Summary of Scene Recognition

The homework is to identify different scenes by a bunch of training images in three different methods: tiny image with KNN, BOW with KNN, SVM with KNN. The functions are as follow:

1. feature = get_tiny_image(img, output_size)

   In this function, with the output_size, we can use the cv2.resize function directly to resize the image into feature image, and we use np.mean, np.sum and np.sqrt functions to normalize the feature image.

2. label_test_pred = predict_knn(feature_train, label_train, feature_test, k)

   In this function, KNeighborsClassifier in sklearn.neighbors was called to fit feature_train and classify feature_test.

3. confusion, accuracy = classify_knn_tiny(label_classes, label_train_list, img_train_list, label_test_list, img_test_list)

   classify_knn_tiny takes in all the training data of images, then transforms these images into tiny images (20x16 here) and vectorizes them all. The classifier is KNN while the k is 15, and the accuracy obtained with this classifier is 0.214.

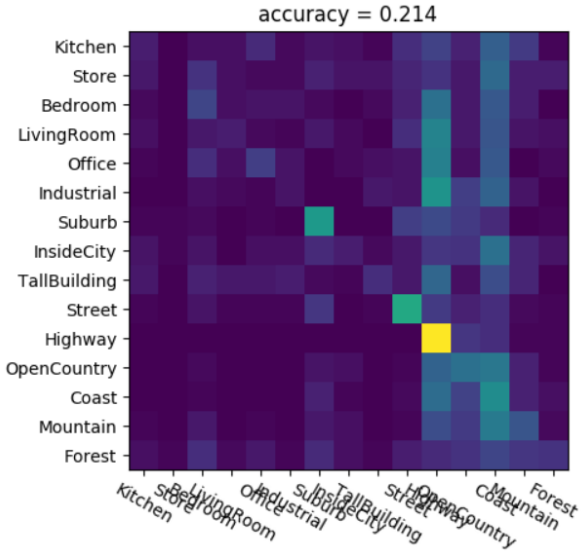

FIG.1 KNN and tiny image

4. dense_feature = compute_dsift(img, stride, size)

   Compute_dsift function is to find the dense sift of all the training images. For all middle points of the grids, cv.KeyPoint is made to carry the information, coordinate X, Y, and the grid size. Sift.compute function can directly compute the input KeyPoints, generate Sift KeyPoints.

5. vocab = build_visual_dictionary(dense_feature_list, dic_size)

   Build_visual_dictionary takes in dense_feature_list generated by compute_dsift function and dic_size is the number of the KMeans centers (50 in this algorithm). KMeans algorithm is used to find cluster centers of the dense_feature_list, vocab is the list of those centers.

6. bow_feature = compute_bow(feature, vocab)

   Compute_bow function takes in the dense sift keypoints of the image, then uses NearestNeighbor to classify the keypoints with the vocab and find the number of keypoints in different bags of the image.

7. confusion, accuracy = classify_knn_bow(label_classes, label_train_list, img_train_list, label_test_list, img_test_list)

   Classify_knn_bow function is to classify testing data with KNN, the first step is to calculate the vocab with all training data, then, in two loops, the bag of world of all the images is obtained by compute_dsift (stride and size are 14) and compute_bow functions. After that, predict_knn takes in training data bow and its label, testing data bow, and a k (k=16 here), the accuracy obtained by this classifier is 0.513.

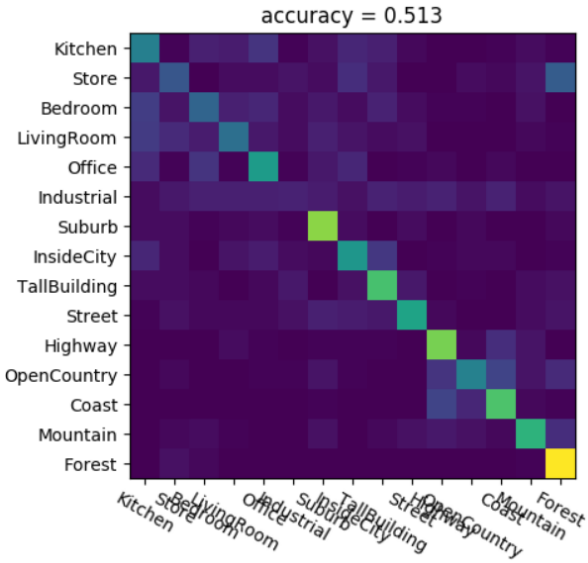

FIG.2 BOW and KNN

8. label_test_pred = predict_svm(feature_train, label_train, feature_test)

In this function, LinearSVC (C = 1.31) is called to classify the testing data, but the training label need to be binary in order to fit this model, so it is important to preprocess the training label and fit different LinearSVC models, and use these models to predict all the testing data and choose the largest one in all predicted results as the final prediction.

9. confusion, accuracy = classify_svm_bow(label_classes, label_train_list, img_train_list, label_test_list, img_test_list)

The steps in this function are almost the same with classify_knn_bow. The accuracy obtained in this classification is 0.619.

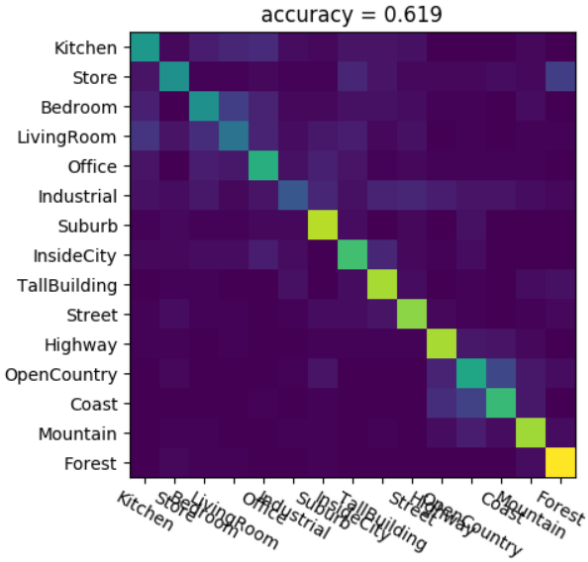

FIG.3 BOW and SVM