Exercise 2 : Image Classification

1 Bag-of-Words Classification with Histograms of Oriented Gradients

In this exercise, you will build an image classifier based on the bag-of-words (BoW) approach and the nearest neighbors algorithm.

1.1 Dataset

We will use the STL-10 dataset for training and testing our model. STL-10 contains 96 $\times$ 96 color images that belong to one of the following ten semantic classes : airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck. It contains a training set with 500 images from each class (5000 images in total) and a test set with 800 images from each class (8000 images in total). The training set is used to optimize the model parameters, whereas the test set is used to evaluate the model performance.

The dataset can be downloaded from the link below :

`https://drive.google.com/file/d/1AUc2-xjsTyj7tSrVbEoGfIuzi3Bnz954/view?usp=share_link`

1.2 Bag-of-Words Classification Pipeline Overview

The BoW pipeline is illustrated in Figure 1. At training time, HOG features are extracted densely from each image. K-means clustering is applied to the set of extracted features from the entire training set, resulting in K centroids which serve as the visual words. Each training image is finally represented as a bag of such words in the form of a BoW histogram vector, based on the proximity of its HOG feature vectors with the K visual words. At testing time, the aforementioned steps for computing a BoW histogram are applied to each test image. The final classifier is nonparametric and it is based on the nearest neighbor principle : the label of the nearest neighbor of the test image in the training set (based on BoW histogram representation of images) is predicted to be the label of the test image. More formally, denote the training set as a set of pairs of BoW histograms and class labels

$$\mathcal{T} = \{(\mathbf{h}_i, c_i) : i = 1, \ldots, T, c_i \in \{1, \ldots, C\}\}, \tag{1}$$

where C is the total number of semantic classes we consider. For a test image that is represented as a BoW histogram $\mathbf{g}$, the nearest neighbor classifier predicts the label

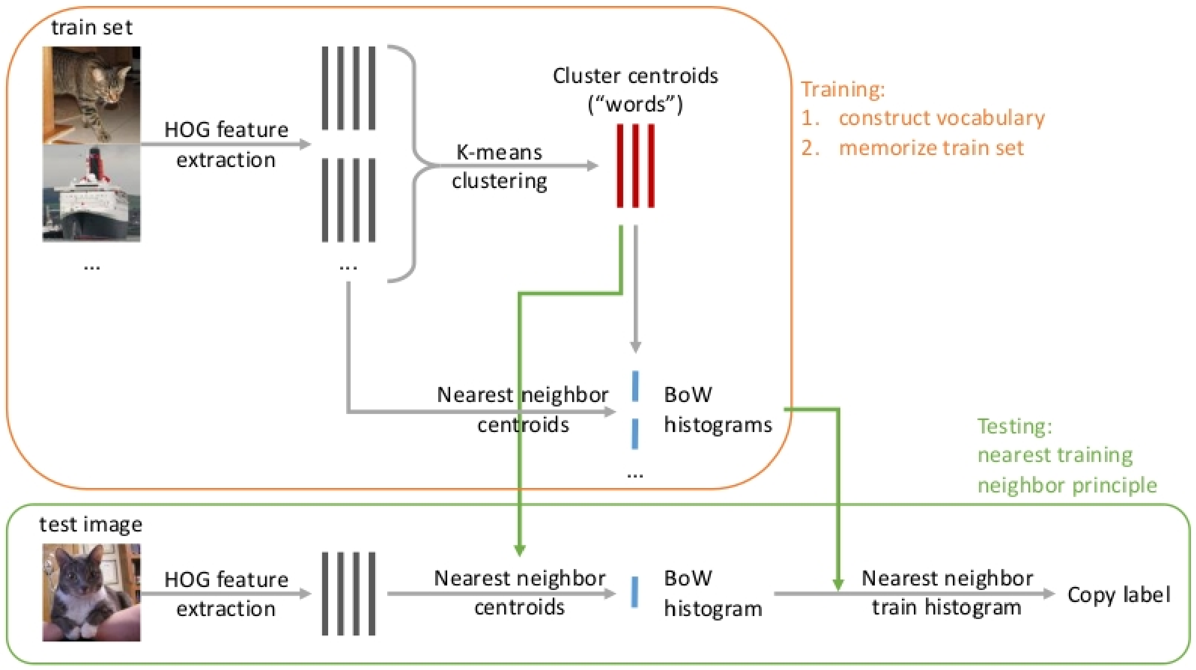

FIGURE 1 – Overview of BoW pipeline with HOG features and nearest neighbor classifier.

$$\hat{c} = c_{\hat{k}}, \tag{2}$$

where $\hat{k} = \arg\min_{k \in \{1,\dots,T\}} \{d(\mathbf{g}, \mathbf{h}_k)\}$, $d(.,.)$ denotes some distance metric between two BoW histograms.

Question : Suppose that we do not use any approximation technique (e.g. K-D tree) to determine the nearest neighbor in the training set, but compute all required distances exactly. What is the space and time complexity of predicting the label of a single test image with respect to the size of the training set ?

1.3 Feature Description with Histograms of Oriented Gradients (HOG)

The first step in the BoW pipeline is feature extraction for a given image. Details of this step are given below :

— **Step 1 :** Define a dense regular grid of $n\_points\_x \times n\_points\_y$ points on the image. Write a function that outputs the column (x) indices and the corresponding row (y) indices of the grid points as two separate 1D NumPy arrays of length $n\_points\_x \times n\_points\_y$. (**Hint :** You might use the functions numpy.linspace and numpy.meshgrid).

— **Step 2 :** Compute HOG feature descriptors for all grid points from the previous step. For each point, the HOG feature description operates on a square image patch centered at the point and partitioned into a 4×4 set of cells. Each cell has a size of $cell\_height \times cell\_width$ pixels. In our case, $cell\_height = cell\_width = 4$, so the entire square patch has a size of 16×16 pixels. Compute the approximate image gradient using the Sobel operator and get the direction of the gradient as an angle in the interval $[-\pi, \pi]$. For each grid point, compute the 8-bin histogram of gradient directions at pixels that belong to each separate cell of the corresponding patch and concatenate the histograms from all the cells into the final 128-D HOG descriptor for the point. Write a function that outputs a 2D NumPy array of size $n\_points \times 128$, where n_points is the total number of grid points defined on the image (**Hint :** You might use the functions scipy.ndimage.sobel and numpy.histogram).

1.4 Visual Vocabulary Construction with K-means Clustering

The second step in the BoW pipeline is to construct a vocabulary (or codebook) of K visual words by applying K-means clustering to the complete set of HOG descriptors from all images in the training set. The centroids which are computed with K-means serve as the visual codebook words. The K-means algorithm follows the steps below :

— **Step 1 :** Initialize random cluster centroids.

— **Step 2 :** Assign each data point to the cluster whose centroid is nearest to the point with respect to the Euclidean distance.

— **Step 3 :** Update each cluster centroid to the mean of all data points that are currently assigned to the cluster.

Steps 2 and 3 are repeated until convergence. Write a function for the K-means algorithm that outputs a 2D NumPy array of size K $\times$ 128, containing the final cluster centroids as its rows (**Hint :** You might use the functions numpy.random.permutation and scipy.spatial.distance.cdist).

1.5 BoW Histogram Representation

The third step in the BoW pipeline is to compute the BoW histogram for each image, which serves as a high-level representation of the image. The BoW histogram is a K-bin histogram, where the k-th bin counts how many HOG descriptors of the input image are closest to the k-th codebook word (centroid). Write a function that computes the BoW histograms for all images belonging to a set (train or test), along with creation of an array containing the ground truth labels of these images (**Hint :** You might reuse the function implemented for the previous task and the function numpy.bincount).

1.6 Nearest Neighbor Classifier

The fourth step in the BoW pipeline is to build a nearest neighbor classifier. Write a function that outputs a 1D NumPy array containing the predicted labels for the corresponding test BoW histograms. We'll use the Euclidean (L_2) norm to measure distances between BoW histograms (**Hint :** You might reuse the function implemented for the previous task).

1.7 Experiments

— Evaluate the accuracy of your BoW model on the test set with the following settings : the number of K-means clusters K is set to 100, the number of K-means iterations is set to 10. The performance should be evaluated over multiple runs (the number of runs is set to 10). Consider only two classes for this evaluation. Why are multiple runs of training and evaluation necessary when reporting accuracy ?

— Evaluate the performance of your model on the full STL-10 dataset, i.e. for all ten classes, with the same settings as above. Measure the reported accuracy on the test set and compare it to the previous case with only two classes. Is the accuracy for ten classes expected to be higher or lower, and why ? Implement a random classifier and compare its performance to that of your model in the two cases.

— Perform the same experiments as above where the L_1 norm is used to measure distances between BoW histograms. Compare the obtained performance to the L_2 norm baseline with different settings of K, i.e. $K = 50, 100, 200, 400$.