Pascal Project Report

Gourab Ghosh Roy and Raabid Hussain

May 29, 2016

1 Abstract

The primary objective of this project was to successfully implement a bag of words based image classification system and test the system on a real dataset. 10 sample visual object classes, divided into a training, a validation and a testing set, from the Pascal challenge were provided to us. The goal was to develop a robust and independent classifier to differentiate between the objects. Different feature extraction techniques were used for this. A dictionary of features was built using the bag of words model. This dictionary was used to train the classifier using the training set. First the different features and classification techniques were tried on the validation set to choose the best combination which was later tested on the testing set to evaluate the performance of the classification system.

2 Introduction

One of the most popular computer vision problems is that of image classification which refers to the process that uses image information and decides whether the image contains a particular type of object or not. This problem has applications in several fields like object tracking, image retrieval and surveillance to cite a few. During the last decade, many new approaches to the image classification problem have surfaced due to the increase in interest on content-based image retrieval systems [1]. The classification process can be supervised or unsupervised by a human or a computer.

Unsupervised classification refers to the use of software analysis of an image alone to classify images without any human assistance. The only human input that is associated with unsupervised learning is to specify the number of classes or to provide an estimate of what type of features/objects are contained in the images. Different techniques are used by computers to determine which pixels are related and groups them together to deduce image classes.

Supervised classification refers to the use of knowledge extracted from training samples with the aim of classifying images. A user selects sample pixels in an image which are representative of a specific class and then directs the classification software to use these as training reference to classify pixels in an image or to classify the image as a whole. The user may also set a threshold that defines the degree of similarity to be achieved in order to group pixels together. The user also delegates the maximum number of classes.

In our project, supervised learning techniques are used as they have been known to be more accurate for recognizing classes of objects. A typical supervised classification consists of three stages: creating a dictionary, training and testing. The main disadvantage of supervised classification is that the results highly depend on the competence of the user to accurately designate object labels to the sample images. Fortunately, this factor had been abstracted by the availability of the PASCAL challenge 2006 [2] dataset. The dataset contains 10 object classes:

- bicycle
- bus
- car
- motorbike
- cat
- cow
- dog
- horse
- sheep
- person

The main task of this project was to classify images from a dataset into one of these classes. Different techniques were selected and combined together to solve the classification problem in this project.

3 Design and Implementation

For implementation of the image classification framework, a code skeleton used in the PASCAL challenge 2006 [2] was used. The framework contains three sets of images. The train dataset is used for extracting features to create a dictionary and then to train the classifier system too. The validation dataset is used to test the system created using train dataset during development and select the combination of techniques that work the best. The test dataset was used to test the accuracy and robustness of the system. Since, the system had been optimized according to the validation dataset, it was important to finally test the performance on an unknown dataset in order to verify the credibility of the system.

The dataset contained over 1000 images containing different images of 10 objects. Some of the images contains more than one object type. Labels were assigned to each image and a text file was provided for each object class. The text file contained labels of each image along with annotations for the presence of the object class. -1 represented that the class is not present in the image, 1 stated that the class is present whereas 0 stated that the class is present but is difficult to identify. These labels were used to train and test the system.

The skeleton provided called all the images in the respective datasets, extracted features for the training and testing datasets, trained the classifier, tested it on test data to draw the ROC curves for each class and calculated the area under ROC curves (AUC). The skeleton used the mean RGB values of all pixels in the image as dummy features. We were to alter the feature extraction, training and testing functions in the skeleton to improve the performance of the system. All the development was done keeping the same structure in place where dictionary creation using bag of words model was added to it, as were added different other feature extraction and classification techniques.

To summarize, this is the basic workflow of the entire image classification framework.

- Read train images, extract image features from 6 positive examples of each class. Run Bag of words model (K-Means) on this data to get

dictionary.

- For each class, build feature vectors for all training images using the same feature extraction technique and the created dictionary. Train chosen classifier.
- Build feature vectors for all test images and test the classifier on this test data.
- Plot ROC curve and compute AUC. Start training for next class until all classes done.

The different important algorithms used in the project framework are explained in detail in the following sections. Most of the implementations are based on the 'vlfeat' toolbox by Vedaldi [3] and 'PRTools' toolbox [4].

3.1 Bag of Words

Bag of words in image classificationas as used in [5], refers to the concept of transforming images as collection of words. It is a sparse vector (histogram) of occurrence counts of words. Once the features are extracted from the image, it is important to transform them into useful information. In our approach, keypoints are extracted from the images using different algorithms. Each keypoint has a corresponding descriptor, which corresponds to a data point in this descriptor feature space. Many such data points can be obtained from different images. A clustering algorithm, K-means, was used to extract useful information from the set of all these data points to be used as vocabulary for describing an image.

In one such approach, all the images in the training dataset were used to build the vocabulary. However, there are over 200 images in the training set and most of the images contain negative results for 90% of the classes. In order to reduce redundancy, only 6 positive images per class were used to build the vocabulary. This reduced the total number images to create the bag of features to 60. This was simply achieved by introducing a counter in the bag of features function that only used the first 6 positive images for each class. This in turn also reduced the computational time to build the dictionary. Once bag of words is formed, the vocabulary is used to train the system using the training data.

3.1.1 Hierarchical Integer K-Means

For building the bag of words vocabulary K-Means algorithm was used. K-means was proposed by Hartigan and Wong in [6]. K-means is a popular technique in image classification [7] because it tends to obtain clusters with comparable spatial extent. K-means is a clustering method for vector quantization. It aims at partitioning n features into k clusters. Each feature is associated with the nearest (mean) cluster. The algorithms starts by placing the clusters randomly in feature space. Then each feature is associated with its closest neighboring cluster and the cluster center is updated. This can be considered as approximating the features with one of the means so as to create the best dictionary/codebook to quantize the vector data.

In our implementation *vl_hikmeans* function from the 'VLFeat' toolbox was used. The hierarchical K-means algorithm applies the K-means algorithm, but in integer space, recursively in order to cluster the data together. This is designed to work efficiently on large datasets which this task is dealing with. The number of clusters was kept to specified value. The function returns the clusters in form of a tree structure.

3.2 SIFT

The most important step in image classification is to extract accurate features from the images. Scale invariant feature transform (SIFT), proposed by Lowe in [8], [9] and [10], is a popular technique used to extract keypoints and their descriptors from an image. The main advantage of SIFT is that it is scale and orientation invariant. It is also partially invariant to affine transformations and illumination changes.

SIFT extracts keypoints from the image by building a pyramid of difference of gaussian images. SIFT first creates a pyramid structure by smoothing the image with a gaussian window of different degrees and then reducing the size of the image and repeating the same process over. Difference of gaussians are computed from the scale-space representation of the image in order to extract the extrema of the pyramid structure to be used as keypoints.

After the keypoints have been extracted, their corresponding descriptors are computed. They are calculated from a histogram of the oriented gradi-

ents of the image around the keypoint. Typically, a 16x16 window is opened around the keypoint in the scale at which it was detected. This window is further sub-divided into small 4x4 sub windows. 8 bi-linearly weighted dominant orientations are extracted from each subwindow. All the 8 dominant orientations are stacked together to constitute the descriptor. This leads to a descriptor of size 128 for each keypoint. In our implementation, we used the *vl_sift* function form the 'VLFeat' toolbox. This function outputs the keypoint locations and the corresponding descriptors.

3.3 HOG

Another popular technique of feature extraction is the histogram of oriented gradients (HOG). It was originally proposed by Dalal and Triggs in [11]. Localized appearances in an image can be described using gradient directions. The main idea here is to count the number of occurrences of gradient orientations in localized sub-divisions of the image. The main difference between SIFT and HOG is that HOG is computed on a denser grid of uniformly spaced regions.

Images are divided into small connected sub-regions. A histogram of gradients is computed for each sub-region which are concatenated together to form the descriptor of the image. Contrast normalization on local histograms is used to improve invariance to illumination changes. It is also invariant to geometric transformations.

In our implementation, we used the *vl_hog* function of the 'VLFeat' toolbox. The function takes the image and the cell size as inputs and outputs the HOG descriptors. The size of the descriptor per image depends on the image size divided by the cell size and the number of descriptors.

3.4 Nearest Neighbor

After extracting the features and building up the vocabulary to train the system, it was time to choose a classifier. K- nearest neighbor is perhaps the most popular classifying technique in computer vision owing to its simplicity. It dates back to 1697, when it first appeared in [12]. It is a non-parametric method to classify objects/images.

In image classification, the input comprises of K closest training sets and the output is a class membership defined by a vote of its neighbors in the training set. It constitutes of computing probability of matches associated with each class. If the probability is 1, it is simply assigned that particular class, otherwise the image/object is associated with the class most common in the K best neighbors. For classification, it is common to assign weight to the contributions of the neighbors such that the nearer neighbors contribute more to the overall weight of the classification task. In this implementation, the version of nearest neighbor given in the skeleton code was used. The only disadvantage of this technique is that it is sensitive to local structure of the data. As a results, other classifiers were observed to serve better in terms of accuracy.

3.5 SVM

A support vector machine (SVM) is a supervised learning model that can be used to train and analyze the data [13]. Although it is a non-probabilistic binary linear classifying technique, it can be used as a non-linear classifier with the help of kernels. Provided annotated dataset and features belonging to each training image, SVM can build a framework that can assign each new entry to one of the classes. Linear SVM is defined in terms of a hyperplane in the feature space. Each new image is then mapped into the same feature space and associated with one of the classes based on where it lies with respect to the hyperplane.

3.5.1 Kernel SVM with Chi2 kernel

Since, we have to deal with more than two classes, higher SVM dimensionality is required. Traditional classifiers are often seen to generalize poorly because of the high dimensionality of the feature space. Kernel based SVM classifiers have been proven to perform better in such cases [14].

In our implementation, SVM functions in 'VLFeat' were used. First features were extracted from the images, which was the input to the *vl_svmdataset* function. This function wraps the feature matrix into a data structure supported by SVM functions. The trick here is to use a homogeneous kernel map to extend the scope of the data. Since the newly expanded data does not need to be stored in physical memory, significant memory optimization

is possible [15].

The idea here is to create an approximated kernel map known as the Chi2 kernel. Each point/feature is expanded into a vector of dimension '2*N+1' where N depends on the dimensions of the kernel. All the mapped vectors are then stacked together to give the dataset used to train the system using 'vl_svmtrain' function. The output of this function is the weight and the bias vectors. During testing phase these generate a score for each image that is used to classify it.

3.6 ADABOOST

Adaptive Boosting (ADABOOST) is another classification technique that has gained popularity in recent publications. It was first published in [16] by Freund and Schapire. It has the ability to adapt other classifiers into its own framework in order to improve the overall performance. It starts by fitting a classifier on the dataset, followed by additional copies of the classifier on the same data. During the latter process, the weights of the classifiers are adjusted according to the misclassification judgments previously made by them. This allows the subsequent classifiers to focus on the misclassified/difficult data. In other words, it combines the outputs of many weak classifiers to attain higher classification accuracy. The only drawback of the algorithm is that it is sensitive to outliers and noisy data. In our implementation, the Adaboost functions in the PRTools [4] toolbox were used. Priors are set using the *setprior* function, which are fed into the *adaboostc* framework to train the system.

3.6.1 Linear Perceptron

A perceptron is an algorithm mostly employed in supervised binary classification. It was the first artificial neural network. It makes its decision based on a linear predictor, which associates the weights with the feature vectors. It processes elements in the train dataset separately [17].

The algorithm starts with an initial estimate of the line/plane separating the classes and updates this guess whenever it makes a mistake in order to correct itself. After the update, all the previously correctly classified feature vectors are re-run to see if they are coherent with the changes before

continuing with the rest of the feature vectors. This is a very simplistic classifier (implemented by PRTools *perlc* function), and several of these simple ones are combined in our implementation in the ADABOOST framework described in the previous section.

4 Image Features

In this section we present the different types of image features used in the given classification problem. The evolution of the project work through these several choices is shown along with the produced results on the development dataset. In this dataset the training is performed on the training set and the testing is performed on the validation set. The performance metric is the area under the generated ROC curve (AUC). The work done in the following sections and the subsequent results helped in selecting the approaches for the final object classification test.

4.1 SIFT

The SIFT method provides image features which being robust, distinct and repeatable are very well-suited for the object classification task at hand. In this project we have used the VLFeat open source library and the *vl_sift* fuunction to obtain SIFT features from an image. The detected keypoints and 128 dimensional SIFT feature vectors are returned by the *vl_sift* function. These local features are then used in the Bag of Words model described before and representation of a new image can be obtained.

4.1.1 Candidate selection for Bag of Words

In the bag of words model, the selection of the data points used in the K-Means clustering process is an important factor. On one hand a good representation of the data in the vector space is needed and on the other the computation time increases with the number of data points. The SIFT method produces on an average more than 100,000 keypoints and hence as many data points for each training image. Using all such data points from all training images is a memory intensive operation hence not practical at all.

So there are two options to have a proper set of data points to be used in the K-Means process. The first one is to use all the training images,

but limit the number of keypoint descriptors taken per image randomly to a fixed number. We used one such approach with a maximum number of 200 per image. Another approach is to use all keypoints in an image but limit the number of images used to only a certain number of positive examples per image class in the training set. We used 6 positive examples per class. The comparison of both the methods is given in Table 1. In both cases SIFT features on the grayscale images are used in the BoW model with the number of clusters being 500. The classifier is the same in both cases.

Class	All training images, 200 descriptors per image	6 positive images per class, all descriptors
	AUC	AUC
Bicycle	0.883	0.863
Bus	0.687	0.662
Car	0.782	0.817
Cat	0.786	0.738
Cow	0.882	0.859
Dog	0.717	0.686
Horse	0.529	0.623
Motorbike	0.729	0.652
Person	0.552	0.621
Sheep	0.772	0.816
Average	0.7319	**0.7337**

Table 1: Classification performance for different BoW candidate selection

From the table it can be seen the performance in the second case on an average is slightly better, as random selection of 200 keypoints can lead to some important features being missed. However the values of average AUC are very similar, so that is not a key factor. The key factor is that in the second approach the run time does not increase with the increase in training data, so it is selected to be used in the sections to follow.

4.1.2 Combining SIFT with color

Color is an important image feature which helps in providing cues to object classification naturally to humans and hence can prove to be useful in this machine classification problem. The SIFT features which are histograms of oriented gradients in a window around the corresponding keypoints are extracted from the grayscale image. Efforts were made to combine the SIFT feature descriptor with color information in the corresponding patch or win-

dow.

The following Table 2 shows the performance for 5 different approaches used for testing the optimal method of using color in the patch descriptor. The first method combines the SIFT features of a keypoint with the R, G and B values in a 15 by 15 (approximately the size used in the *vl_sift* function without the scale factor) window through simple concatenation. The second approach instead of using RGB values uses the combination of SIFT features with a histogram of R,G and B values in the 15 by 15 window. Each such histogram has 40 bins. The third approach is to perform SIFT in R, G and B spaces separately and then combine all the descriptors generated for the image together. Another approach is to do the above but in the HSV space. The classifier and the number of clusters are same in all the cases for comparison.

Class	SIFT + Patch RGB Values	SIFT + Patch RGB Histogram	SIFT in RGB Space	SIFT in HSV Space
	AUC	AUC	AUC	AUC
Bicycle	0.703	0.868	0.893	0.909
Bus	0.640	0.678	0.741	0.788
Car	0.744	0.801	0.812	0.859
Cat	0.645	0.752	0.822	0.745
Cow	0.827	0.849	0.921	0.893
Dog	0.623	0.710	0.642	0.659
Horse	0.761	0.648	0.604	0.607
Motorbike	0.644	0.724	0.710	0.596
Person	0.547	0.643	0.574	0.632
Sheep	0.614	0.813	0.796	0.780
Average	0.6748	0.7486	**0.7515**	0.7468

Table 2: Classification performance for different combinations of color with SIFT features

Using just the RGB values introduces some spatial information on a small window but contradicts with the rotation invaraince property of SIFT features which makes them so robust. Hence the results are not that good in the first case. When histograms are used, the performance results are much better. Comparing that with performing SIFT in R,G and B spaces shows that the results are slightly better in case of the latter. This can be attributed to the fact that while creating the histograms the number of bins and the patch size are parameters which need to be tuned for optimality. Also the scale at which the keypoint is detected in the SIFT method is not considered for

the window around it during histogram creation. However in the other case, everything is managed internally by SIFT.

As in the table, performing SIFT in H,S,V space gives better classification results in some classes. HSV space emulates human perception of colors and subsequently objects so that was actually the motivation behind using this space. However taking the average into account, SIFT in R,G,B space is better. It must be stressed that for certain type of one-class classification problems using the HSV space for performing SIFT might be beneficial as observed. A hybrid version of performing SIFT in the combined R,G,B,H,S,V space was tried but it did not give promising results compared to to the individual cases.

4.1.3 Combing SIFT with spatial information

Using the bag of words model with the SIFT descriptors as local features allows us to represent an image. But this representation does not have any spatial information. Spatial cues can serve to be important features in detecting objects in an image. In this section we present the results for introducing spatial information in our image representation. The image was divided into four quadrants, and the SIFT features in one quadrant are considered first followed by the features in the next quadrant while creating the image feature vector. This gives a spatial structure to the image feature vector. The classification results are shown in Table 3. The cluster number in K-Means and hence the feature vector size has been changed from the one in the previous section for optimality but it is the same for both these cases, as also is the classifier used.

Class	SIFT in RGB Space	SIFT in RGB Space + Spatial ordering
	AUC	AUC
Bicycle	0.886	0.875
Bus	0.812	0.760
Car	0.839	0.810
Cat	0.783	0.810
Cow	0.913	0.887
Dog	0.665	0.723
Horse	0.553	0.530
Motorbike	0.730	0.697
Person	0.596	0.598
Sheep	0.806	0.820
Average	**0.7583**	0.7510

Table 3: Classification performance for combining SIFT with spatial ordering

Though spatial ordering improves the results in certain classes like cat, dog and sheep, in terms of average classification performance SIFT without spatial ordering is better. However this spatial ordering is a promising method which we will be using in conjunction with some other features in later sections.

4.1.4 Combining SIFT with texture

Texture is an important image property which in our previous lab assignments have shown to be helpful in image classification tasks as a global image feature. In this Bag of Words model, an effort was made to include the local texture feature of the patch with the SIFT descriptor. The following table 4 shows the performance of performing SIFT in R,G,B and image standard deviation space compared to only SIFT in R,G,B space as described in the previous section. For each pixel the standard deviation is computed over a 7 by 7 square window using MATLAB function *stdfilt* to create the standard deviation image as a simple measure of texture. Then SIFT is performed on this image along with being performed on the R,G and B images and the descriptors are combined for the entire image.

Class	SIFT in RGB Space	SIFT in RGB + Std. Deviation Space
	AUC	AUC
Bicycle	0.886	0.886
Bus	0.812	0.773
Car	0.839	0.819
Cat	0.783	0.787
Cow	0.913	0.915
Dog	0.665	0.654
Horse	0.553	0.608
Motorbike	0.730	0.801
Person	0.596	0.565
Sheep	0.806	0.802
Average	0.7583	**0.7610**

Table 4: Classification performance for combining SIFT with texture

The results show that introducing texture in the descriptor does improve the classification capability. Other local texture descriptors like GLCM can be used, but that would add to the execution time because computing GLCM values locally for image patches involves more computational complexity than calculating pixel variances as used in this case.

4.1.5 SIFT: Sparse vs Dense

In the previous sections a sparse version of SIFT has been used, where the *vl_sift* function first detects keypoints which are extrema in the difference of Gaussian scale space and then extracts SIFT descriptors around those points. With this approach, many image patches which belong to the background (having low-contrast) and ones that are on the edges are excluded. A dense version of SIFT is available in the VLFeat *vl_dsift* function but it produces too many data points for the K-Means clustering. The number of patches extracted from an image can be controlled by a step parameter. But in this dense approach the Gaussian scale space is not used and the image needs to be pre-smoothed at the required level of scale, which is a difficult parameter to optimize. Hence a better and effective approach to having a denser representation is to use a HOG feature approach as described in the following

section.

4.2 Dense HOG with Spatial Ordering

From the earlier section, it is observed that histograms of oriented gradients as used in SIFT are useful local descriptors which can be used for object classification with good results. We have also seen the benefits of using a dense representation rather than a sparse one. Spatial information can be a good cue for some object classes however spatial ordering does not fit well with the SIFT method itself.

The Histogram of Oriented Gradients (HOG) features have been used in this section. This method uses the similar descriptor idea as in SIFT but without the keypoint detection in scale space. The HOG features are computed for contiguous non-overlapping patches of the image using the *vl_hog* function in VLFeat. The size of the patch is chosen to be 8x8. The color information is introduced while computing the gradients by applying the *vl_hog* function to the RGB image as a whole. From our experiments, we saw that combining texture information in HOG does not improve the results. Another important aspect of this approach is to introduce spatial ordering while creating the image feature vector from the HOG features of the patches. They are in a particular order of their location in the image. The following table shows the results for this approach compared with the best results obtained from SIFT method. The number of clusters in K-Means and the classifier used are the same in both cases.

Class	SIFT in RGB + Std. Dev. Space	Dense HOG + Spatial ordering
	AUC	AUC
Bicycle	0.886	0.856
Bus	0.773	0.922
Car	0.819	0.857
Cat	0.787	0.763
Cow	0.915	0.866
Dog	0.654	0.739
Horse	0.608	0.708
Motorbike	0.801	0.699
Person	0.565	0.635
Sheep	0.802	0.855
Average	0.7610	**0.7900**

Table 5: Classification performance for HOG with spatial ordering

The results show that though SIFT outperforms this HOG + spatial approach in a couple of test classes, on an average the latter is better.

5 Number of clusters

The number of clusters used in the Bag of Words model is an important parameter in the classification process. That determines the size of the image vector in the final feature space and affects the classification output. A value too low would not be a proper representation of the image whereas a value too high would create redundancy and reduce differentiability negative affecting the classification accuracy. Methods like Principal Component Analysis can be used to get a reduced optimum set of features. However in this approach we have relied on the feature selection process and on the ability of the selected classifier to correctly identify objects in the images.

Still there is some manual experimentation needed to select an optimal number of clusters for the K-Means algorithm. Several values were tested however in this section we provide performance comparison of two cases, 500 vs 1000 clusters for both the best approaches of SIFT and HOG. Table 6 compares the SIFT method for 500 and 1000 clusters. Table 7 compares the HOG method for 500 and 1000 clusters.

Class	SIFT in RGB + Std. Dev. Space - 500 Clusters	SIFT in RGB + Std. Dev. Space - 1000 Clusters
	AUC	AUC
Bicycle	0.853	0.886
Bus	0.778	0.773
Car	0.811	0.819
Cat	0.812	0.787
Cow	0.906	0.915
Dog	0.638	0.654
Horse	0.600	0.608
Motorbike	0.716	0.801
Person	0.533	0.565
Sheep	0.795	0.802
Average	0.7442	**0.7610**

Table 6: Classification performance for SIFT for different number of clusters

Class	Dense HOG + Spatial ordering - 500 Clusters	Dense HOG + Spatial ordering - 1000 Clusters
	AUC	AUC
Bicycle	0.860	0.856
Bus	0.926	0.922
Car	0.852	0.857
Cat	0.742	0.763
Cow	0.870	0.866
Dog	0.765	0.739
Horse	0.713	0.708
Motorbike	0.684	0.699
Person	0.622	0.635
Sheep	0.848	0.855
Average	0.7882	**0.7900**

Table 7: Classification performance for HOG for different number of clusters

In both cases, the optimal number of clusters is 1000, so we have used that in all sections to follow. It must be emphasized that the performance does not improve with increasing the cluster number, 1000 is an optimal number for our experiments also considering the run time.

6 Classifier

As described before, we have used 4 classifiers in this problem - Nearest Neighbor, Support Vector Machine, Kernel Support Vector Machine with Chi2 kernel and Adaboost with linear perceptron classifier. The most trivial

case is the nearest neighbor classifier in which case for a test sample the distance of the closest negative example and the distance to the closest positive example in the training set is computed and the ratio of the two gives a classification probability of the sample. This approach allows the ROC curve to be constructed by choosing different threshold values.

For implementation of SVM, VLFeat *vl_svmtrain* function has been used. The optimal regularization parameter is found to be 0.00001 from experiments, and this also controls the maximum number of iterations. For kernel SVM, the VLFeat function *vl_svmdataset* has been used in conjunction with *vl_svmtrain*. The order of the Chi2 kernel is selected to be 4, which yielded the best results. The *adaboostc* function has been used from PRTools where several simple classifiers are combined to get a good classifier. The simple classifiers are chosen to be the linear perceptron (PRTools *perlc* function with 300 iterations) which are combined by the default way of weighted voting.

Table 8 shows the classification results obtained with the classifiers on the best SIFT version from earlier sections. Table 9 shows the same for the best HOG features.

Class	NN	SVM	Kernel SVM	Adaboost
	AUC	AUC	AUC	AUC
Bicycle	0.875	0.886	0.853	0.848
Bus	0.658	0.773	0.776	0.630
Car	0.756	0.819	0.865	0.791
Cat	0.671	0.787	0.835	0.749
Cow	0.830	0.915	0.894	0.916
Dog	0.603	0.654	0.669	0.710
Horse	0.412	0.608	0.643	0.672
Motorbike	0.564	0.801	0.814	0.776
Person	0.593	0.565	0.580	0.514
Sheep	0.716	0.802	0.794	0.794
Average	0.6678	0.7610	**0.7723**	0.7400

Table 8: Classification performance for different classifiers with best SIFT features

Class	NN	SVM	Kernel SVM	Adaboost
	AUC	AUC	AUC	AUC
Bicycle	0.789	0.856	0.884	0.794
Bus	0.754	0.922	0.914	0.932
Car	0.869	0.857	0.912	0.836
Cat	0.786	0.763	0.798	0.754
Cow	0.789	0.866	0.860	0.845
Dog	0.681	0.739	0.790	0.757
Horse	0.731	0.708	0.871	0.789
Motorbike	0.674	0.699	0.776	0.734
Person	0.582	0.635	0.665	0.648
Sheep	0.708	0.855	0.861	0.819
Average	0.7363	0.7900	**0.8331**	0.7908

Table 9: Classification performance for different classifiers with best HOG features

The results show that classifiers SVM, Kernel SVM and Adaboost are all better than nearest neighbor approach which is quite simplistic. Support vector machines perform quite well. But the best results are obtained with Kernel SVM which shows that the data is not exactly linearly separable and hence mapping to a higher dimensional kernel space is highly beneficial in the classification task. As expected, Adaboost combines simple classifiers like the linear perceptron and achieves results much better than simple classifiers like Nearest Neighbor. In one case it outperforms even SVM. One possible approach may be to combine strong classifiers like SVM with Adaboost but that defeats the actual purpose of combining weak classifiers to improve classification accuracy. It was tested and as mentioned it did not give good results. Summarizing all the results and observations, kernel SVM is the classifier which performs best and is selected for running the final set of tests on the Pascal dataset.

7 Results

In this section we present the results of our two best approaches on the 10 class object classification on the Pascal dataset given to us in the project.

The training of the classifier has used the images in both the train and the validation set, so 525 images in total. The bag of words process uses descriptors from only the train set. The classification testing has been done on the test set which comprises of 523 images.

Based on the observations and derived conclusions from the earlier sections, we have settled on two feature extraction approaches. One is SIFT based. It uses Sparse SIFT in R,G,B and Standard deviation space with no spatial ordering. The second is HOG feature based. It uses dense HOG features computed on an RGB image with a spatial ordering on those features. The Bag of Words model has been used in both cases where descriptors from 6 positive examples per train class are used to build the dictionary. The number of clusters used in the K-Means clustering process is 1000. The classifier used in both cases is also the same, which is the Kernel SVM with Chi2 kernel with order 4. Table 10 shows the results obtained with both approaches.

Class	Sparse SIFT in RGB + Std. Dev. Space	Dense HOG + Spatial ordering
	AUC	AUC
Bicycle	0.863	0.892
Bus	0.902	0.942
Car	0.869	0.903
Cat	0.821	0.877
Cow	0.925	0.921
Dog	0.697	0.750
Horse	0.671	0.846
Motorbike	0.799	0.803
Person	0.615	0.626
Sheep	0.849	0.920
Average	0.8011	**0.8480**

Table 10: Classification performance of two best approaches on the whole test set

From the table it is seen that Dense HOG features with spatial ordering performs the best. Overall the classification results are good on all classes except 'Person' which is a difficult recognition problem because of the wide intra-class variability. As seen in Table 2 we achieved slightly better results

on that class using HSV space. In a multi-class classification problem decision choices need to be made considering the average performance over all classes compared to performance on a single class, so RGB was used.

As for the speed of the entire classification task, for the second HOG based approach the bag of words process with K-Means clustering took about 25 minutes. The training and the testing including the time to generate the corresponding feature vectors took approximately 15 minutes. These values are generated on a machine with Intel(R) Core(TM) i5-5200U CPU 2.20 GHz with 4 GB RAM. Times for the first SIFT based approach were also similar.

So from the classification results it can be concluded that selection of a proper set of descriptive yet robust features coupled with the use of a good well-tuned classifier and proper set of associated parameters in the entire process (like the cluster number) can lead to effective object classification in real complex images.

8 Organization and Development of the Coursework

The project was completed in the 6 weeks provided for the work. The first week was reserved for understanding the Pascal challenge skeleton provided to us by the instructors and the 'VLFeat' and 'PRTools' libraries. The next couple of weeks were dedicated to applying different feature extraction techniques and observing the effects. After trying out different feature extraction techniques weeks were assigned to implement different classifiers.

Since, now the code had developed enough to start formalizing the results, one week was allocated on tabulating the results and implementing the framework on the testing set. One week was assigned for this because the system takes about an hour (or several depending on the approach) to train and classify the results and an extensive analysis was required to select the best frameworks. The last week was dedicated to report writing.

9 Conclusion and Future work

This project involved recognizing objects from a number of different visual classes using the doctrine followed in the Pascal Challenge 2006. Ten object classes were classified with an average Area under the ROC curve value of **0.848**. HOG and SIFT methods were used to extract features from the images and they were improved by addition of color, texture and/or spatial information. A bag of words strategy was used to build feature vectors. A detailed analysis of different classifying techniques used was also presented, with kernel SVM producing the best results.

In this project we focused on the task of object classification in images, with extensive tests and comparisons to achieve best possible results. In future we would like to extend this work into the task of object detection/localization, as the bounding boxes for the objects in image annotations are also provided.

References

[1] Datta, Ritendra, Jia Li, and James Z. Wang. "Content-based image retrieval: approaches and trends of the new age." In Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval, pp. 253-262. ACM, 2005.

[2] Everingham, Mark, Andrew Zisserman, Christopher KI Williams, and Luc Van Gool. "The PASCAL visual object classes challenge 2006 (VOC2006) results." (2006)

[3] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[4] R.P.W. Duin, P. Juszczak, P. Paclik, E. Pekalska, D. de Ridder, D.M.J. Tax, S. Verzakov PRTools4.1, A Matlab Toolbox for Pattern Recognition, Delft University of Technology, 2007.

[5] Li Fei-Fei, P. Perona, "A Bayesian Hierarchical Model for Learning Natural Scene Categories". IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), 2005.

[6] Hartigan, J. A.; Wong, M. A. (1979). "Algorithm AS 136: A K-Means Clustering Algorithm". Journal of the Royal Statistical Society, Series C 28 (1): 100108.

[7] L. Zhu, A. Zhang, A. Rao, and R. Srihari, "Keyblock: An Approach for Content-based Image Retrieval", Proc. ACM Multimedia, 2000.

[8] David G. Lowe. "Object recognition from local scale-invariant features". In International Conference on Computer Vision, pages 11501157, 1999.

[9] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". International Journal of Computer Vision, 60(2):91110, November 2004.

[10] Matthew Brown and David G. Lowe. "Invariant features from interest point groups". In British Machine Vision Conference, pages 656665, Cardiff, Wales, 2002.

[11] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In CVPR, pages 886893, 2005.

[12] Cover, Thomas M., and Peter E. Hart. "Nearest neighbor pattern classification". Information Theory, IEEE Transactions on 13, no. 1 (1967): 21-27.

[13] E. Osuna, R. Freund, and F. Girosi. "Support vector machines: Training and applications". A.I. Memo (in press), MIT A. I. Lab., 1996.

[14] Chapelle, O., Haffner, P. and Vapnik, V.N., 1999. "Support vector machines for histogram-based image classification". Neural Networks, IEEE Transactions on, 10(5), pp.1055-1064.

[15] A. Vedaldi and A. Zisserman "Efficient Additive Kernels via Explicit Feature Maps", Proc. CVPR, 2010.

[16] Y. Freund, R. Schapire, "A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting", 1995.

[17] Cristianini, Nello and John Shawe-Taylor, "An Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods". Cambridge, England: Cambridge University Press, 2000.