: 0.24
Bedroom: 0.15
Coast: 0.37
Forest: 0.1
Highway: 0.6
Industrial: 0.1
InsideCity: 0.07
Kitchen: 0.16
LivingRoom: 0.09
Mountain: 0.31
Office: 0.17
OpenCountry: 0.41
Store: 0.04
Street: 0.48
Suburb: 0.39
TallBuilding: 0.16
```

Bag of SIFT Representation

Running

```
cd code
python bag_of_sift_representation.py
```

Implementation

1. Building the vocabulary of visual words

   To extract SIFT feature, I used `cv2.SIFT.create()` from OpenCV and wrote the function `calcSIFT()`.

   ```
   def calcSIFT(img, stride=10, size=16):
       sift = cv2.SIFT_create()
       kp = [cv2.KeyPoint(x, y, stride) for y in range(int(size/2),
   img.shape[0], stride)
                                         for x in range(int(size/2),
   img.shape[1], stride)]
       _, des = sift.compute(img, kp)
       return des
   ```

   To create the library, we are going to sample the feature based on SIFT descriptor and then clustering them to get the category centers. I used `cv2.kmeans()` from OpenCV. Notice that the parameters can be changed.

```
criteria = (cv2.TERM_CRITERIA_EPS +
            cv2.TERM_CRITERIA_MAX_ITER, 20, 0.1)
compactness, labels, centers = cv2.kmeans(bag_of_features,
                                          vocab_size=200,
                                          bestLabels=None,
                                          criteria=criteria,
                                          attempts=20,
                                          flags=cv2.KMEANS_PP_CENTERS)
vocab = np.vstack(centers)
```

Basically, the larger the vocabulary size, the better the prediction results.

2. Represent images as histograms of visual words

   We use L2 distance to measure which cluster the descriptor belongs, creating corresponding histograms of visual words of each image. A normalization is adapted to the histogram.

3. KNN

   That's clear enough.

Results

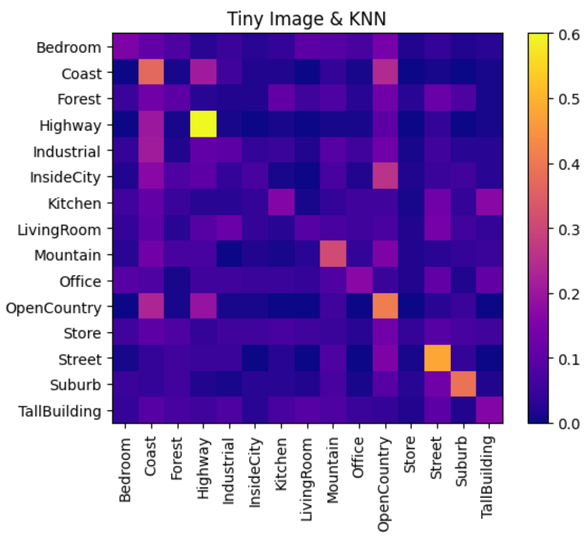
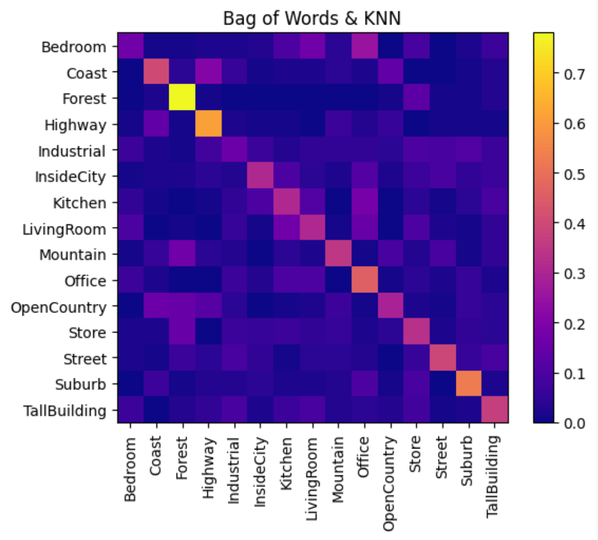
Bag of SIFT method gets 38.47% accuracy on the 15 scene database, which is subject to 30% to 40% accuracy.

```
Average accuracy: 0.38466666666666666
Bedroom: 0.17
Coast: 0.4
Forest: 0.78
Highway: 0.61
Industrial: 0.16
InsideCity: 0.31
Kitchen: 0.31
LivingRoom: 0.31
Mountain: 0.35
Office: 0.46
OpenCountry: 0.29
Store: 0.33
Street: 0.39
Suburb: 0.53
TallBuilding: 0.37
```

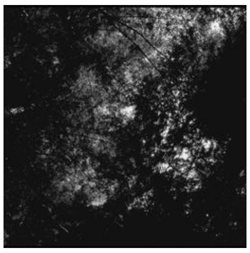
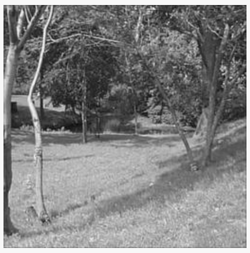
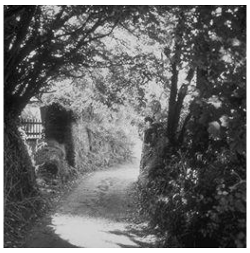
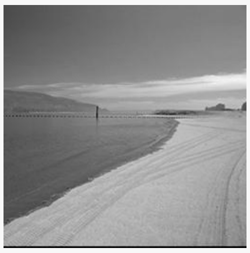
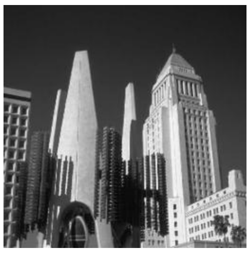
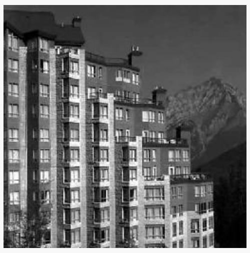
Analysis and Conclusion

Comparison

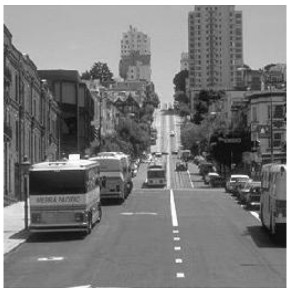
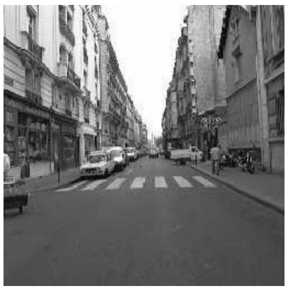
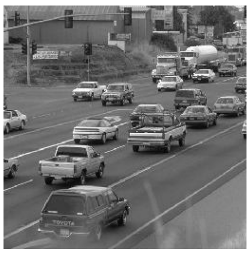
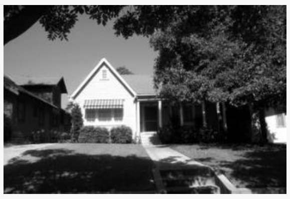
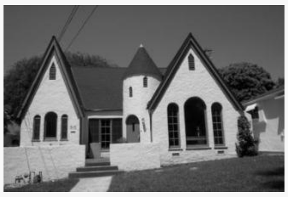
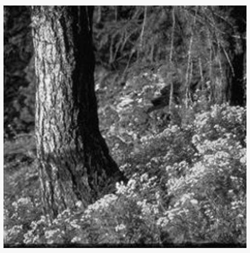
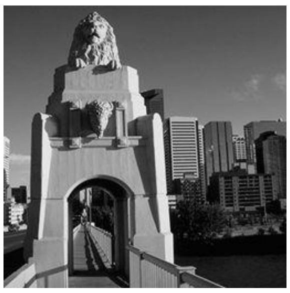
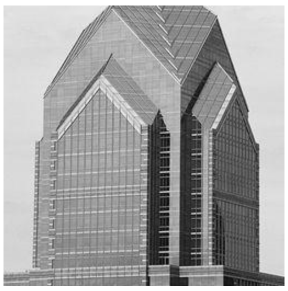
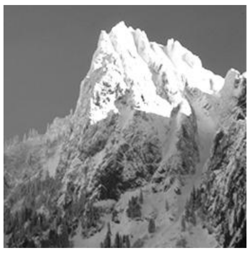
A confusion matrix is very intuitive to explore the accuracy. We mainly focus on the diagonal of the matrix which represents the accuracy on each class.

Method	Accuracy	Confusion Matrix
Tiny Image Representation + KNN	24%	
Bag of SIFTs + KNN	38.47%	

Some example results

Category	Ground Truth Example	Correct Result Example	Incorrect Result Example
Bedroom			
Coast			
Forest			
Highway			
Industrial			
InsideCity			

Category	**Ground Truth Example**	**Correct Result Example**	**Incorrect Result Example**
Kitchen			
LivingRoom			
Mountain			
Office			
OpenCountry			
Store			

Category	Ground Truth Example	Correct Result Example	Incorrect Result Example
Street			
Suburb			
TallBuilding			

Discussion

During the implementation of the Bag of SIFT algorithm, we find out that the larger the vocabulary size, the better the prediction results. In the other hand, the larger vocabulary size means the bigger computation costs, which leads to a long time of execution.

In this 15 scene database, `Forest` has the highest recognition accuracy and `Industrial` has the lowest one. I think this is mostly because of the similarity of SIFT features for industrial scene and city-scape scenes, which leads to a confusion.

A better feature extraction method may improve the algorithm. What's more, during the classification task, we can apply some machine learning methods such as MVS to get a better performance of the model.