CSL462 - Bag of words matching and homography (Assignment 3)

Shivam Mittal
www.linkedin.com/in/shivam-mittal

Indian Institute of Technology,
IIT Ropar

1 Abstract

The assignment was divided into 2 parts. The first task was to create a bag-of-words based matching/categorization solution on the MNIST-fashion database [1]. We created a global descriptor for each of the images. The descriptor was inspired from the HOG descriptor. Each descriptor was treated as a feature, and then k-means clustering was used to find the cluster centers which represented our visual words, from which a visual dictionary was created. Using the visual dictionary, histograms were created for all the training and test images by using soft-assignment (weights were assigned to all the visual words based on the distance of that word from the image feature). Then for each test image, we matched the histogram of the test image with all the histograms of the training images using chi-squared distance. A fixed number of training images (50 chosen empirically) whose histogram had the minimum distance from test histogram were considered, each of those labels were considered and the label which had the majority vote was taken as the predicted label. We achieved an accuracy of 81.13% on the test set which can be considered as a good result considering only traditional computer vision techniques, and no ML based pipeline.
The other task was to create a panorama stitching for two images. The program takes in 2 input images and lets the user select 4 points in both the images. Then the homography matrix for the projective transformation was computed, and one image was warped using the computer projective transformation. Then using the warped image, we stitched both images to get a mosaic.

2 Classifying MNIST-fashion database

The overall approach is described in the abstract. In the following section, each component is explained in greater detail, and the results are discussed.

2.1 Features computation

The train/test images were loaded, the features were computed for each image, and then saved for future use. A global descriptor for each $28*28$ grayscale image was computed. The feature computation step is as described:

1. For the $28*28$ image, divide the image into $8*8$ blocks of 50% overlap. Hence, we will get $6*6=36$ blocks in total.

2. For each $8*8$ block, we call a function to compute the descriptor for that patch.

   (a) The gradient magnitude and orientation is calculated for the patch image.

   (b) The gradient orientations are quantized into 9 bins.

   (c) The vote for each bin is the gradient magnitude.

   (d) The vote of the gradient magnitude is interpolated bi-linearly between the neighbouring bin centers.

   (e) We get a 9 binned histogram for one patch.

3. All the histograms for the various blocks are concatenated. Total size of the descriptor =no_of_blocks * one_histogram_dimension $= 36*9=324$. Hence we get a global descriptor of dimension 324 for each image.

The features once computed, are saved in csv file.

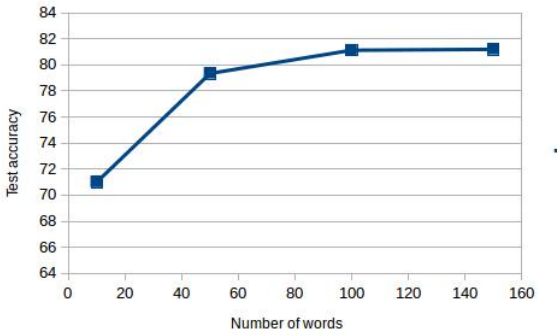

Figure 1: Choosing optimal value of number of cluster.

2.2 CreateDictionary()

Once the features are extracted, we will run k-means clustering upon the features in the feature space to get k cluster centers. Algorithm :

1. We take an initial set of k-means in the feature space by randomly sampling the data.

2. The following two steps are performed till the change (euclidean distance) in the cluster centers is less than a threshold (kept as 1.5). If we keep a high threshold, then the cluster centers may not be well distributed, and if the threshold is low, then the algorithm will take very long to converge. Hence 1.5 was set by empirical experimentation.

   (a) Each data-points (feature) in the feature space is assigned a cluster center which is nearest to it (minimum euclidean distance).

   (b) Then, we update the cluster means by taking the centroid of the data-points in that cluster.

   (c) The euclidean distance between the updated cluster means and previous cluster means is calculaed.

3. Then, we assign label to each cluster center. For each feature, we will vote its label to the closest center.

4. For each center, the label with highest votes would be the final label.

2.2.1 Choosing optimal value of number of cluster

Figure 1 shows the test accuracy obtained upon varying the number of words in the dictionary. Upon increasing the number of words, the accuracy increases but when we increase the number of words above 100, the k-means clustering takes very much time to converge, also the increase in accuracy is very less. Hence, we choose the number of words to be 100.

2.2.2 What the dictionary words represent

The most closest word to the cluster center is saved in a directory for later inspection for visualizing what the words represent. The images for those closest words are saved in Q1/words. We can see the images (closest words) are a combination of all the classes of images. We have given the analysis below.
We make a plot which can be seen in figure 2. The plot shows that for each label, there are approximately 8-12 words (cluster centers) of that label. So, each visual word is a representative word for the features of each class. There are approximately 10 (8-12) representative words (visual words) for each class. But class 6 has less number of words, so the classwise accuracy might be low for class 6.

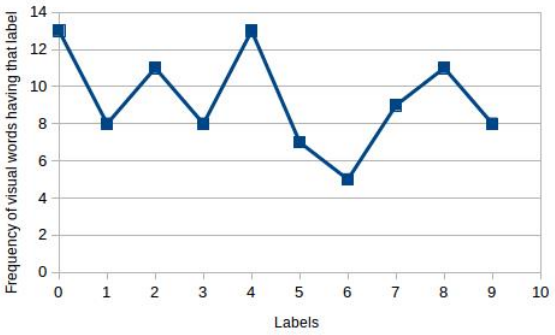

Figure 2: Frequency of words for each class label.

2.3 ComputeHistogram()

After computing the dictionary, we will use the dictionary to compute the histogram for each training and test image using soft assignment. The histogram has number of bins equal to the number of words. For each bin, we calculate the euclidean distance (in the feature space) between the feature and the visual word corresponding to that bin. We take the inverse of the distance, because that gives the similarity (the higher the value, the closer the feature is to that word). We square this similarity measure to make the distinction even more prominent between the words which are close to that feature, and the ones which are far. Then, we normalize all the bins to obtain our final histogram.

2.4 MatchHistogram()

Once, the histograms for all the training and test images are generated. We will use histogram matching to predict the labels for a test image.
For a given test image, we take its histogram and calculate the chi-squared distance between the this histogram and all the training histogram one by one. Instead of just taking the nearest neighbour, we consider 50 nearest neighbours, and then assign the class which has majority votes. This was done because it gives a boost to the test accuracy we obtain.

2.5 Results

2.5.1 Overall accuracy

The overall accuracy obtained over the test set was 81.13%. This was calculated using the formula :

$$Overallaccuracy = \frac{NumberOfInstancesCorrectlyClassified}{TotalInstances}$$

We calculate the confusion matrix for calculating all the following measures. Confusion matrix is of dimension 10*10. Its (i,j) element represent how many instances of class label i are classified as class label j. We use the following formulas for calculating the true positive (tp), false positive (fp), true negative (tn), and false negative (fn) for any class i:

$$tp = confusion\_matrix(i,i)$$

$$fp = sum(confusion\_matrix(:,i)) - confusion\_matrix(i,i)$$

$$fn = sum(confusion\_matrix(i,:)) - confusion\_matrix(i,i)$$

$$N = sum(sum(confusion\_matrix))$$

$$tn = N - (tp + fp + fn)$$

2.5.2 Classwise accuracy

The classwise accuracy is calculated using the formula :

$$classwise\_accuracy(i) = \frac{tp+tn}{N} * 100$$

The results obtained are as shown in figure 3.

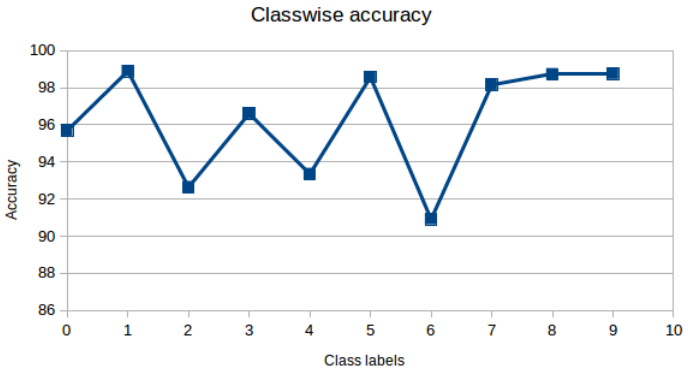

Figure 3: Classwise accuracy.

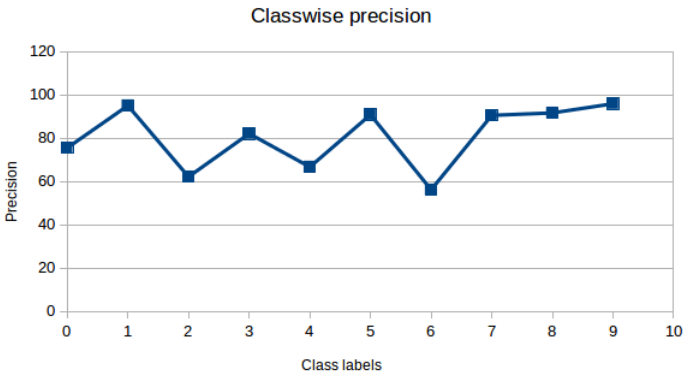

Figure 4: Classwise precision.

2.5.3 Classwise precision

The classwise precision is calculated using the formula :

$$classwise\_accuracy(i) = \frac{tp}{tp+fp} * 100$$

The results obtained are as shown in figure 4.

2.5.4 Classwise recall

The classwise recall is calculated using the formula :

$$classwise\_accuracy(i) = \frac{tp}{tp+fn} * 100$$

The results obtained are as shown in figure 5.

As we can see the accuracy, precision and recall for the 6th class is the lowest. This maybe because of the fact that the feature representation we chose was confusing the 6th class (class corresponding to shirts) with other classes. So, there was no distinguishing feature in our feature representation which could adequately different shirt from other classes. This can be also be seen from figure 2 which shows the frequency of words for each label, we see that class 6 has the least number of words representing it.
Note, if you compute the dictionary again and then get the results, it might vary slightly because the visual words (cluster centers) obtained depends on the initial cluster centers chosen in the k-means algorithm, which are sampled randomly from the data-points. So, the results may vary a bit (increase/decrease in accuracy) if you computer dictionary from scratch.

3 Stitching two images

The program takes in 2 input images, and lets us select 4 points in each image. The 4 points must be selected in order. These points are marked on the GUI and are displayed by a + sign. These points can be used to compute the homography matrix as done in the code.

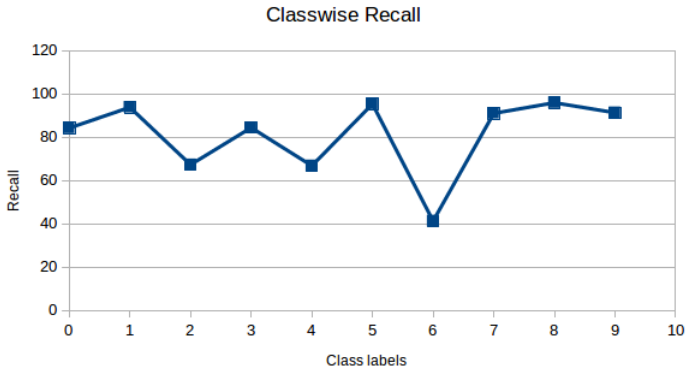

Figure 5: Classwise recall.

Figure 6: Image 1 for stitching.

3.1 Computing homography

Homography is a transformation that maps a projective plane (a quadrilateral) to another projective plane (another quadrilateral, in this case a rectangle). If we represent the points and homography conversion in a matrix format, we will get a matrix A as shown below.

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1x_1 & -x'_1y_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1x_1 & -y'_1y_1 & -y'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x'_4x_4 & -x'_4y_4 & -x'_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -y'_4x_4 & -y'_4y_4 & -y'_4 \end{bmatrix}$$

Here

$$(x_i, y_i) and (x'_i, y'_i)$$

are the corresponding points in image1 and image2 respectively.
Now, the problem reduces to a least squares problem of finding the homography matrix H for which the norm of (AH) is minimum. This is done by finding the eigenvector of (A'A) with smallest eigenvalue.

Figure 7: Image 2 for stitching.

3.2 Warping the image

After finding the homography matrix, we warp image 2 using the projective transform. imwarp function is used to do this. This makes image 2 to lie on the same plane of projection as that of image 1.

Figure 8: Image 2 warped.

3.3 Stitching the two images

Now, to stitch those, we calculate the translations of the warped image and the original image. We get the coordinates in the new image where image 1 and the warped image 2 should lie. We do this by superimposing the 4 marked coordinates, and the other coordinates fall in place. Then, we superimpose image1 and warped image 2. For the overlapping part, we take the average intensity value of both the images.

3.4 Result

We get visually pleasing stitched image given that the 4 points are selected carefully.
The input images [2], the warped result and the stitched image are shown in the figures 6,7,8,9.

Figure 9: Stitched image.

4 References

[1] Han Xiao and Kashif Rasul and Roland Vollgraf `Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms`

[2] Image sourcce : Feature Based Panoramic Image Stitching `http://in.mathworks.com/help/vision/examples/feature-based-panoramic-image-stitching.html`