Report: Image Categorization

Rishabh Singh: 19-953-793

December 13, 2019

1 Local Feature Extraction

1.1 Feature Detection

In this exercise we will be developing a pipeline for image classification into two categories. We will be using Bag of words as the descriptors and we will evaluate the performance of these with two classification approaches, namely nearest neighbor classifier and Bayesian classifier. The first step of the pipeline is thus to extract the Bag of words features from each image. In order to do so, we select points on a uniform grid across the given image. Leaving a border of 8 pixels from all the four sides of the image, a uniform mesh-grid was created with the resolution in the X and Y directions specified by $nPointsX$ and $nPointsY$. The points obtained were considered as local feature points.

1.2 Feature Description

We use Histogram of Oriented Gradients as a feature descriptor. Given an image and the local feature points from the previous step, we extract out a patch of image. Each patch is further divided into individual cells. Thus a patch will be of the dimension: width of cell times number of cells in each patch and similarly for the height dimension. Each cell will contain cell width times cell height number of pixels. We calculate the gradient in both X and Y directions for one cell. Gradient orientations are calculated in each case. We further create a histogram of these orientation values. The range from $0 - 2\pi$ or $(-\pi) - \pi$ can be split into a number of bins. Every orientation value is assigned to a bin based on a voting mechanism. The one we used here is based on the nearest neighbors in terms of the angular values and utilised the *histounts* function provided by Matlab. The results shown in this report are based on this. Moreover, we also computed a voting mechanism based on the magnitude of the gradient, as mentioned in the original paper by *Dalal and Triggs*. The formulation has been coded in the function $computeHOG()$. Here we first bring the angular values to first and second quadrant. We divide the range $0 - \pi$ in a number of bins as specified in the formulation. We calculate the distance of any incoming angle to the centre of the bins. We select the first two nearest neighbors bins. The votes assigned to these two bins is a weighted magnitude of the gradient. The weights are defined by the difference in distance between the angle and the centre of the bins. The final histogram obtained is normalised to account for illumination etc. variations, as suggested in the original paper. We compare the classification performance of the two voting schemes in Fig. 2 and 3. Since we considered a grid which is divided into $4X4$ cells and every cell computes an $8d$ histogram, the final descriptor at one feature point is 128 dimensional.

2 Codebook Construction

In order to capture the appearance variability within an object category, we first need to construct a visual vocabulary (or appearance codebook). We computed the HOG descriptors as explained above for each image in the training set. We then run K-means clustering algorithm for a given number of codebooks. We thus obtain the visual words which serve as some sort of basis vectors and we will later try to learn how much variability is present in the descriptors for an image based on these visual words. The visualisation can be seen in Fig. 1.

3 Bag of Words

In order to check how well these visual words are captured in a particular image, we need to represent the image as a histogram of visual words. To compute the histogram to be returned, we assign the descriptors to the cluster centers and count how many descriptors are assigned to each cluster (i.e. to each 'visual word'). We utilise the knnsearch method to create the histogram. Using this function we process all positive training examples from the directory $carstraining-pos$ and all negative training examples from the directory $cars-trainingneg$ We collect the resulting histograms in the rows of the matrices $vBoWPos$ and $vBoWNeg$.

4 Nearest Neighbor Classification

We first build a bag-of-words image classifier using the nearest neighbor principle. For classifying a new test image, we compute its bag-of-words histogram, and assign it to the category of its nearest-neighbor training histogram using *knnsearch*. The label returned is 1 if a car is present in the image and 0 if there is no car in the image.

5 Bayesian Classification

We first compute the mean and standard deviation of all the dimensions in the descriptor for separate cases of positive and negative samples. We assign a prior value to each class. In our case, we assumed the prior probability of positive and negative classes to be 0.5 each. We then calculate the log likelihood values for each entry in the test histogram. It might be the case that for some dimensions, the mean and/or the standard deviation can be zero. Fitting a normal distribution over these parameters results in NaN values. To circumvent that, we explicitly assign value of 1 to the probabilities returned from such cases. Since we are working in Log space, this accounts for a value of zero. Hence, a check is kept in the code to see if the Normal function returned by Matlab is not NaN and sum the log-likelihood values accordingly. We compute the likelihood values by taking the exponential of log-likelihood values obtained for both classes. We compute the posterior by multiplying the likelihood and the prior. The evidence value is not considered in the decision rule, given it is the same for both the classes. We finally assign the label to the class for which the posterior probability is higher. We chose to work in the log space while computing the likelihood values in order to avoid numerical instabilities arising from multiplying large number of probability values otherwise.

6 Results

We evaluate the above pipeline over 5 iterations for each classifier and report the average accuracy recorded.

- For K-nearest neighbors, the average accuracy over 5 iterations for codebook size of 200 is **92.12%**.
- For Bayesian Classification, the average accuracy over 5 iterations for codebook size of 200 is **89.29%**.

The initialization of the Kmeans clustering algorithm is random. Hence when evaluating the performance of the classifier with varying k, it is important to repeat the test multiple times for each k with a different random number seed each time. We followed the same approach and analysed the performance of both the classifiers for varying K as shown in Fig. 2.

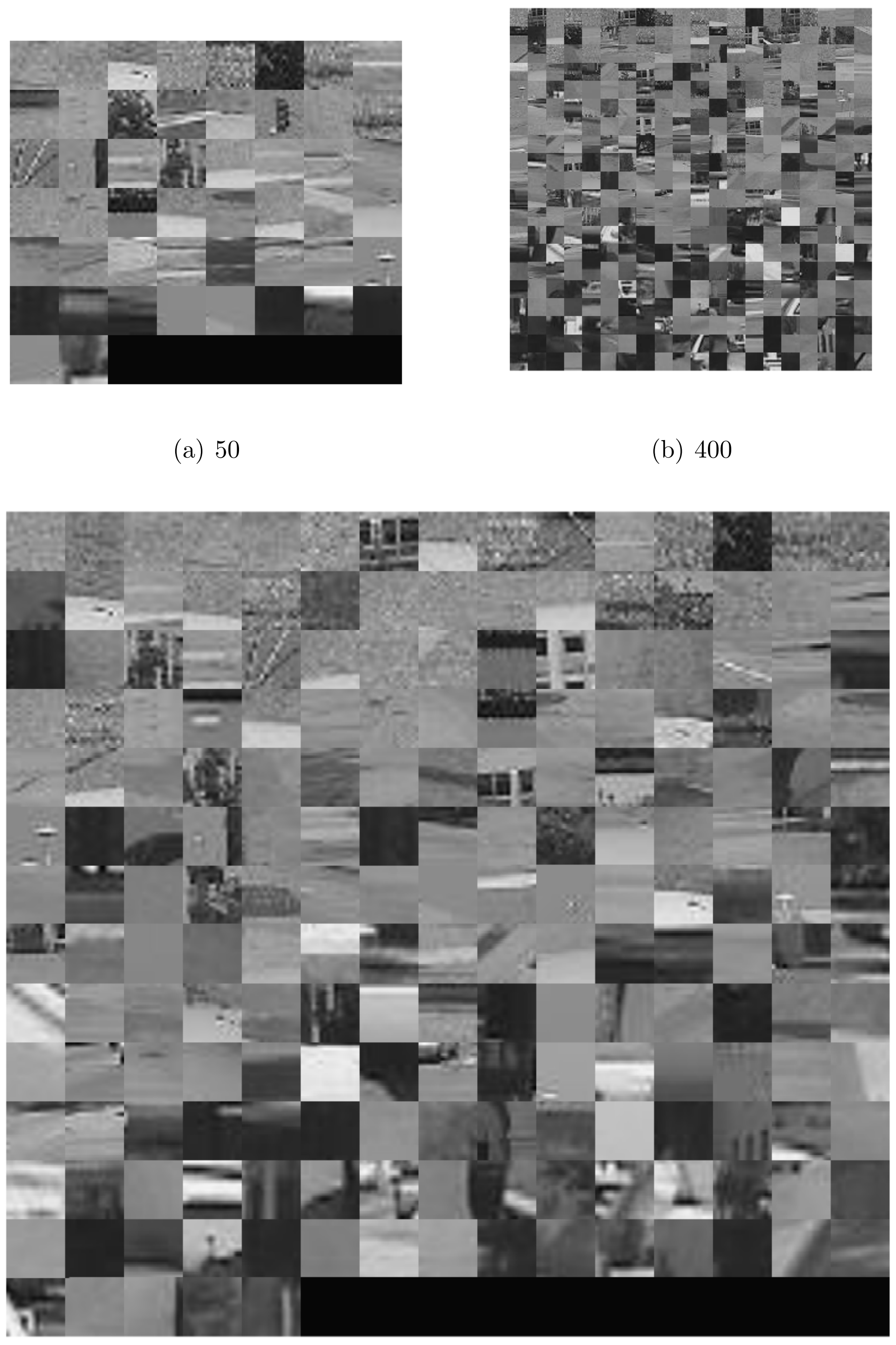
7 Discussion

- Nearest neighbor in general has better performance than Bayesian Classification for this dataset.
- One reason for the inferior performance of Bayesian Classifier can be the assumption of conditional independence of the histogram dimension.
- The assumption that histogram counts follow a Gaussian distribution is a strong assumption. This might not be the case for every dataset.
- Codebook generation is highly non-deterministic. Even though the performance is averaged over multiple iterations, the feature descriptors should not vary much with newer iterations. This also might be one source why Bayesian, which tries to model independent dimension of the descriptor, is susceptible to variability in codebook size.

- The performance of K nearest neighbor stays approximately high (mostly above 90%) with varying codebook size. One reason can be that kNN tries to focus on the relative distance of descriptors in the feature space, instead of the changes in individual dimension of the descriptors. This relative placement of descriptors is highly unlikely to change drastically with small changes in feature dimensions, which are governed by codebook size.
- For higher values of codebook size, the performance of both the classifiers slightly deteriorates. This may be due to presence of visual words which actually aren't contributing to the representation of an image and unnecessarily increasing the dimension of the feature vector.
- The performance of Bayesian classifier, although lesser than nearest neighbor, is still above 85% in most of the cases with varying codebook size.

8 Bonus Task

We evaluated the pipeline on the dataset of two flowers. The training set consisted of 50 images each of Dandelion and Tulips. Similarly, the test set has 50 images of each class as shown in Fig. 4. The codebook generated for a size=200 is shown in Fig.5. The performance of KNN on this dataset is 66.6% while Bayesian Classifier has 69.8%. The poor performance of the pipeline can be attributed to high intra-class variation in the given dataset. Samples of the same class were quite different form each other. Learning visual words which can represent every sample i the class can be over-generalisation. Also, color and texture can be important cues for flower classification which are ignored in this pipeline (all images are converted to grayscale). This might be another reason of the inferior performance.



(a) 50 (b) 400

(c) 200

Figure 1: Visualisation of the code-book generated for different code-book sizes. Some visual words capture parts of cars like edges and tyres.

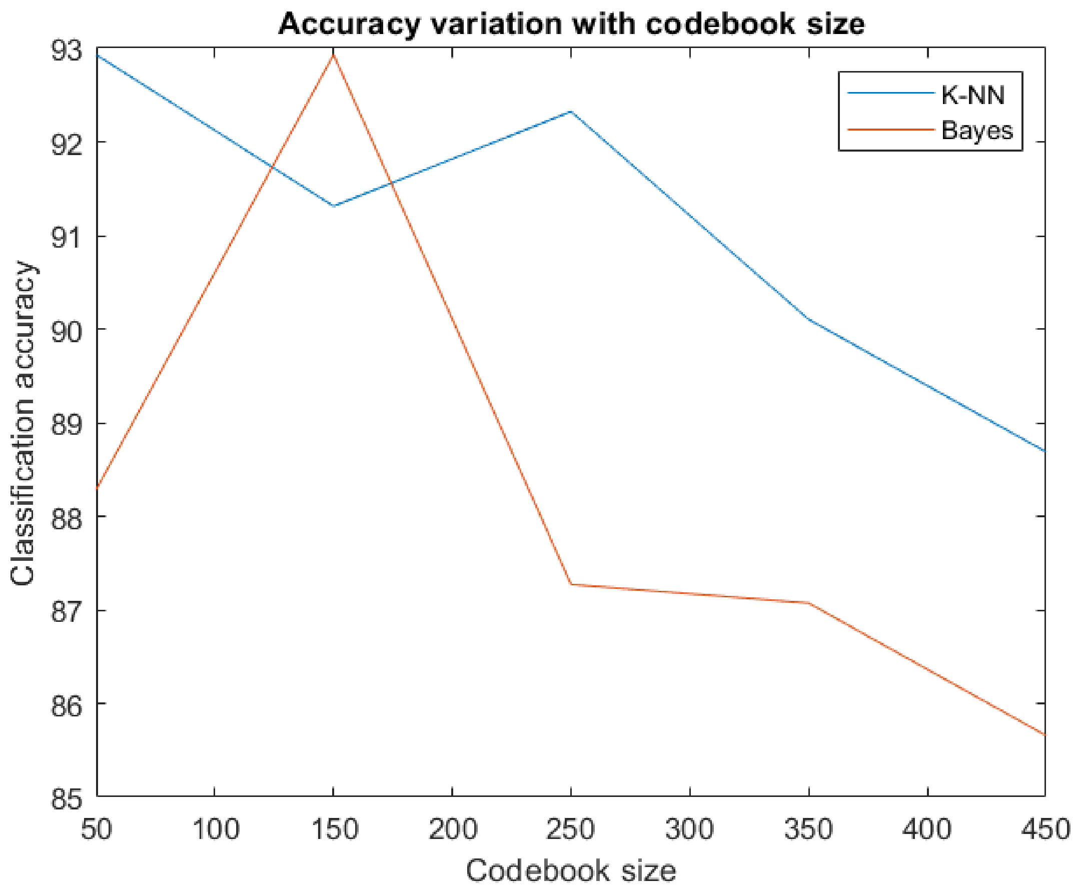


Figure 2: Performance of classifiers with different codebook sizes. The voting mechanism used here is based on increasing the count of a bin based on angular distance from its centre.

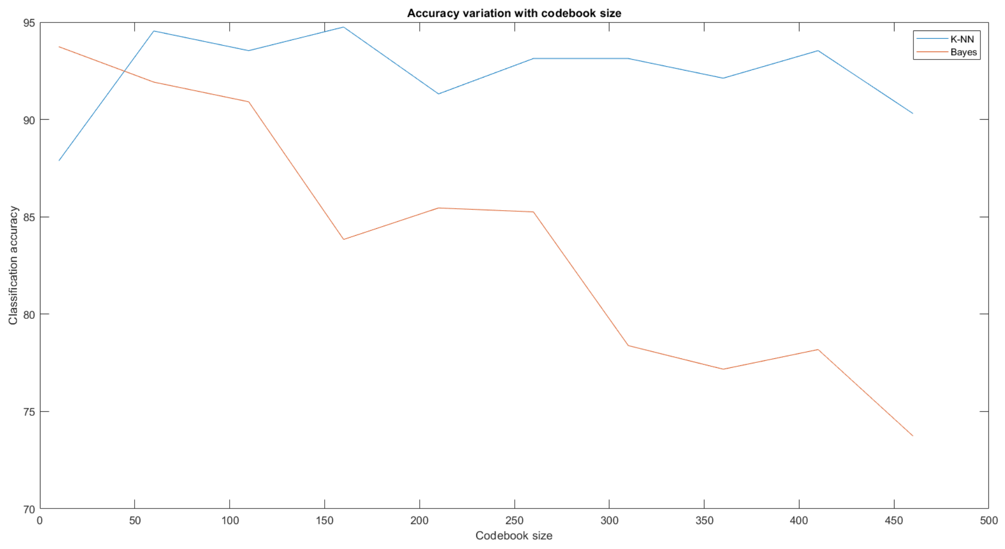


Figure 3: (Additional)Performance of classifiers with different codebook sizes. The voting mechanism used here is based on assigning a weighted magnitude to each bin, as stated in the paper on HOG by Dalal and Triggs .

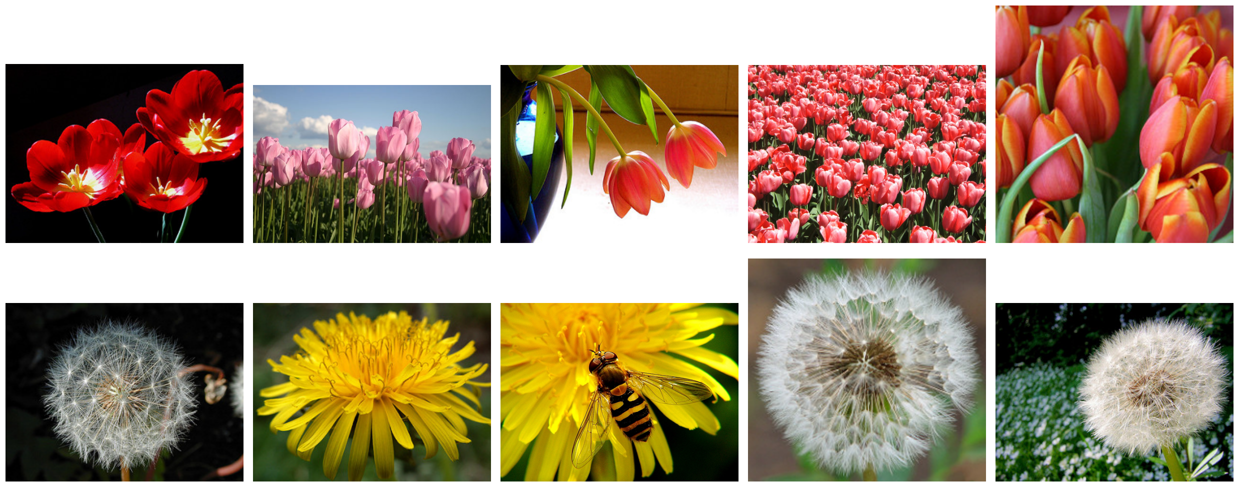


Figure 4: (Top) Class 1 consists of samples of Tulip (Bottom) Class 2 has samples of Dandelion. The dataset has huge intra-class variations.

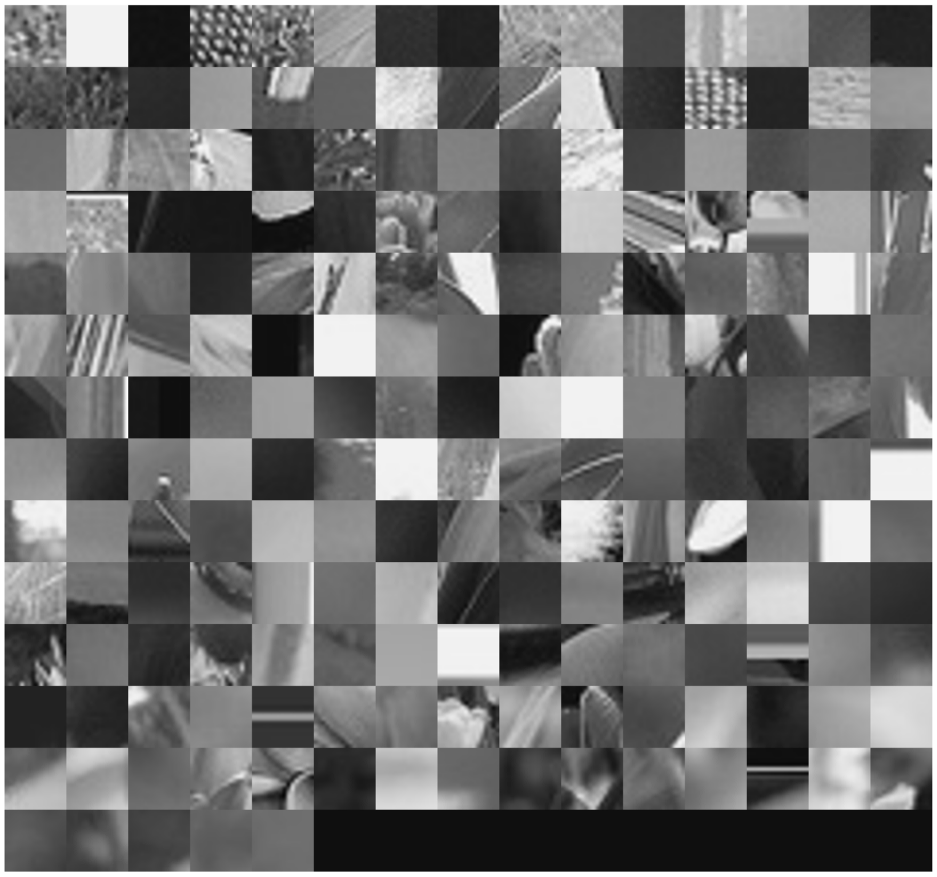


Figure 5: Codebook of size 200 for training samples consisting of Tulips and Dandelions. Note the parabolic contour of tulip and the mesh-like interior of Dandelions can be seen in some of the visual words.