Computer Vision
HW5 Report

Team 3 , 0751231 曾揚 , 309505018 郭俊廷

Introduction:

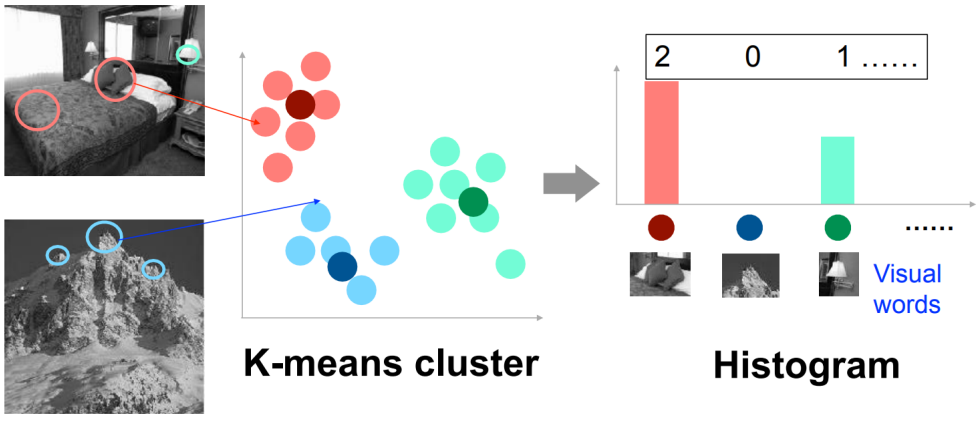
In this assignment, we want to classify the scene datasts. We implemented three methods to classify 15 classes, and show the accuracy in each method. The Scene datasets have 15 classes, and each class has about 100 images in training data and about 10 images testing data. Each image is 200x200 pixels. Finally, we have some discussion of the pros and cons of each method in the end.

Implementation procedure:

1. Tiny images representation + Nearest neighbor classifier:
   1) Resize each image to a small fixed resolution 16x16.
   2) Use a vector of 256 dimensions to represent an image.
   3) Normalize each vector, Length is one and mean is zero.
   4) Construct the distance matrix by using Euclidean distance, and apply K-nearest neighbors algorithm to classify test images.

2. Bag of SIFT representation + Nearest neighbor classifier:
   1) We have to build the vocabulary, so we use SIFT API in vlfeat Library to extract features for each training image. As our implementation, we found most features in an image are redundant, that means some features are the same. So, we only choose 1% of features as our valid feature.

   2) Then, using the K-means clustering algorithm to these chosen features, we can get K cluster centers, and these centers are called “visual words”. We want to plot histograms to represent training data and testing data. For each image, we can extract its features by using SIFT and find out which center it belongs to.



3) After getting the histogram of each image, we can count the number of each cluster center that features were assigned to.

4) Finally, calculate the pairwise distance between each testing image and all the training images, then use the K-nearest neighbors algorithm to classify the testing image.

3. Bag of SIFT representation + Linear SVM classifier:
   1) We have to build the vocabulary, so we use SIFT API in vlfeat Library to extract features for each training image. As our implementation, we found most features in an image are redundant, that means some features are the same. So, we only choose 50% of features as our valid feature.
   2) Then use the SVM classifier with linear kernel training in libsvm Library. Train SVM with bags of words, histograms and the label of images.

4. Deep Learning:

   In this assignment, the dataset is too small, it’s hard to apply deep neural network to classify without augmentation. We normalize the images to prevent overfitting and enhance the result of the classifier. We used two famous models, one is VGG, another is ResNet.

   VGG16 uses multiple convolutional layers with a smaller convolution kernel (3x3) instead. It can reduce parameters, and it's equivalent to performing more nonlinear mappings, which can increase the network's fitting ability. Since the convolution kernel focuses on expanding the number of channels, it makes the model architecture deeper and wider while controlling the increase in the amount of calculation.

   ResNet34 wants to solve the gradient vanish problem while deeper layers. ResNet used short-cut mapping to connect current layer to previous layer inorder to solve gradient vanish problem, it’s called bottleneck architecture.

   We used a pretrained model and removed the last layer. Then we concatenate the output layer with fifteen classes to the pretrained model.

Experimental result :

1) Tiny images representation + Nearest neighbor classifier:
   k-nearest neighbors k =1
   accuracy: **22.0%**

2) Bag of SIFT representation + Nearest neighbor classifier:
   vocabulary shape: (13980, 128)
   vocabulary size: 100
   k-nearest neighbors k =7
   accuracy: **55.33%**

3) Bag of SIFT representation + Linear SVM classifier:
   vocabulary shape: (295829, 128)
   vocabulary size: 100
   accuracy: **65.33%**

4) Deep Learning

   - VGG16:
     image resize: (224,224)
     epoch: 20
     accuracy: **91.0%**

   - RestNet34:
     image resize: (224,224)
     epoch: 20
     accuracy: **94.0%**

Discussion:

1) Normalization or not:
   In task 1, after we normalize the tiny image features, the result improved from 14% to 22%. But in task 2 and task 3 , the normalization for the bag of SIFT features actually leads to a much poorer accuracy. We guess that it is because the features in the SIFT means orientations in the image, so if we normalize it, the descriptor features will not be that clear.

2) Randomly pick features or use them all:
   In task 2, when building the vocabulary of the features, we tried to use two methods, the first method is randomly picking x features in the image, and the second method is building the vocabulary using all the features found in images. The results are surprisingly very similar. So we randomly pick x features to reduce our execution time.

3) Which CNN model is the best fit in this datasets:
   Since the dataset is too small, it is hard to apply deep neural networks to classify without augmentation. The performance of models are different in different dataset. In our experiment, ResNet34 has the best performance.

Conclusion:

We use four different methods to do the scene classification

In task 1, the image is low resolution, so that the feature of images can't be described very well. So, it's reasonable that the accuracy is quite low.

In task 2, we used the SIFT API in the Opencv Library in the beginning, but the performance isn't very well. So we use vlfeat Library instead. First, we need to build vocabulary, and use SIFT to detect features.We only choose 1% of features as our valid feature, that is because most features in an image are redundant. Finally, we reach 55.33% accuracy.

In task 3, we used the SIFT API in the vlfeat Library to detect features. Then we use the SVM classifier API in libsvm Library. The performance reaches 65.33% accuracy.

In task 4, we tried two kinds of CNN models to classify scenes, and ResNet34 has the best performance.