Comupter Vision

Subhasish Basak (MDS201803)

Assignment 4

April 5, 2020

1 Part 4 : Results and Observations

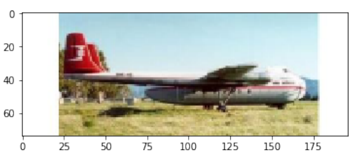
(a)(2 points) Results of your interest point detector on one image from any 3 categories. You can reuse the function provided to you as part of Assignment 3 or use a library function.

1.1 Interest point detection of images

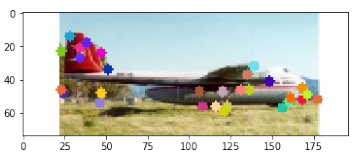
- ***Categories*** : Here we choose the following 3 categories, viz. **airplanes**, **butterfly**, **helicopter**.
- ***Function used*** : Here we use the implementation from our previous *Assignment 3*. We use the function **get_interest_points()** which takes the *image* and *feature_width* as arguments and returns the co-ordinates of the detected interest points.

Throughout our experiments we have used the following parameter values:

- σ for Gaussian Kernel = 1
- κ for Harris response = 0.06
- Robustness parameter of ANMS = 0.9
- Neighbourhood threshold = 0.009
- Neighbour checking window size = 3×3

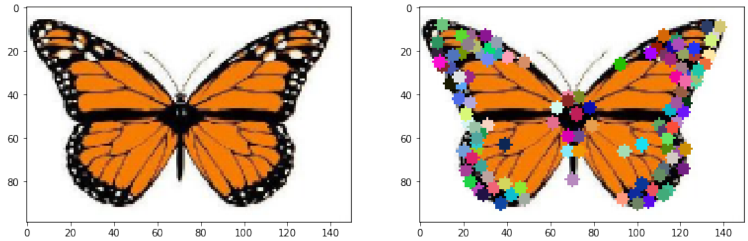


(a) Original

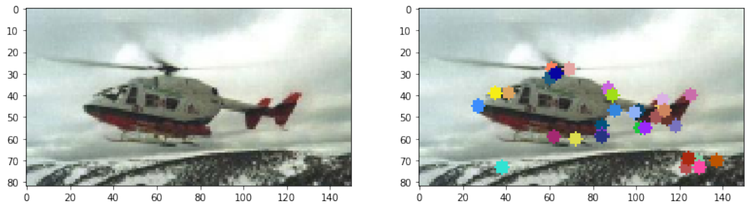


(b) with corners

- For the **Airplane** image total number of feature points after Neighbourhood check was 67. After applying the Feature width constraint we reduce another 26 key-points.
- For the **Butterfly** image total number of feature points after Neighbourhood check was 178. After applying the Feature width constraint we reduce another 140 key-points.
- For the **Helicopter** image total number of feature points after Neighbourhood check was 115. After applying the Feature width constraint we reduce another 62 key-points.



(a) Original (b) with corners



(a) Original (b) with corners

(b)(10 points) Detailed report on the design choices, including dictionary size, similarity and distance metrics used, hyper-parameter selection and the experiments that support your choices.

1.2 Image classification using Bag of features

The main part of *Assignment* 4 was image classification using the **Bag of features** approach. We briefly summarize the steps we have followed:

- Finding the SIFT vector representation of the interest points from all the training images.
- Using K-means clustering algorithm we identify the **K cluster centers** from those SIFT vectors.
- Build frequency histogram of size **K** for each of the training images. Here we also need to normalize the histograms (**L1 normalization**) since the different sized images will have different number of key-points.
- For classifying the test images we use one of the following approaches,
  - **K-NN approach** : In this approach, we classify each test image based on the distance of it from it K nearest neighbours. Here distance referred to the Euclidean distance of the frequency histogram vectors.
  - **SVM approach** : Here we use the support vector machine to classify each image based in the one vs rest approach.

Design of experiment

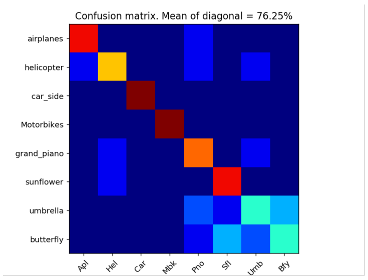
- **Step size** : We have used the inbuilt package of ***vlfeat*** to compute the SIFT descriptors. It includes a **step** parameter which determines the sampling frequency among the adjacent pixels of the image. Here we have kept the step size as 2.
- **No. of key-points** : While building the vocabulary we have to decide upon the number of key-points we need to consider for each image.
  While building the vocabulary we have taken the threshold as 50% of the total detected key-points. While building the feature histogram for each training image we have set the threshold as 75% of the total detected key-points, which is quiet naturally denser than the previous case.
- **Cluster size (K)** : Or the **dictionary size**, which indicates the number of cluster centers in which we divide the whole vocabulary of features. In our experiment we have started with $K = 100$ and later increased to $K = 300$. As K increases the computational time also increases but it also increases the accuracy of the prediction.

- **K-NN neighbourhood size** : Another parameter of our experiment is the size of the neighbourhood for the K-Nearest Neighbourhood classification algorithm. An increase in neighbourhood size from 3 to 5 has resulted in a jump from 76.25% to 80% in accuracy. It again decreases when neighbourhood size is increased above 5.
- **Distance measure** : In the K-NN approach we have used the ***Eucledian*** distance among the feature histogram vectors.
- **SVM free parameter** : There is a free parameter (say λ) in the SVM implementation which controls the how strongly regularized the model is. A change in the value of $\lambda = 0.001$ to $\lambda = 0.1$ increases the accuracy of the model from 76.25% to 81.25%.

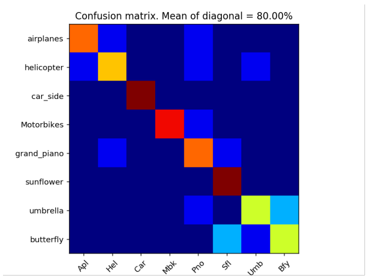
(c)(3 points) Include the output of K-Nearest Neighbor recognition system for 2 values of K (K=1, and the best performing among the other values of K you experimented with) and the two suggested distance metrics (4 confusion matrices and accuracies, in all). How do the distance metrics compare? How did you choose to resolve ties?

1.3 K-NN Results

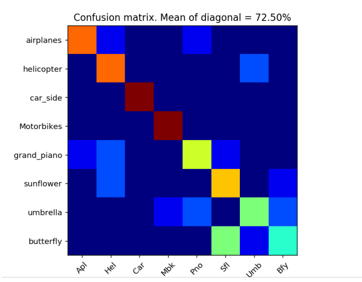
Here we present the confusion matrices and their accuracies.



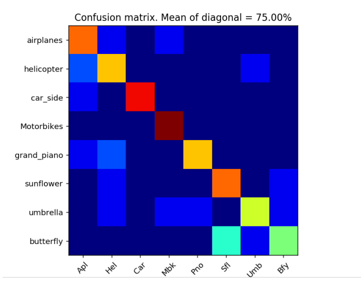
(a) K = 1 and Euclidean dist.
accuracy 76.25%



(b) K = 5 and Euclidean dist.
accuracy 80%



(a) K = 1 and Chi sq. dist.
accuracy 72.5%



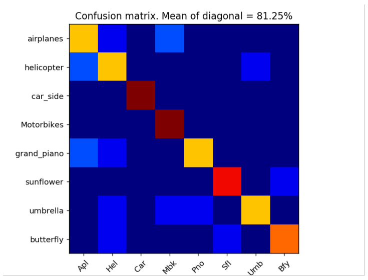
(b) K = 5 and Chi sq. dist.
accuracy 75%

- **Distance metrics** : Among the 2 distance metrices viz. **Euclidean** and **Chi-square** we have seen that the *Euclidean* metric has performed better that the *Chi-square* metric.
- **Tie breaking** : We written a function which selects k minimum elements from a list. For tie cases it returns the first k minimum elements in the order as they are in the given list.

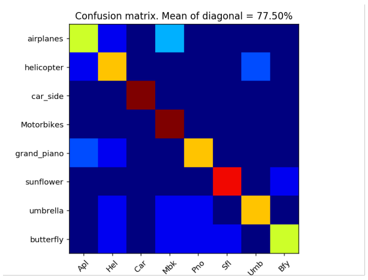
(d)(2 points) Include the output of your linear-SVM based recognition system for 3 values of λ, including the one that yielded the best results in your experiments (3 confusion matrices and accuracies, in all).

1.4 Linear SVM results

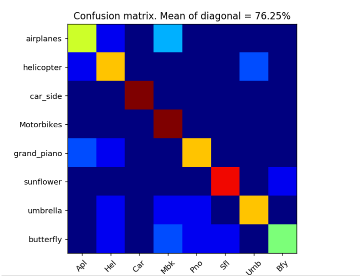
Here we present the confusion matrices and their accuracies. For λ values 0.1,0.01 and 0.001.



(a) $\lambda = 0.1$
accuracy 81.25%



(b) $\lambda = 0.01$
accuracy 77.5%



(c) $\lambda = 0.001$
accuracy 76.25%

(e)(3 points) Which of the 2 recognition systems performed better? What might be the reason for this?

1.5 Comparison of the 2 recognition systems

Both the approaches performed well but the maximum accuracy was achieved using **SVM** that is 81.25%. The overall performance of the **K-NN** based recognition system was better than the **Linear SVM** based approach.
K-NN classifies data based on the distance metric whereas the performance of **Linear SVM** is based on training (for our purpose we have used the *multi-class SVM* with One-vs-All approach). The choice of the distance metric also affects the performance (as we have seen in case of *Euclidean* and *Chi-sq* metric), which may be a plausible cause for its lower performance.

1.6 Extra Credit : TF-IDF implementation

We implement the TF-IDF approach in constructing the feature histograms, it is implemented inside the function ***get_bags_of_sifts***.
In our case the images are the documents and in this context we define,

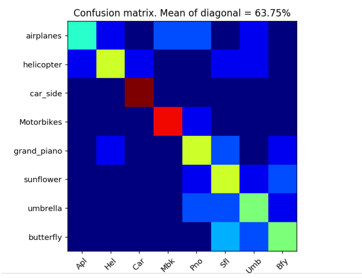
- Term-Frequency(**TF**) : For a perticular cluster point (visual words) the number of key-points descriptors that are assigned to it.

- Document-Frequency(**DF**) : For a perticular cluster point (visual words) the number of images from which it has descriptors assigned to it.

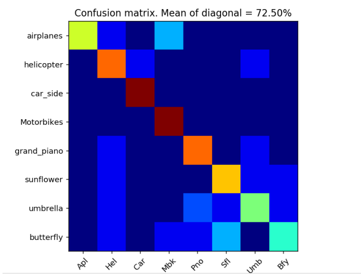
We can compute the **DF** from **TF**. For all the images we scan through its **TF**, i.e. the features histogram. If the *i-th* element of the features histogram is positive we increase the **DF** corresponding to the *i-th* cluster point (visual word) by 1. We do this iterative operation for all the images and at the end we get the **DF** list. Once we have the **DF** list we can compute the **IDF** list using the following formula,

$$IDF_w = \log \frac{T}{DF_w} \tag{1}$$

Here, $T = 520$, i.e. the total number of training images. Although the results did not improve after the implementation, here are the confusion matrices along with the accuracies.



(a) KNN
$K = 5$, accuracy 63.75%



(b) SVM
$\lambda = 0.01$, accuracy 72.5%