PASCAL Project Report

Mohammad Rami Koujan, Duncan Sommer

May 30, 2016

Abstract

In this project, we present a bag-of-words approach to the PASCAL Visual Object Classes Challenge. Using the VLFeat and PRTools libraries, we tested a wide variety of algorithms for various stages of our system. Thorough experimentation with many possible setups allowed us to achieve accurate final results, and was an instructive introduction to the characteristics of the bag-of-words method.

1 Introduction

The bag-of words technique was first discovered in the field of natural language processing, and showed effective enough to gain widespread acceptance for tasks such as document classification. Recently, it has been introduced into the context of computer vision research. By replacing text documents with images and words with their features, the abstract problem translates well enough into the new arena. However, complications quickly arise. Unlike words, image features can be hundreds of elements long and far more detailed. This problem must be resolved by categorizing raw feature vectors extracted from each image into a limited, predefined number of words. Then, a histogram of these keywords is created and used to categorize the picture as one of the ten PASCAL VOC classes. This report will explain our strategy and methodology for tackling this computer vision challenge, offer an overview of the various algorithms involved, and demonstrate the results we have achieved.

2 Strategy

Our strategy splits the bag-of-words idea into four main steps: feature descriptors, clustering, pooling, and classification. Although the real goal of the PASCAL Challenge is image classification, the bag-of-words approach requires several steps before a classifier is invoked. Different algorithms were tested for each of these four steps except clustering, for which we used only k-means. First, several different techniques for dense feature extraction were tested, focusing of the well-known SIFT algorithm. Next, the enormous number of feature vectors for all pixels in all images are clustered using k-means into a designated number of words; this number is one of the key parameters of the bag-of-words system. Each image then has its features categorized into the word bins, and a histogram of words is generated for each image. The histograms resulting from the pooling operation are the input to the two classification algorithms experimented upon. In the implementation section, we provide more detail on each of these steps and selected algorithms.

3 Implementation

3.1 Descriptors

Feature extraction is the first step in the bag-of-words pipeline, and for that we must choose a descriptor. There are a large number of well-known shape, texture, color, and contour descriptors in the literature. Perhaps the most famous of recent years is the Scale-Invariant Feature Transform, or SIFT. This method was originally intended for feature matching, but the descriptor proved robust enough to gain widespread use in a variety of applications. This section will cover the basics of the SIFT algorithm, which we used

for all of our experiments. For some trials, we used it alongside other descriptors which will also be described below.

3.1.1 SIFT

The strengths of SIFT are the multiple steps it takes to minimize situational effects on the features. SIFT is rotation, scale, and translation invariant, as well as partially invariant to illumination changes and geometric stretching. These properties result from its design as a descriptor. The feature vector is created by generating histograms of the gradient for subwindows nearby the chosen pixel. Windows are convolved with a Gaussian kernel to increase the influence of closer pixels. Blurring the images provides some leeway so that noise and small affine transformations will not affect the features too much. The gradient histograms are shifted to begin at the dominant orientation, which ensures rotation invariance. Sparse SIFT first uses a detector to select a small number of interest points, while dense SIFT instead generates a feature vector for each pixel of the image. In order to make sure that we describe all sections of the image, even homogeneous ones, we applied dense SIFT in our experiments.

Figure 1: Dense SIFT applied to several scales

3.1.2 Local Binary Patterns

We next decided to combine DSIFT with other descriptors to include extra information. This is as simple as computing both descriptors and concatenating their results; the clustering algorithm will treat them both as a single feature vector. Local Binary Patterns is an elegant and robust texture descriptor, so we first combined DSIFT with LBP. This algorithm also examines a 16x16 window around the central pixel, but performs a comparison operation on each neighbor pixel. Pixels are compared with their 8-neighbors, and assigned a binary string representing whether each is larger or smaller. The binary strings are treated as numerical values, so a histogram of LBP codes is created across the 16x16 window. The histograms are normalized and concatenated to form the final feature vector. By using LBP and DSIFT at the same time, our features describe the gradient orientations as well as the texture at each pixel in the image.

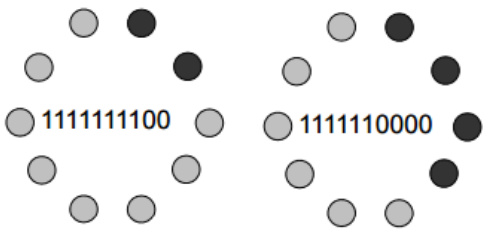

Figure 2: Example LBP neighborhoods

3.1.3 Histogram of Oriented Gradients

In the same way as LBP, we also tried the Histogram of Oriented Gradients technique with SIFT. HOG is computed in a very similar way as SIFT, so combining the two reinforces the gradient information contained in the feature vector. HOG, as its name suggests, computes histograms of the image gradients at pixels in a window around the chosen location. Like SIFT, HOG splits the window into subwindows and creates a histogram for each section. These histograms are again concatenated to form the final feature vector. The differences to SIFT lie in how the histograms are normalized, because HOG also uses contrast normalization. Historically, HOG has been observed to synergize well with the SVM classifiers which we used for most of our experiments. For these reasons, we experimented to see if adding the HOG descriptor to the DSIFT descriptor improved performance.

3.1.4 DSIFT in RGB

Finally, we applied standard DSIFT across all RGB planes. Normally, images are simply converted to grayscale before SIFT is applied, but for this approach the colors remain separate. This allows us to include valuable color information in the feature vector, but comes at the high cost of tripling its size. Nevertheless, creating three DSIFT descriptors for red, green, and blue separately is a well-known technique and worth the time to test.

3.2 Clustering

After generating features from all training images, K-means is applied to extract visual words from the raw features. This consists of finding clusters of feature vectors which are similar to each other to represent some characteristic that various images have in common. Later, the presence or absence of a word will serve as features for classification. In order for this to work, K-means must compute a set of centroids for the features which minimizes inter-class distances.

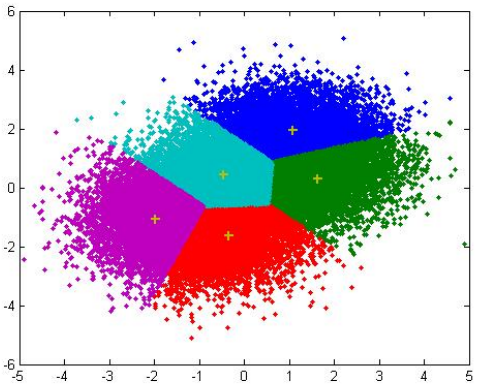

Figure 3: Example clusters resulting from K-means algorithm

K-means works as follows. It takes the number of centroids, or words, as input and initializes them randomly. Then, each sample feature vector is assigned to the closest centroid, using Euclidean distances. After all samples have been given a group the mean of each group is computed, and all centroids are redefined to the mean of their group. We simply iterate these two steps, assignment and update, until the centroids stop moving. This method is generally fast and robust, but falls victim to local minima. In fact, the algorithm is so fast that some implementations perform multiple runs and accept the best result.

3.3 Pooling

Pooling is a relatively simple step which converts images into feature vectors for the classifier. For each image, all of its features are assigned to one of the clusters created with K-means. There are two techniques

used to choose the nearest cluster: simple L2 pooling, or K-d tree pooling with spatial histograms. L2, or Euclidean distance only measures the length of the vector between the centroid of the cluster and the sample feature. K-d trees are more complex. First, the clusters are split using a tree construction algorithm to create an n-dimensional tree which can be searched quickly. Then, when a new feature is encountered, we traverse the tree to find its nearest neighbor.

Whichever method is used, each feature will be assigned to one of the clusters. Now that we know which words are contained in the image, a histogram is generated to describe the image. The histogram will be the only input to the classifier, so the number of words is key. When using k-d trees we also implemented spatial histograms. This means that the image is first divided into subsections, such as top/middle/bottom or left/center/right. Then, a histogram of words is created for each subsection and all are concatenated together to create the final feature vector to represent the image.

3.4 Classification

The classifier is the final step, and one of the most influential. Once the chosen algorithms have been used to prepare the most descriptive feature vectors possible, we must invoke the classifier. This process uses a variety of methods in an attempt to learn which features are associated with which classes, and then apply that knowledge to predict the class of new, unforeseen samples. The number of classes defined is an important decision in any classification situation. With overly broad categories, such as "night" or "day," the problem is trivial and is of little use. Using excessively specific categories will dramatically reduce performance due to sparsity of the training set and greater demands for classification precision. Luckily, the PASCAL VOC Challenge has predetermined ten classes for analysis, so this implementation detail is not in our hands.

There are a huge variety of classification algorithms, but they can be divided into several categories. Most classifiers use the feature vector to compute a probabilistic "score" which is thresholded to determine the output class for each sample. Linear classifiers model the score as a simple linear combination, where each feature is given a weight. The score is computed as the dot product of the weight vector with the feature vector, so the purpose of training is only to determine the values of each weight.

$$y = f(\vec{w} \cdot \vec{x}) = f\left(\sum_j w_j x_j\right)$$

Figure 4: Dot product as the basis of linear classification

Linear formulation does not restrict our classifier to strictly linear problems. Using the kernel method, the dot product can be used to represent nonlinear functions in higher-dimensional space. This greatly improves performance on large classes of problems which cannot be modeled linearly. Classical linear classifiers are also limited to binary decision-making, which again seems to limit the scope of these algorithms. To combat this, multiple binary classifiers can be trained simultaneously. For a given input sample, the binary classifier with the highest confidence value is used as the output class. In fact, this approach also allows the classification to return the normalized probability that the test case is an instance of any of the given classes.

The feature vectors are intricately linked to the classifier's performance. Meta-information about the vector such as its length, format, and normalization affect results just as much as the actual features (and sometimes more!). Choosing feature extraction techniques which synergize well with the type of classifier is critical. In the following sections, we will provide a brief summary of the classification algorithms applied for this challenge.

3.4.1 Nearest Neighbor

Nearest Neighbor classification is one of the simplest machine learning techniques. In the standard implementation, the training set is not processed at all and just saved as a reference during testing. Test samples are simply compared to each training sample, and are given the same label as their nearest

neighbor. The distance between neighbors is calculated as Euclidean distance in feature space. This method is often extended to k - Nearest Neighbors, in which the k closest samples to the test case are examined. The majority class vote among those samples, sometimes weighted by their distances, is used to choose the output label. However, saving the training set for use during test can use large amounts of memory; to avoid this, the set can be reduced in size. In this case, a few representative points known as prototypes are chosen and used as a smaller training set for comparisons. Our application uses 1-NN classification without reducing the training set.

3.4.2 Support Vector Machine

The idea behind a SVM is to express the training points in high-dimensional space, and find a hyperplane which separates the classes. Normally, we would use the same number of dimensions as the number of features, but the kernel trick works well for SVM. To split non-linearly separable classes, a dot product is used to map the features to even higher-dimensional space. Because the SVM is a linear classifier which uses the dot product with its set of weights, the training set is only used once to compute the weights. Calculating the weights is equivalent to minimizing the following equation. Once the weights are found by processing the training set, they can be quickly used to classify each sample in the test data.

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i(w \cdot x_i + b)\right)\right] + \lambda\|w\|^2$$

Figure 5: SVM minimization requirement

4 Experiments and Results

This section presents the results of classifying the set of provided images from Pascal VOC2006 dataset. A number of methods are followed in order to classify the given images, which are divided into 3 different sets with 10 distinct classes: train (256 images), validation (269 images), and test (523 images). The following diagram shows the main pipeline adopted in this project to implement the bag of visual words method.

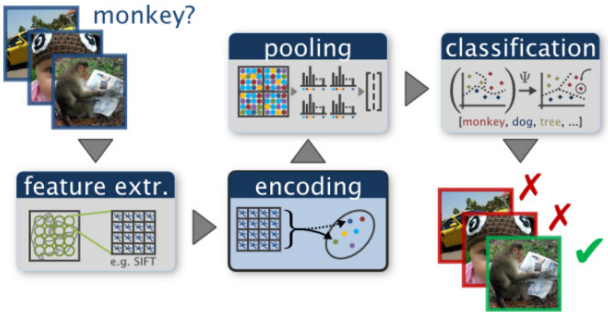

Figure 6: An example of a typical bag of words classification pipeline

The previous figure demonstrates four main steps (feature extraction, encoding, pooling, classification), where a number of techniques are tested in each of these steps. At the beginning, Dense SIFT descriptor is used to extract the features of interest from the training dataset. In fact, *vl_phow* function provided by VLFeat library is used to carry out this task of features extraction. The following parameters are passed to the stated function:

- *Sizes*: 4
  Scales at which the dense SIFT features are extracted. Each value is used as bin size for the $VL\_DSIFT()$ function.

- *Step*: 8
  Step (in pixels) of the grid at which the dense SIFT features are extracted.
- *Color*: gray
- *ContrastThreshold*: 0.005
- *WindowSize*: 1.5

Additionally, 95 per cent of the extracted features is chosen randomly from each image and stored eventually in one MxN matrix, where M represents the size of the feature descriptor, 128 for SIFT, and N is the total number of features extracted from all the images in the training set. Then, Kmeans algorithm is used to do the encoding step using the following set of parameters:

- Number of Centroids (visual words): 500
- Distance: L2
- Initialization: random
- Algorithm: Elkan
- MaxNumIterations: 50

The function that is used for Kmeans algorithm is Vl_kmeans from VLFeat library. Next, pooling is done based on L2 distance of each extracted feature with the visual words, which results in a separate histogram for each image in the training set. This histogram consists of the number of occurrences of the visual words in the corresponding image. Therefore, the size of each histogram is the same as the number of the visual words (500). Finally, in the test step, the validation set is used as a test set where initially the features of each image are extracted using DSIFT with the same previously mentioned parameters. Afterwards, the histogram is built using the visual words (L2 distance) and the result is passed to the classifier, which is of type NN (Nearest Neighbor). This classifier simply computes the distance between the histogram vector of the test image with all the histograms vectors of the training set and eventually returns the ratio of the distance to nearest negative observation to the distance of the nearest positive observation as a confidence value which is used later to draw the ROC (Receiver Operating Curve) and calculate the AUC (Area Under Curve). The following table shows the results, in terms of AUC, of testing the described functions on the validation set. The names of the different algorithms used in each step is mentioned under each of the following tables.

Class name	Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motor bike	person	sheep
$DISFT\_NN$	0.754	0.689	0.784	0.704	0.760	0.743	0.699	0.708	0.542	0.637

Table 1: Results of using $DSIFT\&Kmeans\&L2-pooling\&NN-classifier$

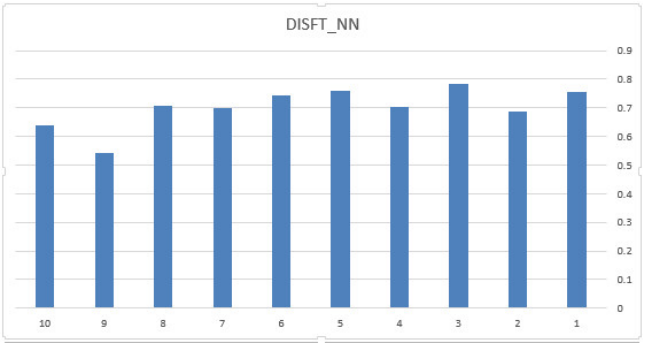

Figure 7: Results of using $DSIFT\&Kmeans\&L2-pooling\&NN-classifier$

In the second trial, non-linear SVM classifier with second order polynomial kernel is used instead of NN classifier. This classifier is imported from the PRTOOLS library. Below are the results which show some improvements:

Class name	Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motor bike	person	sheep
$DISFT\_SVM$	0.828	0.877	0.884	0.669	0.877	0.632	0.667	0.709	0.598	0.802

Table 2: Results of using $DSIFT\&Kmeans\&L2-pooling\&SVM-classifier$

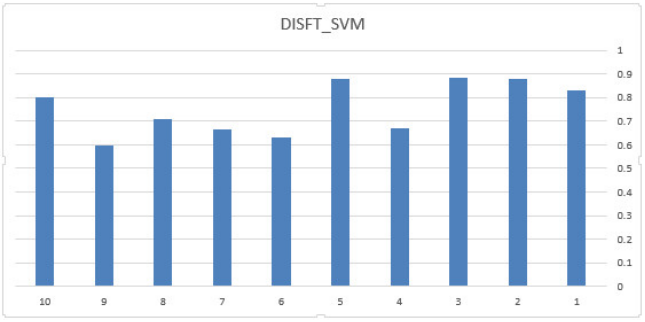

Figure 8: Results of using $DSIFT\&Kmeans\&L2-pooling\&SVM-classifier$

In the third trial, the step of building the histogram is modified in the second version. The new method is to build a spatial histogram which takes into account not only the number of occurrences of the visual words but also their location inside the image. Three levels are used to divide the x and y coordinates of the image (2,4,8), which results in 4,16,64 windows for the first, second, and third level respectively. Moreover, Kdtree is used instead of direct L2 pooling in this step. Next are the obtained results:

Class name	Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motor bike	person	sheep
$DISFT\_SVM\_SP-Hist$	0.838	0.895	0.902	0.659	0.901	0.708	0.728	0.737	0.642	0.784

Table 3: Results of using $DSIFT\&Kmeans\&Kdtree-pooling-with-spatial-histograms\&SVM-classifier$

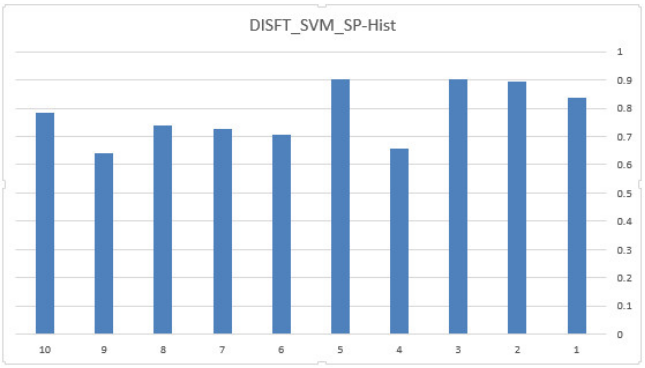

Figure 9: Results of using $DSIFT\&Kmeans\&Kdtree-pooling-with-spatial-histograms\&SVM-classifier$

The previously shown results clearly demonstrate the effect of taking into account the spatial information of the visual words as well. This has played a crucial role in improving the result of some classes, e.g. Cow, Horse, Motorbike, Person.

In the fourth trial, Local Binary Pattern descriptor, which is used to extract texture based information, is incorporated with DSIFT features. In other words, LBP descriptor is extracted at regions (8 pixels size) centered around DSIFT points and the results is augmented with the 128-length vector of SIFT for each feature. Table 4 shows the effect of this modification.

Class name	Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motor bike	person	sheep
$DSIFT\_LBP$	0.783	0.873	0.892	0.7	0.870	0.693	0.673	0.711	0.615	0.824

Table 4: Results of using $DSIFT - LBP\&Kmeans\&L2 - pooling\&SVM - classifier$

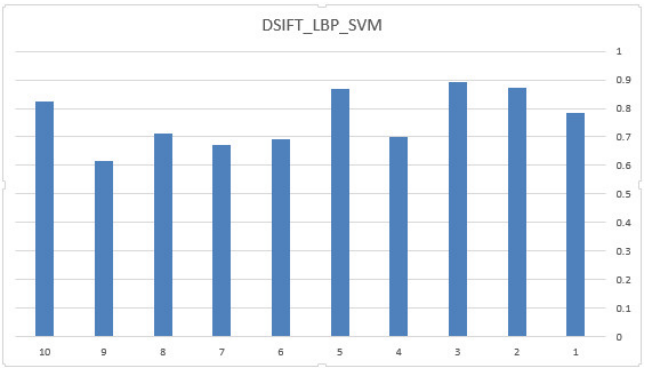

Figure 10: Results of using $DSIFT - LBP\&Kmeans\&L2 - pooling\&SVM - classifier$

another test was done for DSIFT-LBP but with spatial histograms:

Class name	Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motor bike	person	sheep
$DSIFT\_LBP\_Sp - hist$	0.843	0.856	0.907	0.704	0.894	0.727	0.735	0.750	0.656	0.787

Table 5: Results of using $DSIFT - LBP\&Kmeans\&Kdtree - pooling - with - spatial - histograms\&SVM - classifier$

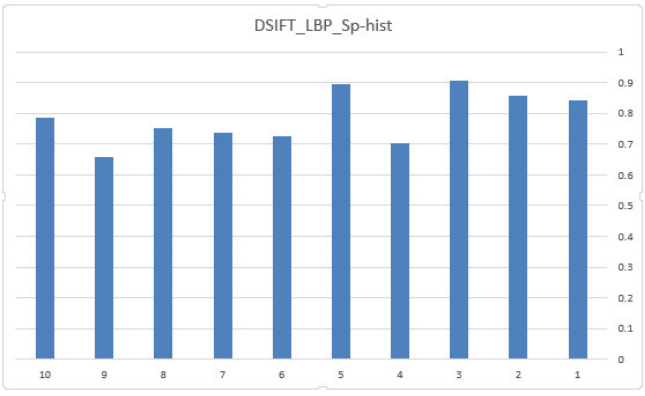

Figure 11: Results of using $DSIFT - LBP\&Kmeans\&Kdtree - pooling - with - spatial - histograms\&SVM - classifier$

In the fifth trial, HOG (Histogram of Oriented gradients) features are used in the same previous manner instead of LBP, extracted also around regions with 8 pixels size and centered around DSIFT points. The following two tables show the results.

Class name	Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motor bike	person	sheep
DSIFT_HOG	0.851	0.849	0.901	0.728	0.866	0.684	0.666	0.663	0.585	0.825

Table 6: Results of using $DSIFT\_HOG\&Kmeans\&L2-pooling\&SVM-classifier$

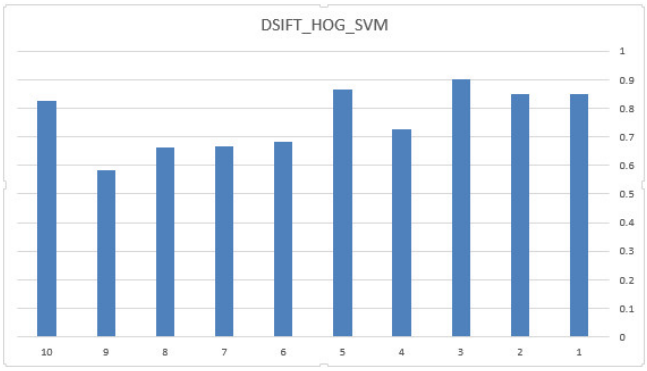

Figure 12: Results of using $DSIFT\_HOG\&Kmeans\&L2-pooling\&SVM-classifier$

Class name	Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motor bike	person	sheep
DSIFT_HOG	0.857	0.879	0.902	0.718	0.881	0.749	0.741	0.752	0.637	0.775

Table 7: Results of using $DSIFT\_HOG\&Kmeans\&Kdtree-pooling-with-spatial-histograms\&SVM-classifier$

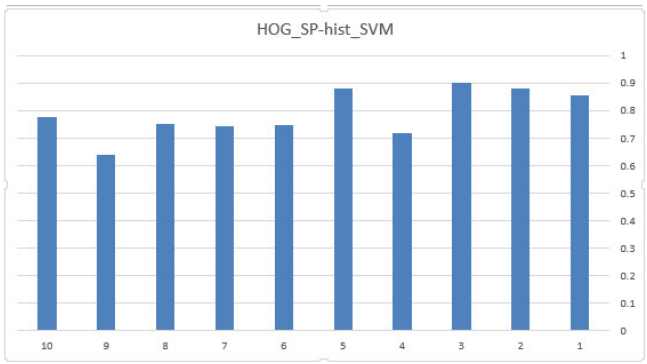

Figure 13: Results of using $DSIFT\_HOG\&Kmeans\&Kdtree-pooling-with-spatial-histograms\&SVM-classifier$

In the sixth trial, DSIFT is used alone but using color information which increases each feature size from 128 to 128*3. This features are accompanied with Kmeans, Kdtree pooling with spatial histograms, and SVM-classifier in the following stages.

Class name	Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motor bike	person	sheep
DSIFT_HOG	0.846	0.848	0.913	0.684	0.856	0.752	0.709	0.736	0.629	0.784

Table 8: Results of using $DSIFT-RGB\&Kmeans\&Kdtree-pooling-with-spatial-histograms\&SVM-classifier$

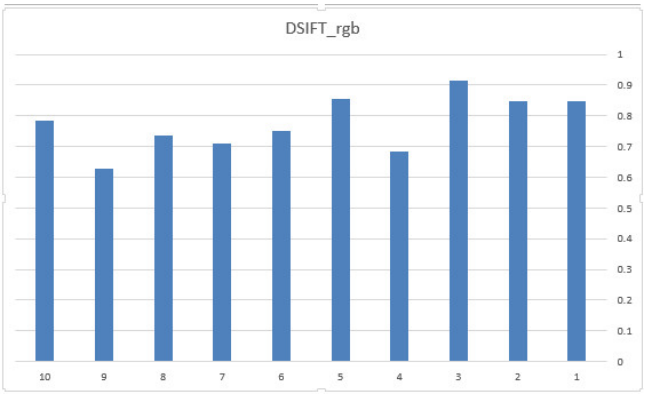

Figure 14: Results of using $DSIFT - RGB\&Kmeans\&Kdtree - pooling - with - spatial - histograms\&SVM - classifier$

In fact, some other experiments were conducted using various algorithms from different libraries, LIBSVM, PRTOOLS, and WEKA, but the presented ones are among the best. Moreover, a wide range of parameters were tested in each stage and eventually have been tuned to give the best performance. The following bar chart shows a comparison between the implemented methods shown in this section.

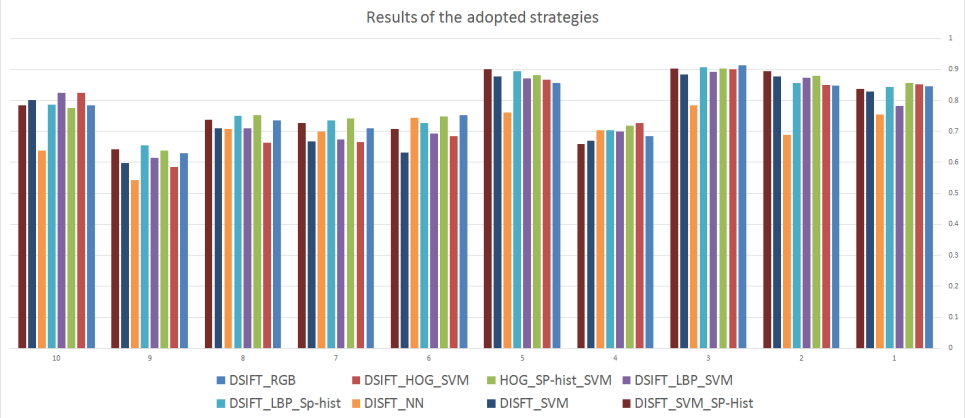

Figure 15: Comparison between the obtained results

The preceding bar chart exhibits different results on average between the 10 tested classes due to the inter-class variability and myriad types of images found in the VOC 2006 dataset. This necessitates the incorporation of different types of features taking into account color, affine changes, textural properties, and so on in order to acquire good results.

The previous tests were indeed applied to the provided validation set. Thus, the following table shows the result of testing the test set using $HOG\_SP-Hist\_SVM$ method, which has shown quite good result for the validation set, and the previously used train set for training.

Class name	Bicycle	Bus	Car	Cat	Cow	Dog	Horse	Motor bike	person	sheep
$HOG\&DSIFT$	0.863	0.867	0.83	0.772	0.872	0.713	0.634	0.718	0.616	0.883

Table 9: Results of testing $HOG\&DSIFT\_SP-hist\_SVM$ method on the test set

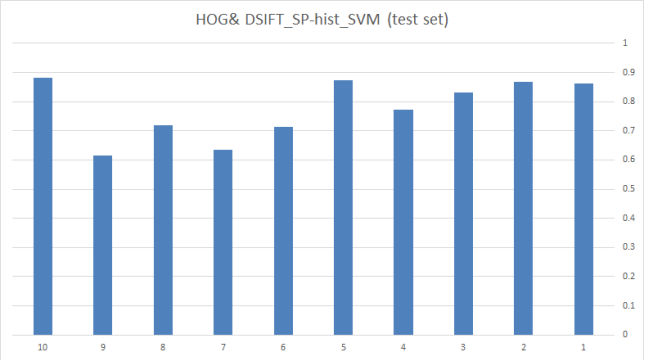

Figure 16: Results of testing $HOG\&DSIFT\_SP - hist\_SVM$ method on the test set

The preceding table reveals that the results of the test set using $HOG\&DSIFT\_SP - hist\_SVM$ method are comparable to the ones obtained by the same method on the validation set even though some classes like the car have less AUC values while others like sheep and Bicycle have improved for the test set.

5 Organization

Due to the significant size of this project, planning in advance was critical. During the first few weeks, we both installed the VLFeat and PRTools library and experimented with their functions. We also discussed the overarching strategy, and came up with an initial list of algorithms and techniques to experiment on for each step of the bag-of-words pipeline.

Next, we divided the ideas and began running tests on two separate computers. For each test, we recorded the results and collaborated to share tips on how to improve efficiency and prevent our computers from falling asleep overnight. The bulk of the time spent on the project was in testing, because we wanted high-quality results to demonstrate the effectiveness of our approach.

Finally, we wrote the report during the last two weeks. Since there were still a few tests remaining to run, we left the results section for last and began by explaining our framework and implementation. Each of us wrote a few sections and proofread the other's parts. During the last few days, we made final corrections and updated the results with our best accuracy scores.

6 Conclusions

In this project, some bag of words models that can applied to image classification are designed, implemented using Matlab, and tested on the PASCAL Visual Object Classes (VOC) 2006 dataset. During each stage of the bag of words model, a number of algorithms are tested to achieve the highest possible performance which is measured by the AUC (Area Under Curve) parameter computed from the ROC (Receiver Operating Curve). For the first stage, features extraction and description, DSIFT and HOG demonstrate competitive results among the other available ones. Both of these features extraction methods are accompanied with K-means, Kd-tree, spatial histograms, and a non-linear SVM classifier in one-versus-all mode in subsequent stages so as to obtain the highest performance. Due to the large class variability presented in VOC 2006 dataset, some of the classes are much better classified than others, which is also the case for the published results of the official Pascal 2006 Challenge. In addition, a comparison is drawn between the obtained results showing that other combination of methods like LBP and DSIFT based on color channels in the image can achieve competitive results.

References

[1] Lowe, David G. (1999). "Object recognition from local scale-invariant features". Proceedings of the International Conference on Computer Vision. pp. 1150–1157.

[2] T. Ojala, M. Pietikäinen, and D. Harwood (1996), "A Comparative Study of Texture Measures with Classification Based on Feature Distributions", Pattern Recognition, vol. 29, pp. 51-59.

[3] Lloyd, S. P. (1957). "Least square quantization in PCM". Bell Telephone Laboratories Paper.

[4] http://www.vlfeat.org/

[5] http://prtools.org/

[6] https://www.csie.ntu.edu.tw/ cjlin/libsvm/

[7] http://host.robots.ox.ac.uk/pascal/VOC/voc2006/index.html