Computer Vision

Chennai Mathematical Institute

Assignment 4

Task

The main task is to do image classification.Specifically, we want to examine the task of object recognition using the Bag of Features approach. We want to classify images into one of 8 given categories by training and testing on a subset of the CalTech-101 database. We will also evaluate the performance of the image recognition system on the test data provided.

Submitted Files

1. PA4_utils.py
2. Bag_of_features_code.py
3. Notebook
4. Documentation

Results of your interest point detector

Here are some result of interest point detector in 4 image categories -

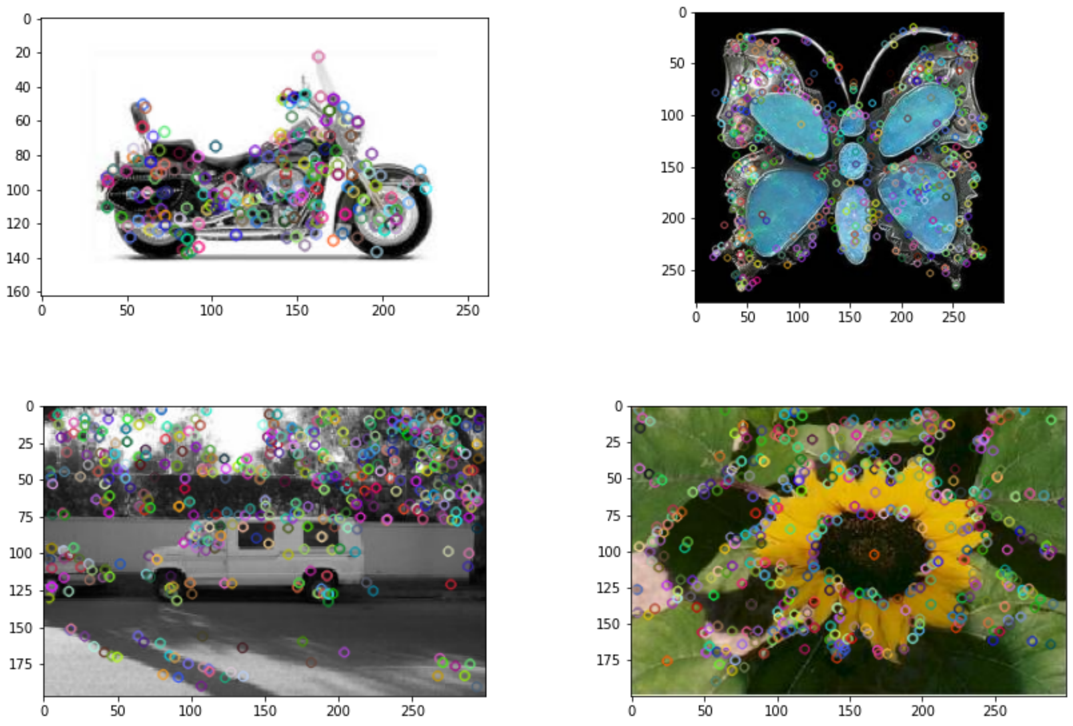

Figure 1: Interest points for 4 different category images

Detailed report

We have two sets of images. (a) Training set and (b)Testing set
The first task was to build vocabulary. Here the vocabulary contains the centroids of k-means clustering on the descriptors. The centroids of the clusters will serve as the visual words in our dictionary.

Step -1:

- First we detect the keypoints and compute the sift descriptors.
  Library Used - Opencv

  - We can limit the number of keypoints we extract from each image, in this dictionary building phase. We have one parameter choice while using opencv for this task. We change the parameter "nfeatures" according to our choice of no of keypoints for an image.

- We get an array which contains the sift descriptors for all training images.
- Now we form clusters of the descriptors using k-means algorithm. We give our choice of k. We can also plot an elbow curve and get an optimal value of k.
  Library Used - sklearn
- Then we find the centroids of the clusters and save it as a vocab file. The centroids of the clusters will serve as the visual words in our dictionary.

Step -2

Here we will compute the feature vector i.e. the frequency histogram of visual words for each training image.

- First We find the keypoints and their SIFT descriptors for each training image.

  Library Used - Opencv

  -Again, we can threshold the number of keypoints we consider for each image for this step. But here the sampling is denser from the first step.

- Then we build the histogram of visual words by determining the closest visual word that corresponds to each keypoint.

  Library Used - Scipy

  - It returns the assignments i.e. the class in which the descriptor belongs. Then from that we compute the histogram counts.

  - Since the images are of differing sizes, the total count in histogram will vary from image to image. To account for this, L1 normalization is used in the histogram.

- So for each image we have got a histogram which is the vector representation of the image. Here the histogram size which is equal to the no of clusters is same for all images.

Step -3

K-nearest neighbour recognition system :

Here we use the image features obtained from step-2. We fit the model on train image features and predict the category of test images based on the test image features.

Library Used - sklearn

Parameters :

1. No of neighbours considered

2. weights: Two choices - (a)uniform (b)distance

   - 'uniform' : uniform weights. All points in each neighborhood are weighted equally.

   - 'distance' : weight points by the inverse of their distance. in this case, closer neighbors of a query point will have a greater influence than neighbors which are further away.

3. distance metric - For this we just change the parameter "p" which is Power parameter for the Minkowski metric.

   - If we set p as 1 then it is Manhattan distance (l1).

   - If we set p as 2 then it is Euclidean distance (l2)

Step -4

SVM based recognition system :

Here we use the image features obtained from step-2. We fit the model on train image features and predict test images based on the test image features.

Library Used - sklearn

Parameters while using SVC from sklearn:

1. random_state
2. tol - Tolerance for stopping criterion.
3. kernel - We can choose the kernel we want.

   - linear - Uses linear kernel.

   - poly - Uses polynomial kernel with degree set to be 3(default).
     We can change "degree" parameter based on our choice of degree of the ploynomial kernel.

   - rbf - Uses Radial Basis Function Kernel

4. decision_function_shape - Whether to return a one-vs-rest ('ovr') decision function of shape (n_samples, n_classes) as all other classifiers, or the original one-vs-one ('ovo') decision function of libsvm which has shape (n_samples, n_classes * (n_classes - 1) / 2). Default is one vs rest("ovr")

5. break_ties - (default=False) If true, decision_function_shape='ovr', and number of classes ¿ 2, predict will break ties according to the confidence values of decision_function; otherwise the first class among the tied classes is returned. Please note that breaking ties comes at a relatively high computational cost compared to a simple predict.

6. C - Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.Default is set to be 1.0.

Parameters while using LinearSVC from sklearn:

1. random_state
2. tol - Tolerance for stopping criterion.
3. C - Regularization parameter. The strength of the regularization is inversely proportional to C. Must be strictly positive. The penalty is a squared l2 penalty.Default is set to be 1.0.
4. multi_class - Set to be "ovr" i.e one vs rest.

Output of K-Nearest Neighbor recognition system

Results for different choices of K and distance metric :

1. Results for K = 1

Design choices -
Clusters - 250
Max no of keypoints considered for each image to form clusters - 200
Max no of keypoints considered while computing histogram-350

- Accuracy : Using Manhattan Distance - 56.25% ,Using Euclidean Distance - 55%
- Confusion matrix:

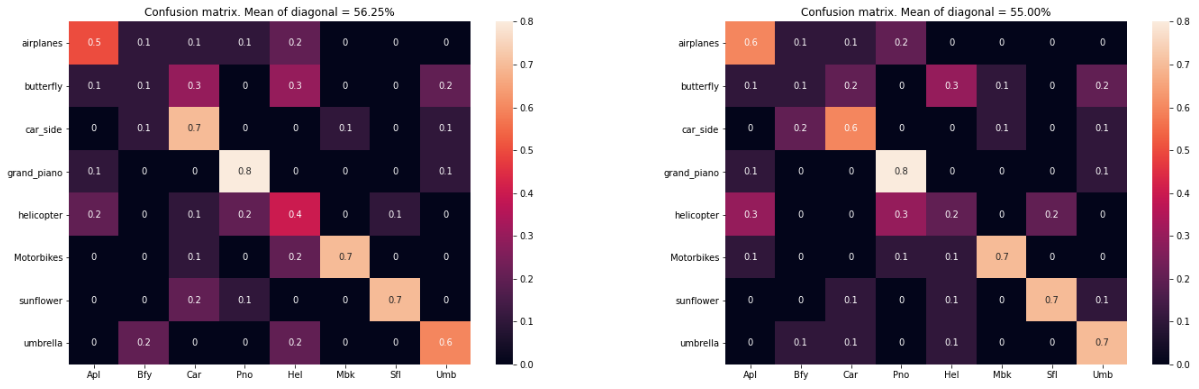

Figure 2: (Left: Using Manhattan Distance, Right: Using Euclidean Distance)

2. Results for K = 11

Design choices -
Clusters - 250
Max no of keypoints considered for each image to form clusters - 200
Max no of keypoints considered while computing histogram-350

- Accuracy : Using Manhattan Distance - 65% ,Using Euclidean Distance - 62.50%
- Confusion matrix:

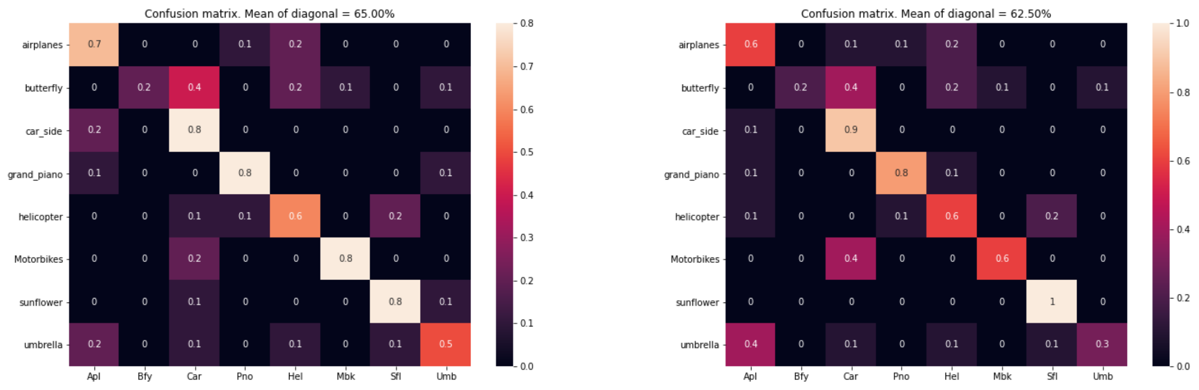

Figure 3: (Left: Using Manhattan Distance, Right: Using Euclidean Distance)

Distance metrics: If we have two vectors then distance between these vectors can be computed using several ways. In general we can compute L_m distance. We can choose m as we wish. If m=1 then it is called Manhattan Distance and If m=2 then it is called Euclidean distance.
In general, formula for L_m distance, $d\left(p,q\right) = \left(\sum_{i=1}^{n} |q_i - p_i|^m\right)^{\frac{1}{m}}$
where p and q are two n dimensional vectors.

In KNN if we find that two neighbours k1 and k2 has identical distances but different labels then based on the order of the training data we output the result.

Output of linear-SVM based recognition system

Both Linear SVM and SVM with different kernels are tried out.

Results for different choices of lambda(Regularization paramter)

Design choices -
Clusters - 250
Max no of keypoints considered for each image to form clusters - 200
Max no of keypoints considered while computing histogram-350

- **Accuracy :** For Lambda value,5 - 71.25% ,For lambda value ,8 - 72.50% , For Lambda value,11 - 73.75%
  With small lambda value the accuracy is less.
- Confusion matrix:

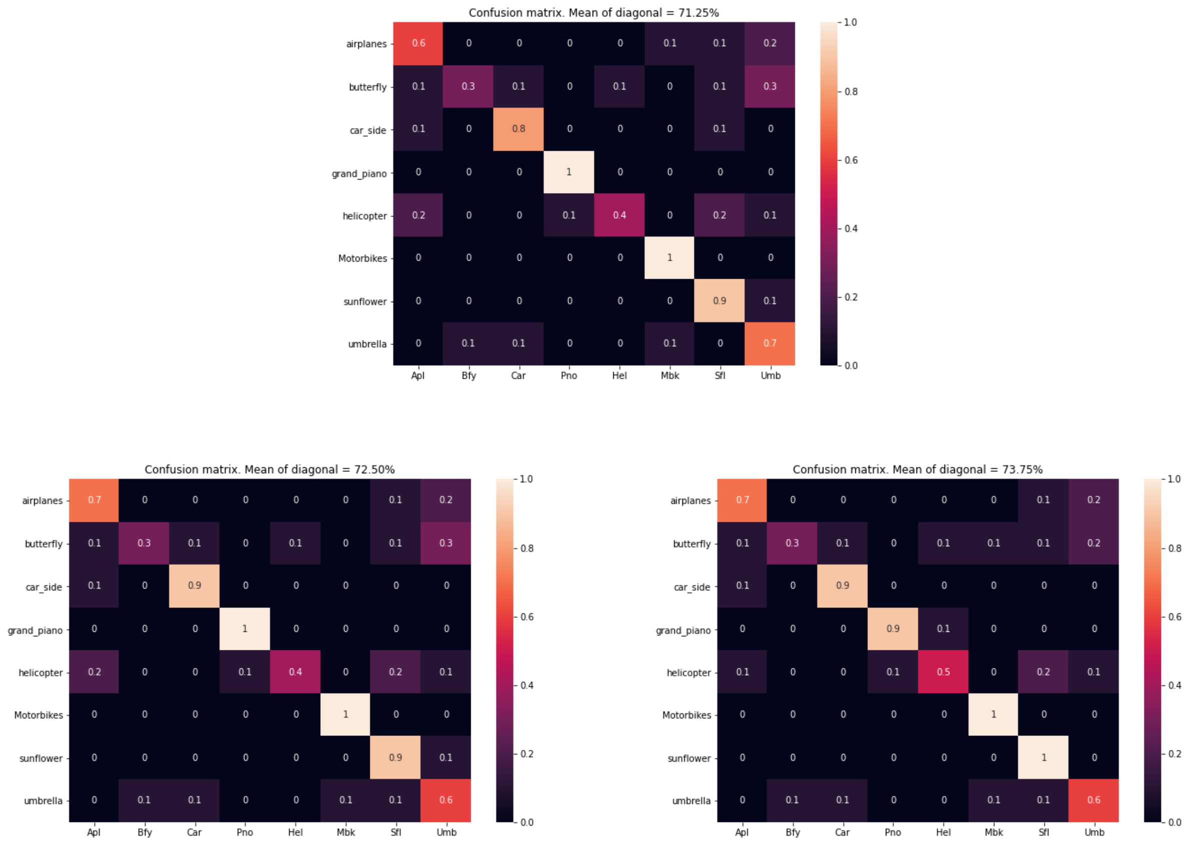

Figure 4: For different values of Regularization parameter

For some design choices, with increase in value of lambda, accuracy is getting decreased and for some design choices the accuracy is getting increased with lambda.

Some Other good results

Design choices -
Clusters - 250
Max no of keypoints considered for each image to form clusters - 250
Max no of keypoints considered while computing histogram-350

For KNN, With K=11 the accuracy obatained is 68.75% and SVM achieved accuracy of 75% with lambda as 8.

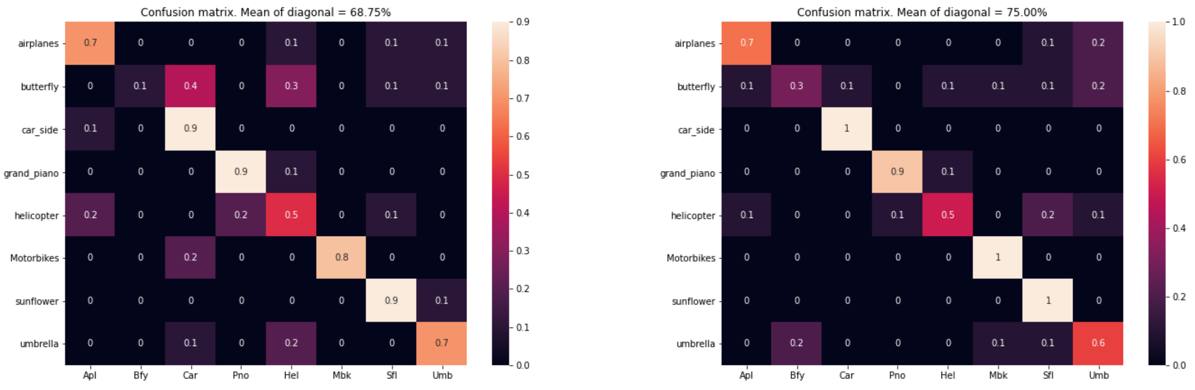

Figure 5: Left:KNN result, Right:Linear SVM result

Performence comparison

Overall Linear SVM worked better. The accuracy obtained in case of KNN is less compared to Linear SVM. Again for KNN if we take only one one nearest neighbour then the accuracy obtained is less than what we are getting with higher values of K. For K=11 the performence was good compared to other values of k for particular design choices(No of clusters etc.). Again if we use SVM with rbf kernel of polynomial kernel then the accuracy is increasing slightly. Performence of the two models, KNN and Linear SVM highly depends on the data we are working with. If classes are linearly separable then Linear SVM works better. KNN will beat a linear SVM when the data is not linear. KNN performs well when we have too many data points and few dimensions. Again SVM can be thought as an improved version of KNN as we only keep the points that are near the frontier for classification (support vectors). So for these reasons Linear SVM worked better than KNN in our case.

Extra Credit

TF-IDF is implemented. We have considered the counts(term frequencies) while computing the histograms for each image. But considering only TF is not a good idea as some of the visual word may occur in most of the images and it may be not be right to just classify a image based on the term frequencies of the visual word. As this visual word occurs in most of the images we may want to give less weight to this visual word while classifying an image. So it will be a good idea to use TF-IDF instead of TF.
We have the histograms for each image. Now for each visual word we can check in how many images that visual word has occured(In how many images the visual word has TF greater than zero). In this way we can get the DF. From DF we compute the IDF and then compute TF-IDF. Using TF-IDF performence increased in most cases.