EC – 413 Computer Vision
Scene Classification using Bag of Words

Azka Rehman (arehman.ee38ceme) – 174227
Mushkbar Fatima (fmushkbar.ee38ceme) – 175540

June 21, 2020

Contribution to work

- Pre-processing of data and accessing the data - Mushkbar Fatima
- Extraction of Features -Azka Rehman
- Training SVM -Mushkbar Fatima
- Report - Azka Rehman

Contents

1 Problem Description

Dataset of differents scenes were given. Classification of each scene was required using following steps:

- Pre-processing of data
- Extraction of descriptors
- Generating vocabulary
- Construction of Bag of words
- Training of SVM classifiers

2 Important Functions and Libraries

Coding is done in python.

Following libraries are used while coding this assignment.

- numpy
- math
- OpenCV
- os
- matplotlib
- pyplot
- sklearn.cluster KMeans
- scipy.cluster.vq
- sklearn.preprocessing StandardScaler
- sklearn metrics
- joblib

3 Important:

- **No.of Training Images=100 images per category**
- **No.of Testing Images=20 images per category**

- **Algorithm for feature extraction=SIFT**
- **Kernel in SVM classifier="rbf"**
- **Standard scalar function from skearn library was used to standardize the data. It Standardize features by removing the mean and scaling to unit variance.**
- **Pickel files from joblib were used to store extracted features to avoid processing again and again.**

4 Discussion of Code

4.1 Removing unwanted data

This function takes path of scene categories as input and removes image files after image number 120. From this data, training and testing data was chosen.

4.2 Accessing the Training and Testing Data:

This function takes path of "scenecategories" as input and return labelled lists of path of testing and training data as output. It also returns name of categories as outputs.

4.3 Feature Extraction:

$gensiftfeatures$ extracts features of all images and returns a list containing arrays of descriptors. It also returns labels of descriptor array for each image.

4.4 Formation of Bag of words:

Vocabulary is formed by performing K means clustering on the array which contains all descriptors. This vocabulary is used to generate bag of features for all images using list of descriptors. It returns kmeans model, and bag of words.

4.5 Training of SVM classifier:

This function takes bag of words and labels as input and return SVM classifier model and standard scalar model as output.

4.6 Testing of SVM classifier:

This function takes kmeans model, svm model, bag of words of test data, labels of test data, number of clusters and standard scalar model as input and plots the confusion matrix and prints the accuracy as output.

5 Code Listings

5.1 Importing Libraries

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import os
import random
from sklearn.cluster import KMeans
from sklearn.neighbors import NearestNeighbors
from scipy.cluster.vq import kmeans, vq
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, accuracy_score
import joblib
import time
```

5.2 Execution of Defined Funtions

```
#getting the data
path=r'C:\Users\AZKA_REHMAN\Desktop\8th_sem\CV\scene_categories'
category_names,train_data,test_data=train_test_split(path)

#process of training
img_desc,y=gen_sift_features(train_data)
k=50 #clusters
X,kmeans=bow_features(img_desc,k)
model,standardScalar=SVC_training(X,y)

#testing the model
test_desc,t=gen_sift_features(test_data)
model_testing(model,test_desc,t,kmeans,k,standardScalar,category_names)
```

5.3 Accessing the data

```
def train_test_split(path):
    for folder in path:
        names_of_categories = os.listdir(path)
    training_labelled_paths=[]
    testing_labelled_paths=[]
    label=0
    for i in  names_of_categories:
        folder = os.path.join(path,i)
        imgs = os.listdir(folder)
        split=0
        for img in imgs:
            image= os.path.join(folder,img)
            split+=1
            if split<=100:
```

```
                training_labelled_paths.append((image,label))
            else:
                testing_labelled_paths.append((image,label))
        label+=1
    return names_of_categories, training_labelled_paths,testing_labelled_paths
```

5.4 Generating List of features

```
def gen_sift_features(labeled_img_paths):
    img_descs = []
    for img_path, label in labeled_img_paths:
        img = cv2.imread(img_path)
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        sift = cv2.xfeatures2d.SIFT_create()
        kp, desc = sift.detectAndCompute(gray, None)
        img_descs.append(desc)
        y = np.array(labeled_img_paths)[:,1]
    return img_descs, y
```

5.5 Generating Bag of Words

```
def bow_features(img_desc,k):
    all_train_descriptors = [desc for desc_list in img_desc for desc in desc_list]
    all_train_descriptors = np.array(all_train_descriptors)
    kmeans = KMeans(n_clusters=k, random_state=0).fit(all_train_descriptors)
    vocabulary=kmeans.cluster_centers_
    bag_of_words = [kmeans.predict(raw_words) for raw_words in img_desc]
    img_bow_hist = np.array(
        [np.bincount(clustered_words, minlength=k) for clustered_words in bag_of_words

    X = img_bow_hist
    print('done generating BoW histograms.')
    return X, kmeans
```

5.6 Training SVM

```
def SVC_training(X,labels):
    standardScalar = StandardScaler().fit(X)
    X = standardScalar.transform(X)
    model = SVC(gamma='auto',kernel='rbf')
    m=model.fit(X,labels)
    return model,standardScalar
```

5.7 Testing of Model

```
def model_testing(model,test_desc,t,kmeans,k,standardScalar,category_names):
    all_test_descriptors = [desc for desc_list in test_desc for desc in desc_list]
    all_test_descriptors = np.array(all_test_descriptors)
```

```
    bag_of_words = [kmeans.predict(raw_words) for raw_words in test_desc]
    img_bow_hist = np.array(
                        [np.bincount(clustered_words, minlength=k)
                         for clustered_words in bag_of_words])
    X = img_bow_hist
    X = standardScalar.transform(X)
    t = np.array(t).astype(int)
    predictions = model.predict(X)
    predictions=np.array(predictions).astype(int)
    true_class =  [category_names[i] for i in t]

# Perform the predictions and report predicted class names.
    predictions =  [category_names[i] for i in predictions]
    accuracy = accuracy_score(true_class, predictions)
    print("Accuracy=",accuracy)

#plotting confusion matrrix
    conf_mat= confusion_matrix(true_class,predictions,labels=category_names)
    print("confusion matrix=")
    print(conf_mat)
    plt.matshow(conf_mat)
    plt.ylabel('True Category')
    plt.xlabel('Predicted Category')
    plt.colorbar()
    plt.xticks(np.arange(len(category_names)),category_names,rotation=90)
    plt.yticks(np.arange(len(category_names)),category_names,rotation=45)
    plt.show()
```

6 Results

6.1 Accuracy

```
Accuracy= 0.45666666666666667
confusion matrix=
[[ 3  0  0  3  7  1  0  0  0  0  0  1  0  4  1]
 [ 0 19  0  0  1  0  0  0  0  0  0  0  0  0  0]
 [ 1  0  2  2  4  0  0  1  0  0  0  6  2  0  2]
 [ 3  0  1  5  6  0  0  0  0  0  0  3  0  1  1]
 [ 4  0  0  2  9  0  0  0  1  0  0  0  1  2  1]
 [ 0  0  0  0  0 11  0  3  0  6  0  0  0  0  0]
 [ 0  0  0  0  0  0 17  0  0  1  1  1  0  0  0]
 [ 1  0  1  0  0  2  0 16  0  0  0  0  0  0  0]
 [ 3  0  0  4  0  0  0  0 12  0  0  0  0  1  0]
 [ 0  0  0  0  1  2  1  2  0 12  0  1  0  0  1]
 [ 0  0  0  0  0  5  0  5  0  7  3  0  0  0  0]
 [ 0  0  3  1  0  0  0  1  1  4  0  7  1  0  2]
 [ 0  1  3  0  2  0  0  0  1  1  0  2  7  3  0]
 [ 5  1  0  2  4  0  0  0  1  0  0  0  0  6  1]
 [ 1  0  2  4  3  0  0  0  0  0  0  2  0  0  8]]
```

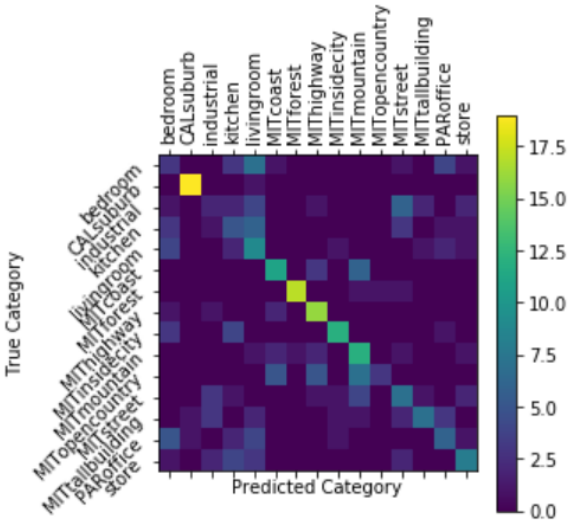
(a) Accuracy with 50 clusters

```
Accuracy= 0.5066666666666667
confusion matrix=
[[ 4  0  0  3  7  1  0  0  0  0  0  1  0  3  1]
 [ 0 20  0  0  0  0  0  0  0  0  0  0  0  0  0]
 [ 1  0  2  1  3  0  0  1  0  0  0  5  2  1  4]
 [ 3  0  1  6  4  0  0  0  0  0  0  3  0  2  1]
 [ 3  0  0  3  7  0  1  0  1  0  0  1  0  3  1]
 [ 0  0  0  0  0 12  0  0  0  6  1  0  1  0  0]
 [ 0  0  0  0  0  0 19  0  0  1  0  0  0  0  0]
 [ 0  0  0  0  0  1  0 19  0  0  0  0  0  0  0]
 [ 3  0  0  2  0  0  0  0 12  0  0  0  0  2  1]
 [ 0  0  0  0  0  2  1  1  0 14  1  0  0  0  1]
 [ 1  0  0  0  0  4  0  2  0  7  6  0  0  0  0]
 [ 0  0  2  1  0  0  0  1  0  1  0 11  1  0  3]
 [ 1  0  2  1  3  0  0  0  6  1  0  0  6  0  0]
 [ 3  0  1  3  3  0  0  0  2  0  0  0  0  7  1]
 [ 1  0  2  1  3  0  1  1  1  0  0  1  0  2  7]]
```

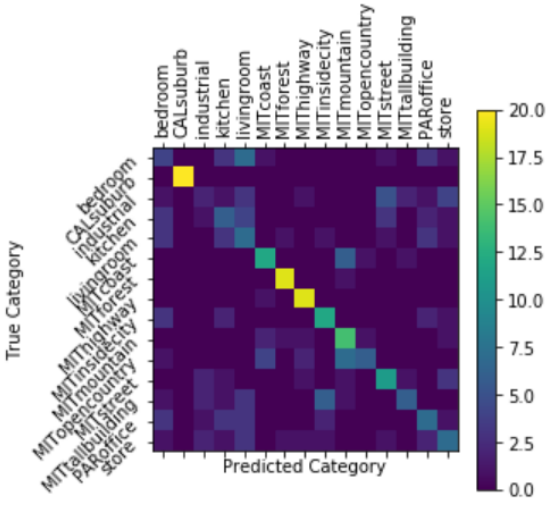
(b) Accuracy with 200 clusters

Figure 1: Accuracy of SVM classifier

6.2 Confusion Matrix



(a) Confusion Matrix with 50 clusters



(b) Confusion Matrix with 200 clusters

Figure 2: Plots of Confusion Matrix

7 Important Findings

Following are some important observations done while implementing the algorithm of given task:

- Accuracy is higher with higher number of clusters as the classifier has more information available
- If only selected number of features from each image is used then accuracy is comparatively low. It tends to increase by 3 to 4 percent if all descriptors are used.
- Confusion matrix is used to summarize the number of correct and incorrect predictions.
- It can be seen from confusion matrix plot that for half of categories the classifier performed very good i.e. it predicted positive and it's true whereas half of the time its predictions were false.