University of Girona

Spain

Scene Segmentation and Interpretation
Pascal Project Challenge

Submitted by :
Mr. Shubham WAGH
Mr. Nayee Muddin Khan
DOUSAI

Contents

List of Figures

1 Objective

The main objective of this project ”PASCAL” is to detect objects from the given Image sets and plot the ROC curve for given object classes. For training and testing classes, we have provided with ten different object classes as : bicycle, bus, car, cat, cow, dog, horse, motorbike, person and sheep.

2 Introduction

PASCAL project was started in the year 2005 at University of Oxford [12] with few other collaboration of universities and Professors. Since then it was held every year till the end of 2012 year. For this given project we have the data from PASCAL 2006 which consists of ten object classes as bicycle, bus, car, cat, cow, dog, horse, motorbike, person and sheep. The imagesets consists of four sets of images for both training and testing classification. They are called as:

- *train*: which consists of the training data
- *val*: Validation data
- *trainval*: Combination of train and val data
- *test*: testing data to match from the trained data

For the above classification, We are using 256 data classes for the training and 523 images for testing. To make the task easier and we have provided the ground truth of all the training images which are stored in *.txt* format. When we open this text file we can observe that there will be two different numbers as the format $000XXX$ or $1, 0, -1$ which are called as image identifier and ground truth labels. The ground representation is explained as:

- **Negative Value(-1):** It doesn't consists of that particular class of interest for the given image
- **Positive Value(1):** It predicts there should be atleast one class of interest for the given image
- **UnPredicted Value(0):** When the text file states zero it shows the difficulty in detecting the objects

2.0.1 Project Framework

PASCAL Development Kit, MATLAB, VL_FEAT, PRTOOLS and CNN AlexNet

3 Strategy Analysis

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

1. Read train images, extract image features from 8 positive examples of each class. Run Bag of words model (K-Means) on this data to get dictionary.
2. For each class, build feature vectors for all training images using the same feature extraction technique and the created dictionary. Train chosen classifier.
3. Build feature vectors for all test images and test the classifier on this test data.
4. Plot ROC curve and compute AUC. Start training for next class until all classes done.

4 Design and Implementation

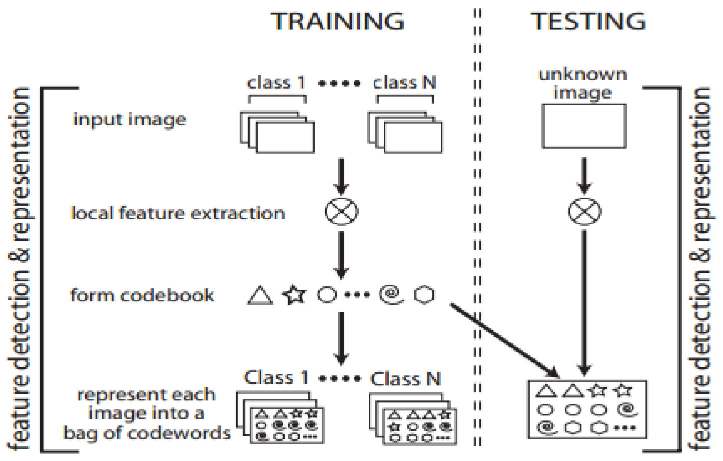


Figure 1: Flow chart of Implementation (Image Source : Link)

4.1 Bag of Visual Words

In computer vision, the bag-of-words model (BoW model) can be applied to image classification, by treating image features as words. A bag of words is a sparse vector of occurrence counts of words; that is, a sparse histogram over the vocabulary [3]. In computer vision, a bag of visual words is a vector of occurrence counts of a vocabulary of local image features. Here, we extract keypoints from the images using algorithms mentioned ahead in the report. These keypoints have a corresponding descriptor, which is a data point in the feature space. Clustering algorithm (ex. K-means) is then applied to extract useful information from the set of all these data points to be used as vocabulary for describing an image.

For this project, instead of using all 256 images in the training set, we use only 8 positive images (first positive 8 images) per class to build the dictionary, as 90% of the images in the training set are negative. Hence avoids redundancy. This reduced the total number of images for bag of visual words to 80. Also it reduced the computation time for building the dictionary. Once the dictionary is built, this bag of features is then used to train the system using the training data.

4.1.1 Clustering using K-means

For building the bag of words dictionary K-Means algorithm was used. k-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into k clusters. 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 to quantize the vector data.

For the project, we used *vl_hikmeans* function from **VLFeat** toolbox. 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 function returns the clusters in form of a tree structure.

4.2 SIFT

Scale-invariant feature transform (SIFT) is an algorithm in computer vision to detect and describe local features in images. It is a popular technique to extract keypoint and descriptors from an image [4]. The main advantage of SIFT is that it is scale and orientation invariant. It is also partially invariant to affine transformations and illumination changes. Also it is robust to local geometric distortion.

A Gaussian pyramid is constructed from the input image by repeated smoothing and subsampling, and a difference-of-Gaussians pyramid is computed from the differences between the adjacent levels in the Gaussian pyramid. Then, interest points are obtained from the points at which the difference-of-Gaussians values assume extrema with respect to both the spatial coordinates in the image domain and the scale level in the pyramid. After the keypoints have been extracted, their corresponding descriptors are computed. They are calculated from a histogram of the oriented gradients of the image around the keypoint. The scale at which keypoint is located, a 16×16 window is opened around it. It is further sub-divided into small 4×4 window. Then 8 dominant orientations are extracted which are stacked together to constitute the descriptor. This leads to a descriptor of size 128 for each keypoint. Here, we used the *vl_sift* function form the **VLFeat** toolbox for extracting sift features from the image. This function outputs the keypoint locations and the corresponding descriptors.

4.3 Histogram of Oriented Gradients

The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image [5]. HOG decomposes an image into small squared cells, computes an histogram of oriented gradients in each cell, normalizes the result using a block-wise pattern, and return a descriptor for each cell [8]. Stacking the cells into a squared image region can be used as an image window descriptor for object detection, for example by means of an nearest neighbor, SVM etc.

For the project, 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.

4.4 Classifiers

4.4.1 L2 norm distance classifier

L2 norm distance classifier is the trivial classifier also the default one in Pascal's development kit. It computes ratio of L2 distance betweeen nearest positive (class) feature vector and nearest negative (non-class) feature vector. This ratio is used as confidence score for correct classification.

4.4.2 Multi-class SVM classifier

The key idea of SVM is based on the notion of margin. For a binary classifier, the margin of a hyperplane for a dataset is the distance between the hyperplane and the data point nearest to it. Equivalently,margin is the smallest distance between a positive example and a negative example. To generalize SVM to multiclass case, the notion of margin is generalized in the multiclass scenario [9].

Here, we used *fitcecoc* from MATLAB to build the classification model using multiclass SVM and *predict* function to compute the cofidence scores using the model and the test data.

4.4.3 Discriminative Restricted Boltzmann Machine classifier

Restricted Boltzmann Machines are generative stochastic models that can model a probability distribution over its set of inputs using a set of hidden (or latent) units. They are represented as a bi-partitie graphical model where the visible layer is the observed data and the hidden layer models latent features. By trying to reconstruct the input vector, we can make the RBM learn a different representation of the input data. This representation is then used in a higher classification algorithm.

For the project, we use *drbmc* function from **PRTools** toolbox. The classifier trains a discriminative Restricted Boltzmann Machine (RBM) on training dataset. The discriminative

RBM can be viewed as a logistic regressor with hidden units. The discriminative RBM has N hidden units. It is trained with L2 regularization using regularization parameter L. New objects are classified in the same way as in a logistic regressor, using a softmax function over the labels. DRBMs are universal approximators of conditional distributions with binary inputs.

4.4.4 SVM classifier using Kernel Chi2

Traditional classfiers are often seen to generalize poorly because of the high dimensionality of the feature space. Kernel based SVM classfiers have been proven to perform better in such cases. Here, we used SVM functions from **VLFeat** toolbox. First features were extracted from the images, which was the input to the *vl_svmdataset* function. It wraps the feature matrix into a data structure supported by SVM functions. A homogeneous kernel map is used to extend the scope of the data [6]. This leads to significant memory optimization as the expanded data does not need to be stored in the memory.

Firstly, an approximated kernel map known as the Chi2 kernel. Each feature is expanded into a vector of dimension $2N + 1$ 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.

4.5 Convolutional Neural Network Approach

A Convolutional Neural Network (CNN) is a powerful machine learning technique from the field of deep learning. CNNs are trained using large collections of diverse images. From these large collections, CNNs can learn rich feature representations for a wide range of images [10]. These feature representations often outperform hand-crafted features such as HOG, LBP, or SURF. An easy way to leverage the power of CNNs, without investing time and effort into training, is to use a pre-trained CNN as a feature extractor.

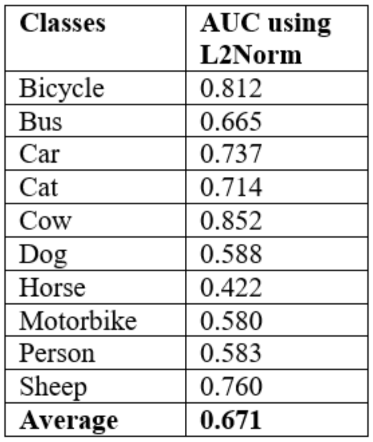
For the project, we used pre-trined AlexNet model. AlexNet is a pretrained Convolutional Neural Network (CNN) that has been trained on approximately 1.2 million images from the ImageNet Dataset. The model has 23 layers and can classify images into 1000 object categories. For the project, we classify the given images into categories using a kernel SVM trained with CNN features extracted from the images. The difference here is that instead of using image features such as HOG or SIFT, features are extracted using a CNN. The classifier trained using CNN features provides accuracy more than 90%, which is higher than the accuracy achieved using bag of features. Further details on implementation is explained in the experimental and results section.

5 Experimental and Result analysis

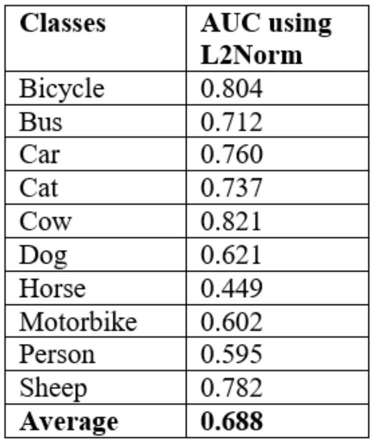
In this section we present the different types of image features used in the given classification problem. The evolution of the project work through these several choices is shown along with the produced results on the development dataset. In this dataset the training is performed on the training set 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.

5.0.1 Using SIFT color feature

Firstly, we make a *extractsiftcolorfd* function which extracts sift feature descriptor from each channel of the color image and then concatenate these descriptors. We also build a Bag of Words (BoW) of 400 and 500 clusters using the above mentioned feature. This is then used to train the system with the the training dataset for classification. For classification we used L2 norm distance classifier. Above table in fig.(2) shows the Area Under Curve (AUC) values for each class and an avergae AUC.



Classes	**AUC using L2Norm**
Bicycle	0.812
Bus	0.665
Car	0.737
Cat	0.714
Cow	0.852
Dog	0.588
Horse	0.422
Motorbike	0.580
Person	0.583
Sheep	0.760
Average	**0.671**



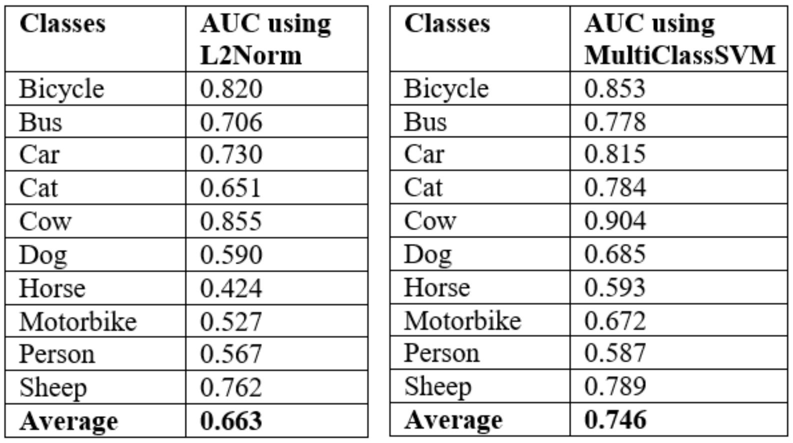
Classes	**AUC using L2Norm**
Bicycle	0.804
Bus	0.712
Car	0.760
Cat	0.737
Cow	0.821
Dog	0.621
Horse	0.449
Motorbike	0.602
Person	0.595
Sheep	0.782
Average	**0.688**

Figure 2: **LEFT**: AUC with SIFTcolor feature, L2 norm classifier, 400 clusters; **RIGHT**: AUC with SIFTcolor feature, L2 norm classifier, 500 clusters

We observe that keeping 500 clusters gave us good result. For further experiments we write our own classifier apart from the default one to check the performance of the system.

5.0.2 Using SIFT color and texture feature

Texture is an important feature of the image. Here, including the color SIFt feature, we also manage to compute texture of the image and concatente all these descriptors giving us SIFT color texture feature. For each pixel the standard deviation is computed over a 7×7 square window using MATLAB function *stdfilt* to create the standard deviation image as a simple measure of texture.



Classes	**AUC using L2Norm**
Bicycle	0.820
Bus	0.706
Car	0.730
Cat	0.651
Cow	0.855
Dog	0.590
Horse	0.424
Motorbike	0.527
Person	0.567
Sheep	0.762
Average	**0.663**

Classes	**AUC using MultiClassSVM**
Bicycle	0.853
Bus	0.778
Car	0.815
Cat	0.784
Cow	0.904
Dog	0.685
Horse	0.593
Motorbike	0.672
Person	0.587
Sheep	0.789
Average	**0.746**

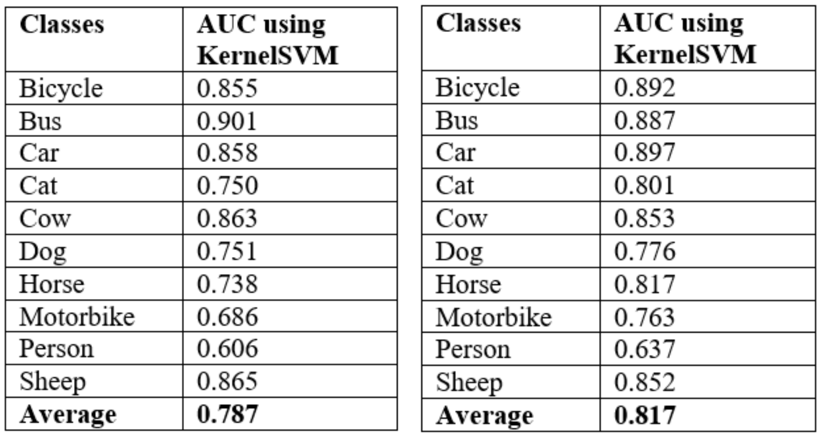
Figure 3: **LEFT**: AUC with SIFTcolorTexture feature, L2 norm classifier, 500 clusters; **RIGHT**: AUC with SIFTcolorTexture feature, Multi-Class classifier, 500 clusters

We create *extractcolortexturefd* to compute these feature descriptors. We keep the number of clusters as 500 for our dictionary but the dictionary is built using *extractcolortexturefd* feature function this time. For classification we used both L2 norm distance classifier and Mutliclass SVM classifier and compare both the results. Above table fig.(3) shows the performance for both of the classifiers. We observe that multiclass SVM outperforms L2 norm distance classifier. For further experiments we try to improvise our SVM classifier as well as we introduce HOG features.

5.0.3 Using HOG feature and kernel SVM

From the above experiments we observed that SIFT (sparse) feature does not improve the performance of the classifier to classify objects. We try to compute dense SIFT feature i.e HOG feature for the image. We create *extracthogfd* function to implement this.

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 8×8. 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. For BoW, we now compute HOG features instead of SIFT.



Classes	AUC using KernelSVM
Bicycle	0.855
Bus	0.901
Car	0.858
Cat	0.750
Cow	0.863
Dog	0.751
Horse	0.738
Motorbike	0.686
Person	0.606
Sheep	0.865
Average	**0.787**

Classes	AUC using KernelSVM
Bicycle	0.892
Bus	0.887
Car	0.897
Cat	0.801
Cow	0.853
Dog	0.776
Horse	0.817
Motorbike	0.763
Person	0.637
Sheep	0.852
Average	**0.817**

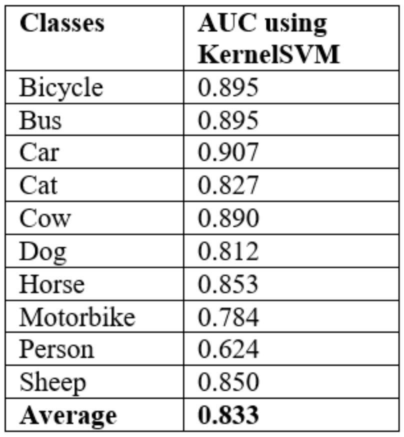
Figure 4: **LEFT**: AUC with HOG feature, Kernel SVM classifier with order = -1, 500 clusters; **RIGHT**: AUC with HOG feature, Kernel SVM classifier with order = 3, 500 clusters

For implementation of kernel 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 -1 and 3, to check the performance difference if any.

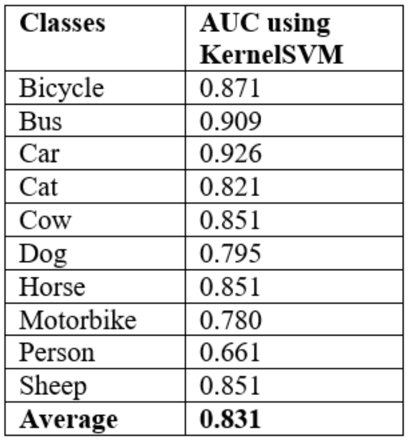
The above table fig.(4) shows the results for this approach. The number of clusters in K-Means and the classifier used are the same in both cases. We observe that 3rd order kernel outperformed the -1 order kernel. And certainly HOG features has improved the classification accuracy as well. Next we try to play with the number of clusters of BoW.

5.0.4 Increasing the number of clusters

Here, we increase the number of clusters from 500 to 800 and 1000 keeping the same classifier (Kernel SVM, order 3) and feature extraction property (HOG with spatial ordering) to build our dictionary. From the below table we observe that increasing the number of clusters (800) do improve the performance of object classifiation, but on further increasing (1000 clusters), we don't see any change in the AUC values. We also observe that computation time increases on increasing the number of clusters. For all further experiments we now use 800 clusters for BoW as it certainly improved the performance of our classification system.



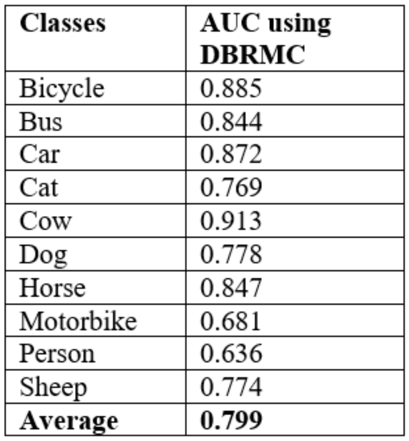
Classes	AUC using KernelSVM
Bicycle	0.895
Bus	0.895
Car	0.907
Cat	0.827
Cow	0.890
Dog	0.812
Horse	0.853
Motorbike	0.784
Person	0.624
Sheep	0.850
Average	**0.833**



Classes	AUC using KernelSVM
Bicycle	0.871
Bus	0.909
Car	0.926
Cat	0.821
Cow	0.851
Dog	0.795
Horse	0.851
Motorbike	0.780
Person	0.661
Sheep	0.851
Average	**0.831**

Figure 5: **LEFT**: AUC with HOG feature, Kernel SVM classifier with order = 3, **800** clusters; **RIGHT**: AUC with HOG feature, Kernel SVM classifier with order = 3, **1000** clusters

5.0.5 Using Deep learning classifier



Classes	AUC using DBRMC
Bicycle	0.885
Bus	0.844
Car	0.872
Cat	0.769
Cow	0.913
Dog	0.778
Horse	0.847
Motorbike	0.681
Person	0.636
Sheep	0.774
Average	**0.799**

Figure 6: AUC HOG feature, Discriminative Restricted Boltzmann Classifier, 800 clusters

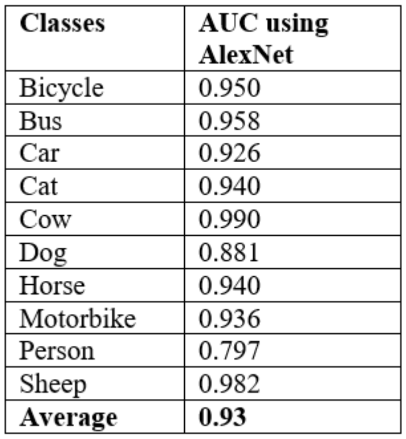
Earlier we tried many classifiers like k-nearest neighbor, multiclass SVM, kernel SVM, naive bayes etc. out of which kernel SVM outperformed others. We also tried implementing deep learning classifier from **PRTools** toolbox. We tried automatic neural classifier but results were worse than k-nearest neighbor as well. However, we implemented Discriminative Restricted Boltzmann Classifier (DRBMC) results of which were not as good as kernel SVM classifier but were better than k-nearest neighbor, naive bayes classifier. Above table shows the performance of DRBMC which gives average classification results.

We also tried to concatenate HOG and sparse SIFT features but didn't improve the performance of the classification. Hence we make an attempt for neural network approach for classification.

6 Convolutional Neural Network Approach

Till now we were using handcrafted features for our classification. Now, instead we use features from our pre-trained AlexNet- CNN model. Each layer of a CNN produces a response, or activation, to an input image. However, there are only a few layers within a CNN that are suitable for image feature extraction. The layers at the beginning of the network capture basic image features, such as edges and blobs. Features can be extracted from one of the deeper layers using the activations method. Here, we extract feature from *fc7* layer, one of the deep layer of AlexNet. This extracted feature vector is then given to our kernel SVM classifier for classification. Following table (fig.7) shows the results for this approach.

We observe that there is a big increment of approximately 10% in the values of AUC as comapred to handcrafted feature method discussed above (HOG feature and Kernel SVM). CNN certainly has boosted our performance of the classification.



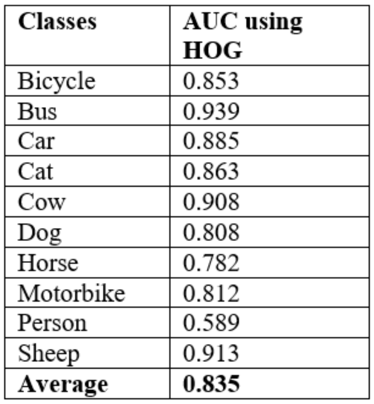
Classes	**AUC using AlexNet**
Bicycle	0.950
Bus	0.958
Car	0.926
Cat	0.940
Cow	0.990
Dog	0.881
Horse	0.940
Motorbike	0.936
Person	0.797
Sheep	0.982
Average	**0.93**

Figure 7: **LEFT**: AUC with HOG feature, Kernel SVM classifier, 800 clusters with order = 3; **RIGHT**: AUC with AlexNet, Kernel SVM classifier with order = 3

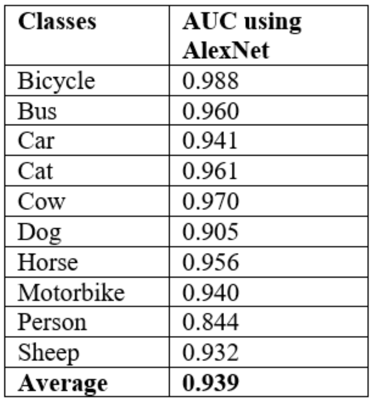
7 Final Results

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 set, so 256 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.



Classes	AUC using HOG
Bicycle	0.853
Bus	0.939
Car	0.885
Cat	0.863
Cow	0.908
Dog	0.808
Horse	0.782
Motorbike	0.812
Person	0.589
Sheep	0.913
Average	**0.835**



Classes	AUC using AlexNet
Bicycle	0.988
Bus	0.960
Car	0.941
Cat	0.961
Cow	0.970
Dog	0.905
Horse	0.956
Motorbike	0.940
Person	0.844
Sheep	0.932
Average	**0.939**

Figure 8: **LEFT**: AUC with HOG feature, Kernel SVM classifier, 800 clusters with order = 3; **RIGHT**: AUC with AlexNet, Kernel SVM classifier with order = 3

Based on the observations and derived conclusions from the earlier sections, we have settled on two feature extraction approaches. First one is HOG feature based. It uses dense HOG features computed on an RGB image with a spatial ordering on those features. The second one is features from our pre-trained CNN model AlexNEt extracted from *fc7* layer. The Bag of Words model has been used in the first case where descriptors from 8 positive examples per train class are used to build the dictionary. The number of clusters used in the K-Means clustering process is 800. The classifier used in both cases is also the same, which is the Kernel SVM with Chi2 kernel with order 3. Fig.(8) shows the results obtained with both approaches.

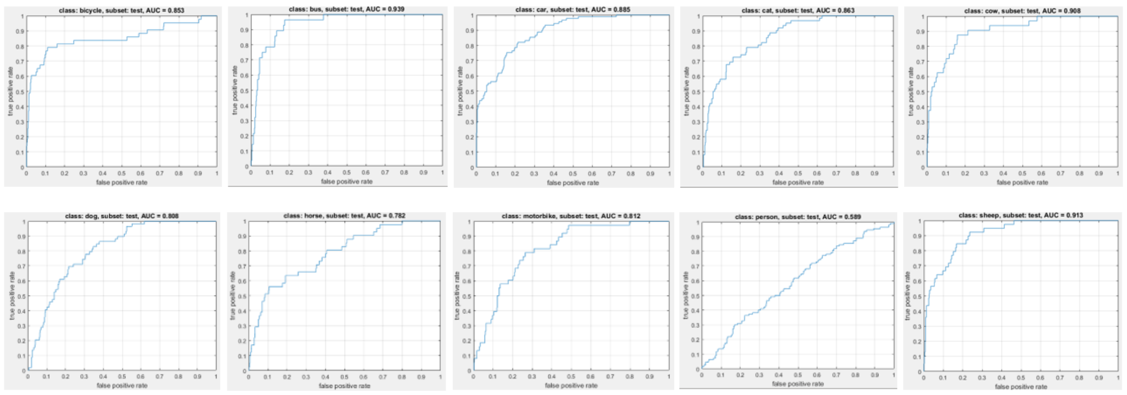


Figure 9: AUC with HOG feature, Kernel SVM classifier, 800 clusters with order = 3

As for the speed of the entire classification task, for the HOG based approach the BoW process with K-Means clustering took about 30 minutes. The training and the testing including the time to generate the corresponding feature vectors took approximately 20 minutes. While the neural network approach took about 10 minutes of testing time as AlexNet model used was pre-trained, hence training time is saved in this case. These values are generated on a machine with Intel(R) Core(TM) i5-5200U CPU 2.20 GHz with 4GB RAM. Times for the SIFT based approach were also similar.

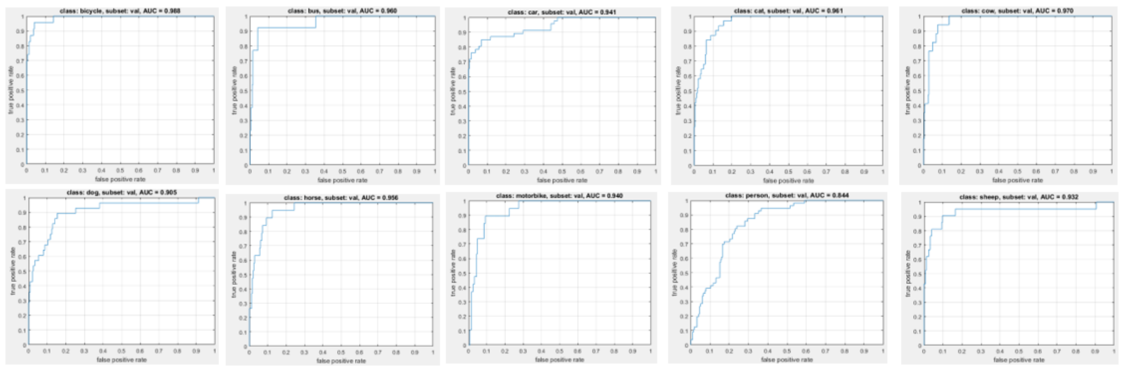


Figure 10: AUC with AlexNet, Kernel SVM classifier with order = 3

8 Organization and development

The project was completed in the 5 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. The testing of pre-trained CNN model AlexNet features was implemented in one day.

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

This project involved recognizing objects from a number of different visual classes using the template followed in the Pascal Challenge 2006. Ten object classes were classified with an average Area under the ROC curve value of **0.939** using AlexNet and **0.835** using Hand-Crafted features (HOG with spatial ordering). 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. Also features were extracted from pre-trained AlexNet CNN model. 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. Deep learning methods are certainly the future for successful classification of objects.

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.

10 References

1. http://www.vlfeat.org/
2. http://prtools.org/
3. https://en.wikipedia.org/wiki/Bag-of-words_model_in_computer_vision
4. https://en.wikipedia.org/wiki/Scale-invariant_feature_transform
5. https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients
6. A. Vedaldi and A. Zisserman ”Effcient Additive Kernels via Explicit Feature Maps”, Proc. CVPR, 2010
7. Y. Freund, R. Schapire, ”A Decision-Theoretic Generalization of on-Line Learning and an Application to Boosting”, 1995.
8. N. Dalal and B. Triggs. ”Histograms of oriented gradients for human detection”. In CVPR, pages 886893, 2005.
9. E. Osuna, R. Freund, and F. Girosi. ”Support vector machines: Training and applications”. A.I. Memo (in press), MIT A. I. Lab., 1996.
10. Image category classification using deep learning
11. AlexNet download link
12. http://host.robots.ox.ac.uk/pascal/VOC/voc2006/index.html