Assignment 3

bansalro-iganjoo-molasia

Note: Please ensure that the svm multiclass files are available on your system. We have included the make command for svm files in our make file, and have also uploaded the modified Sift.h file.

Simple Baseline

The simple baseline approach gives better results than random guessing.
We got an accuracy of **24%** in the simple baseline using color.
Not using the color gives us lesser features and therefore the results were not as good as that with color. Using gray scale images, we got an accuracy ranging between **11%-16%**.

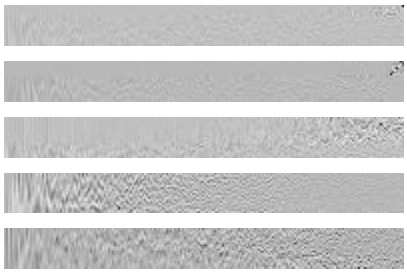
To run from command line:
./a3 train baseline
./a3 test baseline

Eigenfood

To calculate the eigen vectors and eigen values, we followed the below steps:

- Resize the input image into a square matrix
- Compute the covariance of the input image.
- Perform eigen decomposition on the covariance matrix.
- This returns a list of eigen values and eigen vectors.
- Next, we chose top k eigen vectors from the list obtained, and tested multiple times.

We plotted the top eigenvectors as images and below are some of the results obtained as shown below:
The size of the below images depend on the number of features i.e. the eigen vectors that we decide to take up from an image and on the dimensions of the original image.
In the below case, we took an image size of 200*200 and the number of eigen vectors per image were 20, hence we got a 200x20 image.



The code used for plotting the above images has been commented, if you wish to see the plotted images, you may uncomment the code.(Uncomment in PCA.h)

After observing the eigen values, we found that initially for approximately top 10 eigen values, the values are decreasing at the rate of approximately 8% but as we observe the latter values, the rate of change becomes almost negligible.
Hence, we deduced that the eigen values are decrease at a much faster rate at the top and the rate of change decreases when we move down the list of eigen values.
Using eigen decomposition and training our SVM using the top k eigen vectors, we are getting an accuracy of **6.2%**.

To run from command line:
./a3 train eigen
./a3 test eigen

Haar-like features:

For Haar-like features, we used four types of 4x4 and 2x2 filters like the ones shown below:

The steps for this part are as follows:

1. Firstly, we converted our image into an integral image which will make computation faster.
2. We placed all the configurations of the rectangular filters on the integral image and shifted it by a stride of 2 for a 2*2 filter, and a stride of 4 for a 4*4 filter.
3. We performed the sum and difference on all the above locations and calculated haar like features for all the possible filter sizes and configurations. This was done for an image size of 500x500, hence the number of features were approximately 25k.
4. We trained our SVM using these 25k features and then tested it which gave us a meagre accuracy of just 14%

The accuracy of this technique ranges from **6% - 14%** and it largely depends on the number of features we are using, the image size and the different type of Haar-like features we are using.

To run from command line:
./a3 train haar
./a3 test haar

Bag-of-words
We stored the descriptors of all the training images in a csv file which was around 750 megabytes. Mlpack is used to calculate 25 centroids from the bag of descriptors. (Source:http://www.mlpack.org/docs/mlpack-2.2.0)
We have commented the code for the bag of descriptors and centroid calculation as it was taking too long. We have uploaded centroids.csv which is used to create histogram for training and testing.We got an accuracy of **34%** with the bag of words features.

In your report, include an experimental comparison (quantitative and qualitative) of the two approaches you
choose with each other and the baseline from Part 1

Qualitative Comparison:

1. The Bag-of-Words definitely works better than the Haar-like features which just gave a meagre accuracy of 14% as compared to that of Bow of 34%. Increasing the number of features and the type of Haar-like features can give us an accuracy similar if not better to that of Bag-of-Words but there is always a trade off between the computation time and the accuracy. Our SVM took almost 1 hour to train for 25k features of Haar-like and increasing the number of features will be computationally very expensive and hence a lot of time.
2. The Principal Component Analysis is working very poorly as compared to both: Bag-of-Words and Haar-like features. We just used 1000 features for PCA part, increasing the number of eigen vectors may increase the accuracy but on the basis of our experiments and observations, that does not help very much and hence PCA performs poorly if compared to the former two.

Quantitative Comparison:

1. Increasing the number of features and the type of Haar-like features can increase the accuracy but there's a trade off between the computation time and the accuracy. The Bag-of-Words is by far working with an accuracy of 34% and the Haar-like can catch up to that if we increase the number of features.
2. Increasing the number of eigen vectors as features does not alter the accuracy by much and based on our observations we can say that the PCA is the worst performer of all of the three traditional features.

To run from command line:
./a3 train bow
./a3 test bow

Deep features

We used an image of size 231*231 which is the minimum size required by overfeat. This gave us around 4100 features per training image. The Deep model works better

quantitatively and qualitatively than all other methods. The results are better than all the other methods implemented above.

Qualitative and Quantitative Analysis:
The accuracy is around **70%** for the deep model. Increasing the size of the image from 231 may give better results, because overfeat will give more features per image. But we limited our experimentation to the minimum possible size acceptable by overfeat to avoid huge computation time.

Note: The testing for deep model will take a long time(1 minute per image) because it will read the model file (around 300 megabytes) for every test image.

To run from command line:
./a3 train deep
./a3 test deep