CSCI 5561

Homework 3 – Scene Recognition

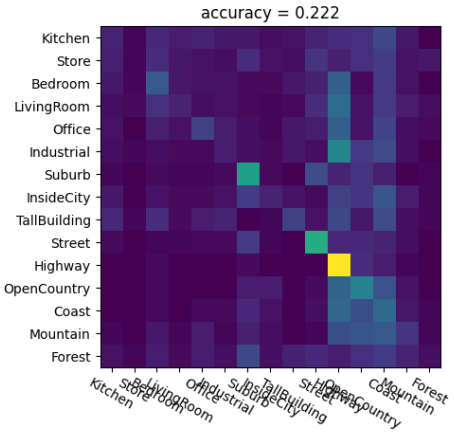
He Zhou

November 5, 2021

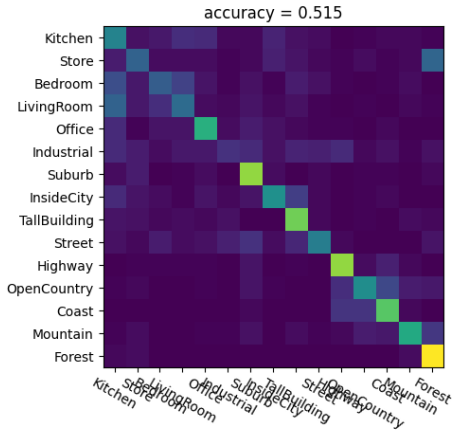
The scene classification dataset consists of 15 scene categories including office, kitchen, forest and so on. The visual recognition system will compute a set of image representations (tiny image and bag-of-word visual vocabulary) and predict the category of each testing image using the classifiers (k-nearest neighbor and SVM).

Tiny Image KNN Classification: The Tiny Image approach uses the function `get_tiny_image` to resize each image to a small, fixed resolution using function `cv2.resize` and normalize the tiny image to having zero mean and unit length. Given the tiny image representations, the function `predict_knn` uses a k-nearest neighbor classifier (`sklearn.neighbors.NearestNeighbors`) to predict the label of the testing data. The number of neighbors for label prediction is set to be $k = 10$ and label is predicted as the voted majority of the labels of those 10 nearest neighbors. Based on the predicted labels on the testing data, the function `classify_knn_tiny` returns a 15×15 confusion matrix and the accuracy of the testing data prediction.

The visualization of confusion matrix is given in Figure (1a) and the accuracy of the prediction is 0.222. We can see that the tiny image representation is not a particularly good representation. This is because it discards all of the high frequency image content and is not especially invariant to spatial or brightness shifts.



(1a) Tiny image KNN classification with accuracy 0.222



(1b) Bag-of-word KNN classification with accuracy 0.515

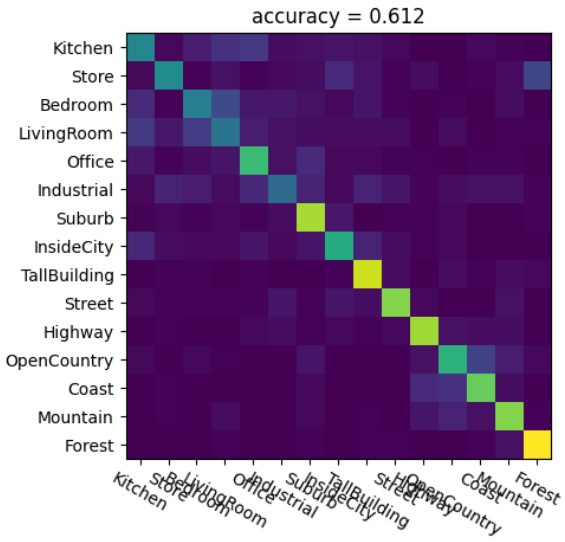
BoW + KNN: The function `compute_dsift` divides each image into small patches with both length and width equal to `stride` = 20, and the keypoint for the sift descriptor is set to be the right botton corner of each small patch with keypoint diameter equals to `size` = 20. Then this function returns a collection of sift features on the dense set of locations on image. Given the list of dense sift feature representation of training images, the function `build_visual_dictionary` builds a visual dictionary made of quantized SIFT features with `dic_size` = 50, in which the function `sklearn.cluster.KMeans`(..., `n_init` = 20, `max_iter` = 300) is used to find the cluster centers. The

output `vocab` is saved in the file `'vacob_knn_bow.txt'`. In function `compute_bow`, the bag-of-word (BoW) feature is constructed by counting SIFT features that fall into each cluster of the vocabulary, in which nearest neighbor with `n_neighbors` = 1 is used to find the closest cluster center. Given the BoW features, the function `predict_knn` again uses the k-nearest neighbor classifier with $k = 10$ to predict the testing labels.

The visualization of confusion matrix is given in Figure (1b) and the accuracy of the prediction is 0.515. We can see that the BoW representation is a better representation.

BoW + SVM We use the BoW representation with `stride` = 15 and `size` = 15. The output `vocab` is saved in the file `'vacob_svm_bow.txt'`. Given the BoW features, function `predict_svm` uses a SVM classifier `sklearn.svm.LinearSVC` to predict the label of the testing data. We train 15 binary, 1-vs-all SVMs, where 1-vs-all means that each classifier will be trained to recognize 'forest' vs 'non-forest', 'kitchen' vs 'non-kitchen', etc. All 15 binary SVM classifiers are evaluated on each test case and the classifier with the highest decision function score (given by `decision_function`) wins. The free regularization parameter 'lambda' is set to be $C = 5$ to obtain good classification performance.

The visualization of confusion matrix is given in Figure (2) and the accuracy of the prediction is 0.612. We can see that the BoW representation with SVM classifier performs the best.



(2) Bag-of-word SVM classification with accuracy 0.612

References

S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 2169–2178.