UNIVERSITAT DE GIRONA

SCENE SEGMENTATION

PROJECT REPORT

Pascal Image Classification Challenge

Author
Mohit VAISHNAV
Malav BATERIWALA

Supervisor
Dr. Xavier LLADÓ

Submission Last Date 29th, May

Contents

List of Figures

List of Tables

1 Objective

The primary objective of this 'PASCAL' project is to detect objects from the given pre defined Image sets and plot the ROC curve for the given object classes. We have provided with ten different object classes for training and testing different classes. We use different feature extraction techniques in this program. We make a dictionary of features using the bag of words model. The dictionary we defined is used to train the classifier using the training set of images. Then we test it with a various different algorithms for different systems. Thorough many possible setups and experimentation, we are able to achieve accurate final results, and this is a introduction to the characteristics of the bag of words or bag of features method.

2 Introduction

The Pascal Visual Object Classes (VOC) challenge is a benchmark in visual object category recognition and detection, providing the vision and machine learning communities with a standard dataset of images and annotation, and standard evaluation procedures. This challenge is organized annually from 2005 to present, the challenge and its associated dataset has become accepted as the benchmark for object detection. The main objectives of this challenge are 2 things : first to provide challenging images and high quality annotation, together with a standard evaluation methodology which can used with any dataset of training and testing so that performance of algorithms can be compared and second to measure the state of the art each year to find out which technique is better.

Applications of this problem solution can be used in several fields like object in video tracking, image retrieval from different image sets, surveillance of any object etc. For this problem, many new approaches have been surfaced during the last few years due to the increase in interest on content-based image retrieval systems in real life problems. The classification process can be supervised or unsupervised by a computer if we use a deep neural network approach. 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 provided with unsupervised learning is to specify the number of classes or to provide an estimate of what type of 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 is based on the data extracted from training samples for classifying images. In this we define everything from size and classes to the threshold for grouping the pixels together.

In our project, we have used supervised learning techniques which are known to be more accurate for recognizing classes of objects. A typical supervised classification is divided into three stages: creating a dictionary(bag of words or bag of features), 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. In the PASCAL 2006 dataset given to us, we a good definition of images and objects in the pictures. The PASCAL dataset contains 10 object classes namely : bicycle, bus, car, motorbike, cat, cow, dog, horse, sheep, person. Our main task of this project is to train the image dataset and then create an ROC curve for validation. Then we will be given a test set to check our accuracy for the final answer. We will use many different techniques to create bag of features and also different classifiers to segregate the classes separately in the image.

2.1 Bag of Words

In computer vision, the bag-of-words model (BoW model) can be applied to image classification, by treating image features as words. In document classification, a bag of words is a sparse vector of occurrence counts of words; that is, a sparse histogram over the vocabulary. In computer vision, a bag of visual words is a vector of occurrence counts of a vocabulary of local image features. So we extract the key-points by detecting it from the images using algorithms like SIFT and SURF. Using this key-points, we find out the feature detectors which is the data point in the feature space. Then we use the Clustering method to extract a useful pattern of the descriptors and to use them for different images.

In this project, all the previous year groups have defined their own bag of words and used them in the program. Currently, Matlab provides a In-Build function called 'Bag-of-Features' , which does the same work as the bag of words definition. We have used this to create our features bag.

3 Code and Project Framework

The Matlab code is divided into many parts according to their functions and definition. We use the main file called 'example_classifier' to run the code. We have been given the Image dataset as :

- TRAIN : image set for training the data.
- VAL : Validation data (suggested to use). The validation data may be used as additional training data in the program.
- TRAINVAL : This is the union of training and validation data set.

- TEST : This is the Test data set. The test set is not provided in the development kit, and this is will be used to by the challenge setters to check the accuracy of the model.

We are also given a folder called 'Annotations' which has .txt files for every image in the dataset. That '.txt' files contains size of every image and also if the object of the class is available or not in that image. So for the training image , the text file has classified the classes as a ground truth labels as -1, 0 and 1. So -1(Negative) is there, that means the image does not have that particular class. If it's 0(Neutral) , this means that the image is hard to classify the object. And if the value is 1(positive), it means that there is a match on the image for that particular class.

4 Strategy

The main to create first is the bag of words, which can be made by dividing the function into four things:

- feature detection
- feature descriptors
- clustering
- classification

So our main goal is to classify the object, and we are using the bag of words approach which is different than the deep learning approach but it is also quite accurate if the dataset is well defined. for getting the features, we will use different methods like SIFT, SURF and Histogram to generate it from individual image. Then, all the feature vectors of all the images will be clustered using k-means or SVM to generate the segregate detection model. Then we will plot the ROC curve and compute the AUC, this will continue for all the classes in the system.

5 Implementation

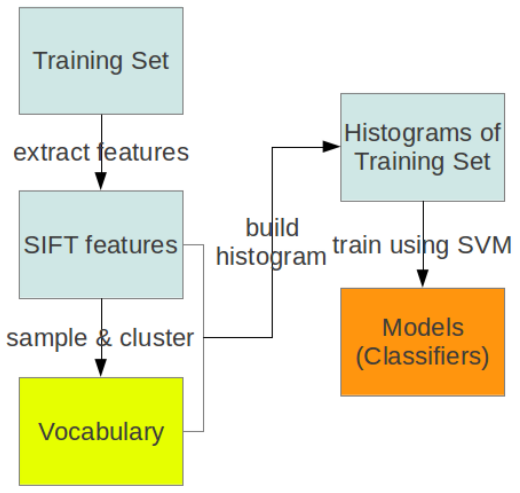


Figure 1: FlowChart of Project *(Source : Google Images)*

Approach followed to finish this project was to take the road less taken but once done will be easy to understand and implement. In the new version of *MATLab* various functions are already incorporated taking into account the various development taking place around the globe. Image classification challenge is not so challenging in terms of implementation part because all the building blocks have been converted to a modular form in the most recent version of *MATLab* which is 2017*b*.

In the initial part of the implementation, focus was to first understand the coding style of the *example* file and convert it in such a way that already available modules of the MATLab can be used. Various steps involved could be as follows:

5.1 Set Up Image Category Sets

In this the image set in partitioned into training and test subsets. *VOCInit* is used to initialize the parameters required to access the image data set and the labels. A *for* loop is created for all the 10 classes whose name can be accessed by the *VOCopts.classessi*. Each class file is read using *textread* and the indexes are saved in variable *ids* and the labels are accessed via *classifier.gt*. This label is useful in dividing the set in various classes which acts as a input for the *bagoffeatures* module.

A mechanism has to devised to find a way for segregating each image into different class and prepare the imageset accordingly. Hence these ground truth labels were used to know if a particular object to be classified is available in the image or not which if found true $(classifier.gt(j) == 1)$

is kept in that subset. This is done for all the classes and one single image-set is created to be used further. They are also labelled for the convenience using the variable name *VOCopts.classessi*. So finally the set looks like 10 sets of images containing that particular object as the name mentioned.

Next task is to partition the subsets obtained in the above step. *Partition* function is used for this purpose or *splitEachLabel* which takes the same input. Trying both showed that the second option was not workable on the the system and the reason is yet to be explored.

5.2 Create Bag of Features

Feature descriptors are extracted forming *visual vocabulary or Bag of Features* from representative images of each category. It defines the visual words using $K - Means$ clustering method implemented in the *Statistics and Machine Learning Toolbox* on *trainingsets*. Various parameters associated with this bag can be illustrated as follows:

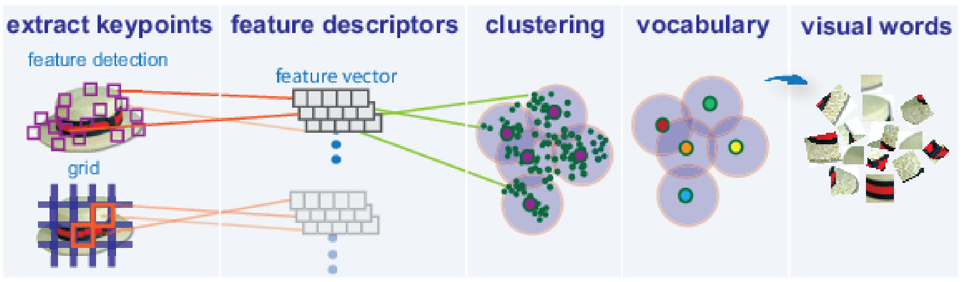


Figure 2: Bag of Features Flow diagram (source:MATlab)

- *Vocabulary Size*: It is the number of visual words, by default which is 500. Its range could be changed from 2 to infinity. It uses *CK Means Clustering*.
- *Strongest Features:* It is the fraction of each features used from each of the labels and it is by default value 0.8 (max value 1).
- *Verbose*: It is enabled **TRUE** shows all the processing taking place.

The algorithm workflow can be visualized as in Fig. 2. It requires images to be properly labelled and relies on detection without localization.

5.3 Train Image classifier with Bag of Features

Image classifier is implemented using *trainImageCategoryClasssifier* function of MATlab. It trains using binary support vector machine with error correcting output codes framework. Input given to this function is the bag of visual words obtained from *bagOfFeatures* and encode the imageset into histogram of visual words which are further used as positive and negative examples for training classifier.

Step 1 : Function *encode* is used to encode the features. Using the K nearest mean algorithm it forms histogram of features whose length is equal to number of visual words which are reconstructed (Fig. 3).

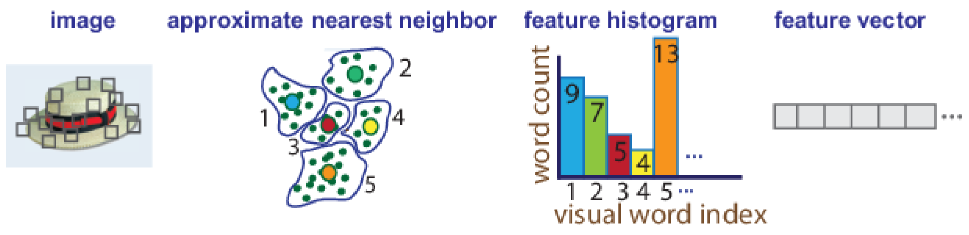


Figure 3: Encoder Flow diagram (source:MATlab)

Step 2 : Repeat the above step for each class of items to be classified Fig 4.

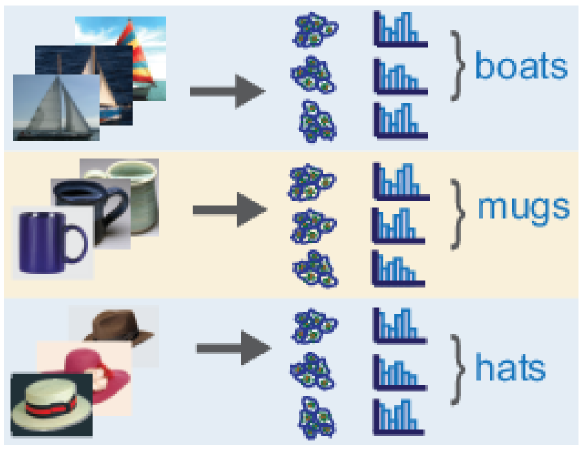


Figure 4: Classification Flow diagram (source:MATlab)

Step 3 : To evaluate the quality of the classifier, we compute the *confusion matrix* which gives an idea about how many of the classes are classified into which group and in a way let the user know about the accuracy of the classifier. It is a normalized matrix having 1 on the diagonal element in case of a perfect classifier otherwise fractional values giving overview of analysis of the prediction (Fig. 5).

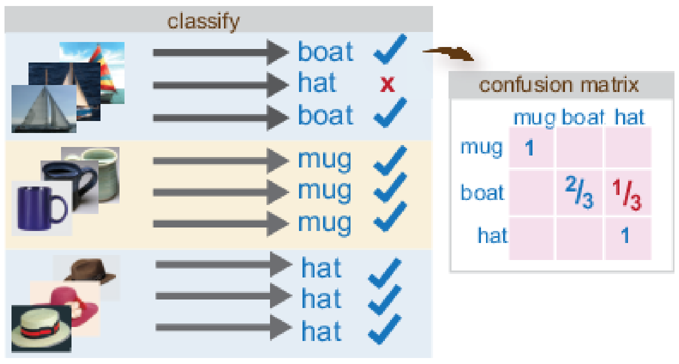


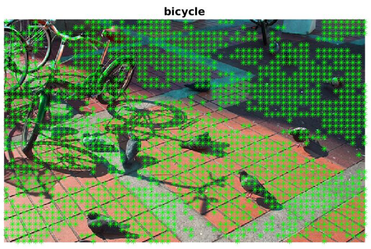
Figure 5: Classification Result (source:MATlab)

6 Analysis

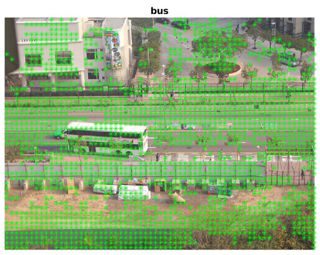
We did quite a bit of analysis by changing the values and parameters in bag-of-features , dataset size etc. by creating histograms, features representation in different cases and how a little change in parameters affects the histogram bins. Here is a random image of each class of the dataset :

Figure 6: Random Images of Set

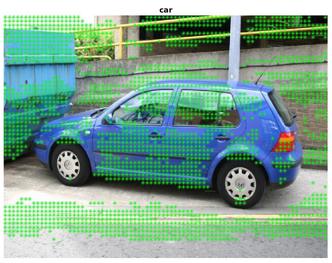
As you can see in the image set that there are some images having 2 class in the single image. That is case for some training and validation images, so we need to take that also into consideration while making the code for classification of images. The next step was creating the *bagoffeatures* , in which we had to modify the program according to the use of the command. So the first step is to extract the features of the image, for every class images. For that we used different size of vocabulary to test how the most common features are being extracted from the image set of each particular class.



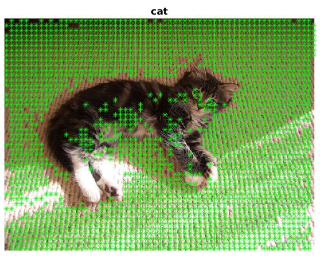
(a) Bicycle - 100 Vocabulary Size



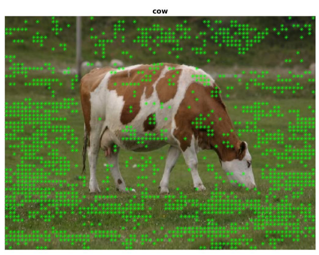
(b) Bus - 100 Vocabulary Size



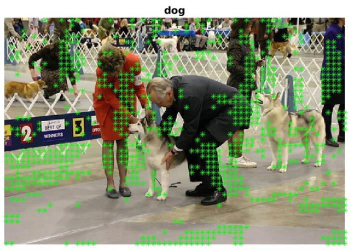
(a) Car - 200 Vocabulary Size



(b) Cat - 200 Vocabulary Size



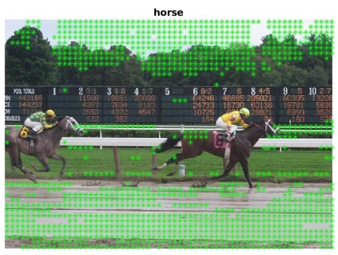
(a) Cow - 300 Vocabulary Size



(b) Dog - 300 Vocabulary Size

In the figure set of different vocabulary size of every class you can see a lot of green star type symbols, these are the most occurring words in the set of images of the same class. It basically represents that for each particular class these are the most occurring words in the image set of the same class. As the size of the bag of words vocabulary changes, so does the location of the most occurring words for each class.

As we can see in the figure of Cow Class that with different size of vocabulary , the position of the most occurring words also changes. We tested that for 5 different vocabulary size starting from 100 till 500 size. This set of images for each class is available in the folder along with the

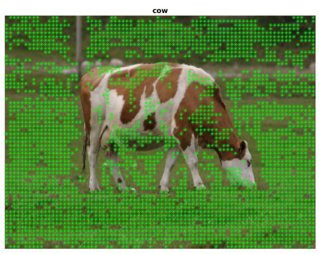


(a) Horse - 400 Vocabulary Size

(b) MotorBike - 400 Vocabulary Size

(a) Person - 500 Vocabulary Size

(b) Sheep - 500 Vocabulary Size



(a) Cow - 100 Size

(b) Cow - 200 Size

report. We have tested this for 100, 200, 300, 400 and 500. Then we decided to change the vocabulary size and see how the feature extracted changes the accuracy and set of features obtained. We also did this experimentation on 100, 200, 300, 400, and 500 size of vocabulary. If we look at the histogram bars available for each class by changing the vocabulary size , we will see that the features obtained are way different than the case of different size.

Size of Vocab	100	200	300	400	500
Accuracy(in percentage)	74.8	83.7	87.8	87.7	89.1
Time (in s)	308	267	179	339	416

Table 1: Mean Accuracy with different Vocab Size

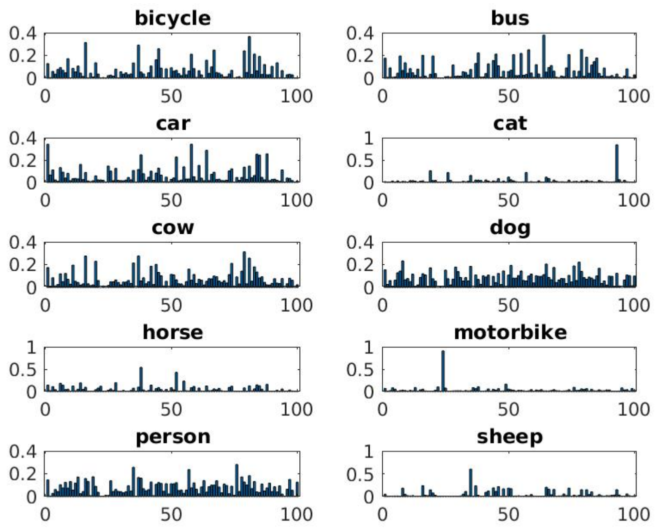


Figure 13: Vocab Size 100 - Feature Vectors

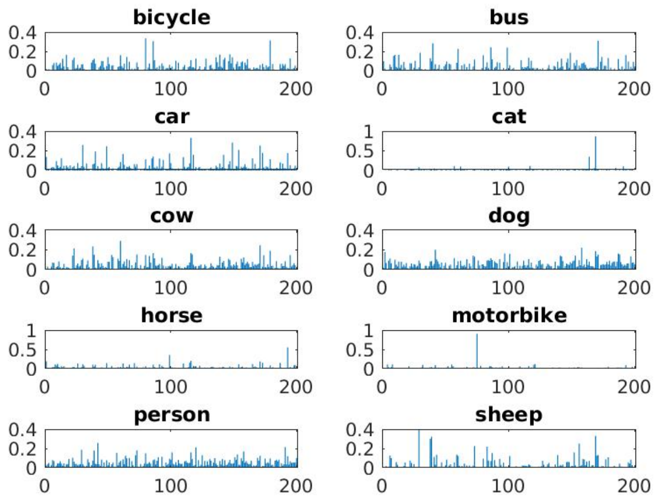


Figure 14: Vocab Size 200 - Feature Vectors

In table 1 , we have analyzed the data for different vocabulary sizes. As we can see in the table we have taken 5 different sizes starting from 100 till 500 and we saved time and mean accuracy in a notepad file. The mean accuracy for all the classes for 100 words comes about 74.8 which is the lowest among other sizes. But as we increase the size of the vocabulary to

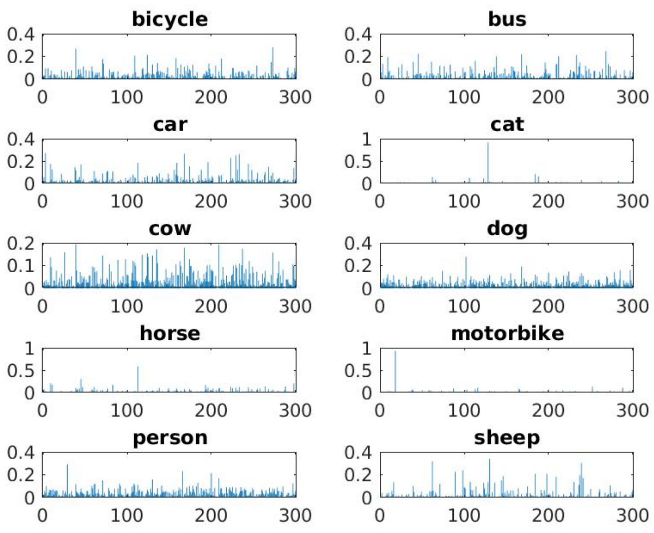


Figure 15: Vocab Size 300 - Feature Vectors

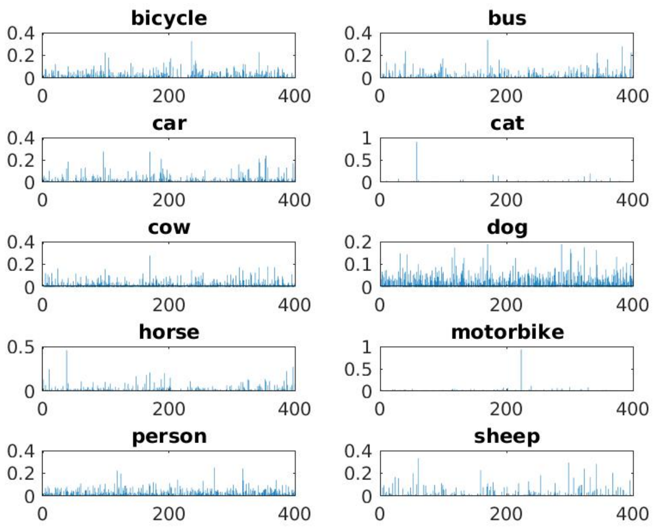


Figure 16: Vocab Size 400 - Feature Vectors

200 the accuracy increases about 10 and reaches 83.7 percentage. Then the accuracy doesn't rise in a big jump after 300 size bag, and it reached the maximum of 89.1 according to the data provided by the confusion matrix generated. Then we also took notice of the time taken for creating the bag.

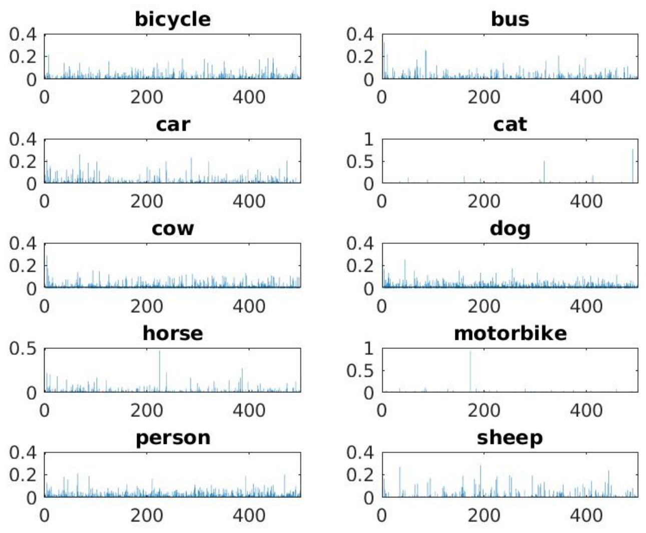


Figure 17: Vocab Size 500 - Feature Vectors

The time taken for creating the vocabulary for 100 words was around 310 sec, but what was interesting to know that the time taken by 200 and 300 size of words was less than it. The time taken by 300 size Vocab was around 180sec which is the least among all the 5 different sizes. It must be due to the algorithm inside the function. We also tried to create a Profile for the same to understand it but it was not enough to understand the main reason for it. So we can say the maximum time taken for creating a vocabulary was by 500 size bag and it was around 420 sec.

From table 2 to table 11, we have mentioned all the accuracy of each class separately with different sizes of bag. This is what we got from confusion matrix plotted in the notepad file. Basically s confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. This is the result we got by checking the result with the validation set, but if we check with other image set, we might get a different result. As we can see in the table 10, which is the result of the class person, we can see that the accuracy is very less and it doesn't increase with change of vocabulary. Also , we didn't want to over fit the data, so we had reshuffled the images before making the vocabulary again for different sizes. These all results are made with partition of data by 70-30 ratio, 70 being training data. By checking the results we can say that, in the validation set when the algorithm was testing with the images, it was able to perfectly classify the class with this algorithm. That is what all these accuracy represent in the tables. Also, if the data set is good and the testing images are also according to the regulation, the results will be nearby the validation accuracy.

Table 2: Accuracy with different Vocab Size

	Size of Vocab				
Class Name	100	200	300	400	500
Bicycle	85.7	92.8	92.8	92.8	100
Bus	90.9	90.9	90.9	100	100
Car	58.06	74.2	77.4	80.6	80.6
Cat	81.8	96.36	86.36	81.8	86.3
Cow	81.8	90.9	100	100	100
Dog	66.6	90.4	90.4	85.7	85.7
Horse	73.3	80	100	80	93.33
MotorBike	84.6	100	100	100	100
Person	33.3	47.2	55.5	63.8	52.77
Sheep	92.3	84.6	84.6	92.3	92.3

After gathering data and testing out different size of vocabulary , we decided to change the parameters of partition data. We wanted to analyze data when we had different features set due to change in the number of images and also the validation image changes accordingly. We calculated the mean accuracy for all the classes and the average time taken for the same. We started with creating a partition of 60% as initial value for the training set, and gradually increased the value by 5% till we reached the value of 85%.

For carrying out this analysis, we fixed the size of vocabulary size to 500 and the strongest feature size to 0.8 threshold. As we can see in the figure 18, the feature vector set which is available as the bins of histogram. We can say that, the features extracted for Cow, Dog and Person class are relatively very high compared to the other classes. We can notice that the class Motorbike and Cat features are very less for this partition of 60%. We don't get any particular set of features for cat and motorbike even after we have a partition of 85%. We can say that the images don't go well with this function we are using or the image set doesn't have that particular images, which goes with our feature vector function. As we have used SURF for getting the features, it might be due to the less change in gradient around the edge parts, that's why we might not get a major change in histogram of features for these particular classes.

In table 3, we have showed analysis of accuracy change when we change the size of the partition. Here by partition means change in training to validation image ratio. So if we have 100 images and partition size is 60% , then the images will be divided into two parts in random. Out of 100 , 60 images will be used for training the system or network , while the other 40

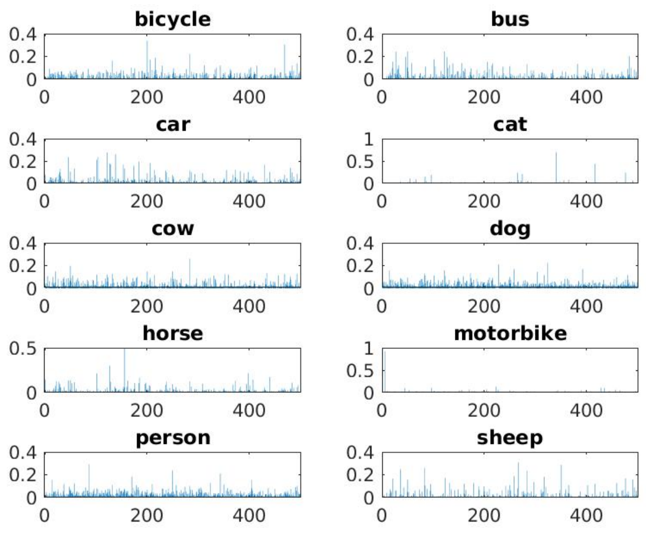


Figure 18: Partition 60% - Feature Vectors

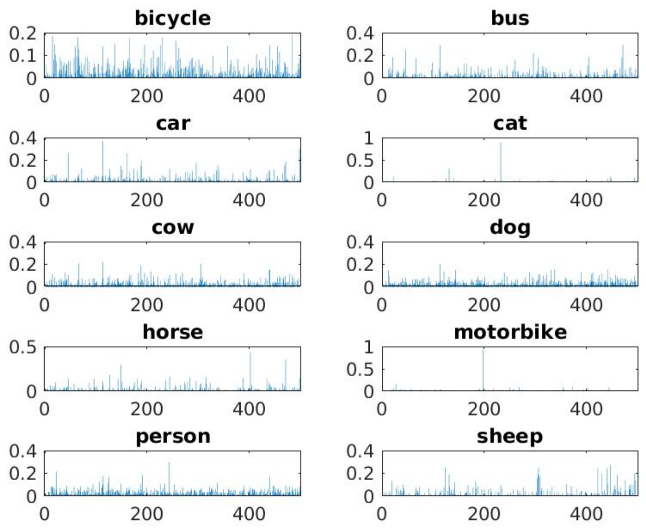


Figure 19: Partition 65% - Feature Vectors

will be used for cross validation with the output of this training data before it can be used for testing purpose. By doing this, we can check if the network is suitable for further use or needs to be trained again with different parameters and arguments. This table 3, is done with 266 classes images

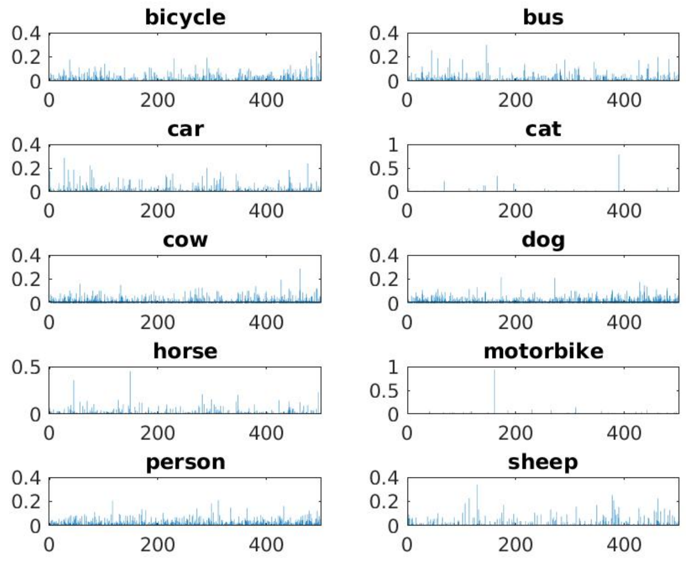


Figure 20: Partition 70% - Feature Vectors

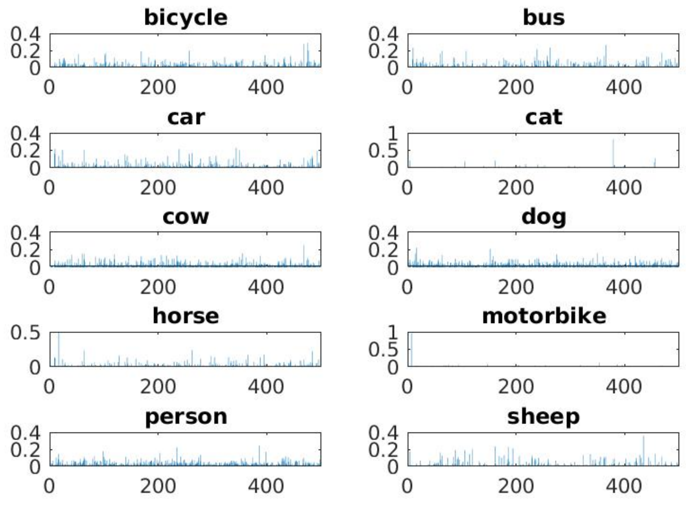


Figure 21: Partition 75% - Feature Vectors

in total. The accuracy was highest noted when the partition size was of 70-30%. We note that the accuracy is nearly similar when the partition range is from 60-75%, but then it drops by some 5%.

Finally we decided to change the value of the function called 'Strongest

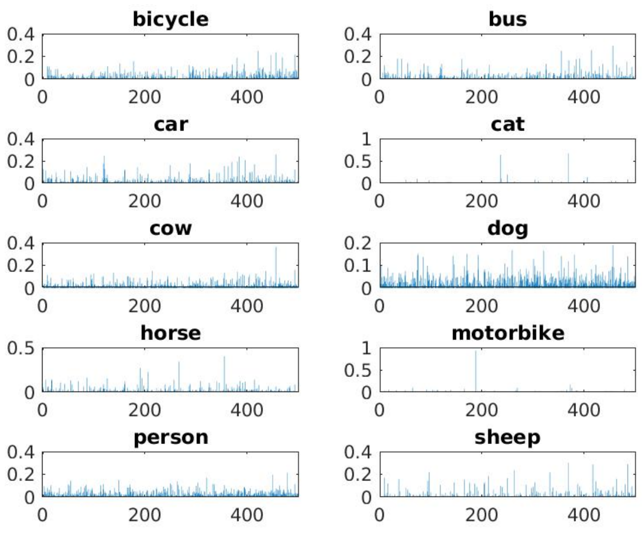


Figure 22: Partition 80% - Feature Vectors

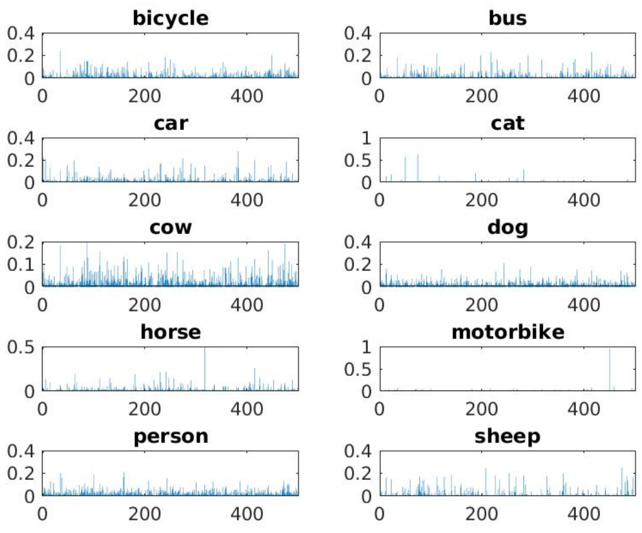


Figure 23: Partition 85% - Feature Vectors

Feature'. That parameter shows the fraction of strongest features, specified as the comma-separated pair consisting of Features and a value in the range [0,1]. The value represents the strongest of strongest features to use from each label contained in the image set input.

Table 3: Accuracy with different Parition Size

	Size of Partition					
Partition Size	60%	65%	70%	75%	80%	85%
Mean Accuracy	89.8	88.8	90.3	88.7	84.9	85.4
Time (in sec)	190	190	113	166	223	209

Table 4: Accuracy with different Strongest Features Size

	Value of Strongest Feature					
Feature value	60%	65%	70%	75%	80%	85%
Mean Accuracy	89.3	87.8	86.4	90.4	86.6	92.3
Time (in sec)	148	223	178	260	180	308

7 Difficulties Faced

This was altogether a new approach to be followed and it became a challenging task. All the files are to be read and converted in a way *MATLab* requires. It incorporates in itself all the study of functions *MATlab* has in itself as well the *example_classifier.m* files provided in the project related documents. Once this part is thoroughly done, a new file is to be created incorporating all the readings. Most of the time taken was to make an example program to run the full pipeline in a way Module available in MATlab needs and another test dummy file provided in the project. After this each variable are individually checked how they have their outcome and in what data type. Then utilizing them either directly or converting into the given form and appending them in the new code. Following this strategy decreased the length of the program drastically and made it convenient to do any changes required for thorough analysis.

8 Conclusion

The result of all the analysis shows that as the size of vocabulary increases, the mean accuracy was increases, but in return the time taken for creating the bag also increases by 100 sec. This concludes that as the number of feature increases in the bag, we get better results. By changing the partition of the training and validation size, it is concluded that accuracy remains nearly same, for the same size of the bag, but the time taken to train it increases as the partition size increases. When the strongest feature parameters were being optimized, the conclusion can be made that by changing the values the difference between the final accuracy is not very diverse.

Thus, by generating all these results, the optimal suggestions will be to use Bag size of 500 vocabulary with partition of 70-30% ratio and strongest feature value being 0.7. By keeping these parameters, it is possible to generate the best result for this dataset.

9 Approaches Followed -Part II of Report

In Part I of the project there were quiet a few things which were required to be done. First of all, train, validation and testing data has to be taken from the file given. As the aim of the first section was defined to focus on bag of words instead anything else, we left the rest for the second part of the project.

From the first section, various parameters were set as *vocabulary size* as 500 and *Strongest Features* as 0.8 in the *bagofFeatures* function. Now moving ahead, unlike dividing the training data itself into the training and validation purpose, separate validation data-set is used.

9.1 Trial 1

First three different types of set is created namely, *Training set, Validation set & Testing set*. In the first iteration training of the SVM is done using the testing test and the evaluation matrix is computed whose output is used for generating ROC Curve.

$$categoryClassifier = trainImageCategoryClassifier(trainingSets, bag)$$

$$confMatrix = evaluate(categoryClassifier, validationSets);$$

Mean of the results are computed as:

$$s = mean(diag(confMatrix))$$

Later the classifier is applied on Testing Set:

$$confMatrix = evaluate(categoryClassifier, testingSets);$$

$$s = mean(diag(confMatrix))$$

9.2 Trial 2

In this part, various hit and trials were done tweaking parameters of bagofFeatures command from MATLab like, *VocabularySize, StrongestFeatures*. But this did not bring any visible changes in the performance of the algorithm.

9.3 Trial 3

Here, another way of defining the SVM classifier was defined. In this, templateSVM from MATLab was used to define the SVM and look for the possible changes in the confusion Matrix. Various parameters were tweaked to see the changes in the result, few of which are mentioned below.

Kernal Function: Various functions tried are, Gaussian, linear and polynomial.

Leaveout: This is leaveout cross validation flag by default which is off. It reserves the observation as validation set and train model using another n-1 observation.

Polynomial order: Order of polynomial used in the kernal function.

OutlierFraction: Which is the flag for expected proportions of ouliers in training data.

10 Results from Traditional Approach

In creating templateSVM first parameters polynomial order was changed. It was varied from polynomial order of 3,10 and 50. Result of validation set showed as considerable change in overall performance increasing from 80% to 85% while on the other hand for the testing set, it did not show considerable change and increased from 40% to 42%.

All three types of classification method in Kernal function was used in this like linear, Gaussian and Polynomial but polynomial was the best suited for the purpose hence the next step became to try changing its order from 3,5,10,15 till 50. Next parameter which was taken into account is Solver which is optimization routine. In this, first *L1QP* was used which implements L1 soft margin minimization by quadratic programming. It gave the best performance when compared to other solvers like Sequential Minimal Optimization (SMO) and Iterative Single Data Algorithm (ISDA). Standardization of the data contained in the columns of predictors is not done automatically, so this is done via changing its default value to *TRUE*.

In the *Leaveout* part, some of the variations tried are, computing model using *CVPartition, Holdout, Kfold*. This also did not made any considerable change in the performance of the model. As we already have adequate amount of training and validation set. One more thing tried in this was to jointly combine the training and validation set then apply this cross validation partition and checks for the average performance. *KFold* randomly partition the data into k sets. For each set, one of them is kept as validation data while training is done using other $k-1$ sets. In *Holdout*, fraction of the data is prescribed as the validation set. For all these trials, *CrossVal* has to be kept *ON*.

Parameter *OutlierFractions* was kept as 0, .01 and .001 this also did not make any change noticeable in confusion matrix. Finally the results are shown in Fig. 24.

Table 5: Accuracy with L1PQ order 50

Class Name	Validation Set	Test Set
Bicycle	65.2	65.1
Bus	38.4	42.9
Car	67.39	61.8
Cat	74.19	51.6
Cow	41.17	59.4
Dog	17.85	18.6
Horse	15.8	29.3
MotorBike	15.8	18.2
Person	10.7	10
Sheep	33.3	64.1
Average	38	42.02

Table 6: Accuracy with Validation Set

Class Name	Training Set	Validation Set	Testing Set
Bicycle	20	23	43
Bus	15	13	28
Car	44	46	89
Cat	31	31	62
Cow	16	17	32
Dog	30	28	59
Horse	21	19	41
MotorBike	19	19	38
Person	52	56	110
Sheep	18	21	39

11 Neural Network Approach

A Convolutional Neural Network as known as CNN is a machine learning technique used in the field of deep learning. This network is derived from large set of trained images. As there is such a large collection of images, it learns from all the features extracted from it and makes a good CNN network. They are much better in performance compared to traditional techniques used till now. One of the best features of CNN is that is can be pretrained to have a feature set, that can be used for any different programs.

For the project, we used pre-trained AlexNet model for extracting the features. AlexNet is a pretrained Convolutions Neural Network 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 classifier trained using CNN features provides accuracy more than 90%, which is higher than the accuracy achieved using bag of features. Here instead of using extracting the bag using bagoffeatures command, we have pretrained network, that already has a bag.

11.1 Implementation

As the framework is already provided, only the implementation of the network has to be done to extract the features. Till now, we are extracting the features using the traditional methods, but now we will create a bag of features using a Alexnet network. Each layer in Alexnet CNN produces results of activation's with an input image. We don't require all the layers for training the image , and for that we are using 6th,7th or 8th layers. The beginning layers of the net are used for obtaining the edges and sides of the images, so we have to go into the deeper network layers to extract all the features of the image. Thus it is recommended to extract from the last 3 layers. Then all these features can be used for classification purpose. Below is the overall flow of the program :

- The first task is to load the net, we have used Alexnet in this case for testing purpose. As the pre-trained net is loaded and assigned, we take all the training images and then give it to the CNN network to extract the features. For extracting the features we first make it into an RGB image as that is required input for the net and also resize them to 227X227 size. After the images are converted to this particular requirement we then can extract the features directly, or we can add the activation layers 6th,7th and 8th for training purpose.

- After extracting the features, kernal SVM was used to classify the test images. This is a function of Vl_feat used to classify the test images. For this, the feature are saved using a function called Vl_trainSVM and it are saved into the 'local' folder. These features are then loaded in the testSVM function and then are tested using the vl_svmdataset function of the vlfeat. The svm classifier will class with the method of 'KChi' (Chi squared method) with a period of 3. The parameters that have been defined in the vlfeat functions are lambda and the kernal size. With the change in lamba the training rate changes and we have fixed kernal size of 3.

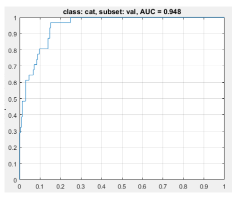
11.2 Setting the labels from Alexnet

Imagenet is divided into 1000 class problem, and for that there are many labels given among each class. For the given images with the help of alexnet it was possible to get label of each image. To perform that , all the image ids were read and then were called in order. A new text file was created to save the list of the labels in order. First, all the images were converted to the size of 277X277 , but before re-sizing it has to be made sure that the images are in RGB format. Then, we use the command called classify and pass the parameter of net along with the image ids. Here the net used is Alexnet, but we can define whichever architecture required by the user to classify. All the images will be labeled in the file called as final_result. Now as all the label results are available , they can be classified into their class according to the labels. To do that, all the labels have to classified according to the words in the labels first and then put in each definite class of the given word. That class will be compared to the list given in the imagenet and the class can be determined. The link regarding that is given as imagenet issue in the reference.

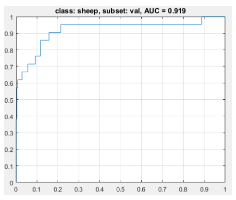
11.3 Results

In this section , the result of the neural network approach is presented. Before, in the traditional approach , the average results obtained was around 86% when done with normal approach. But it is noticed that with the AlexNet approach , the results have improved around 10% overall. The training of the classifier has used the images in both the train set including the validation set, so 256 images in total. The bag of words process uses descriptors from only the training image set. The classification testing has been done on the test set which comprises of 523 images in total.

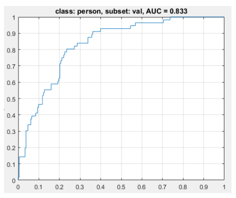
- When we extract the features from 6th layer of the AlexNet , we get this results as shown in figure 25 and 26. Accuracy obtained from this net activation when passed into the classifier is 0.936 as an average for all the 10 classes combined. The time taken is around 14 min to extract the features from the layer on a i3 computer with 4GB ram.



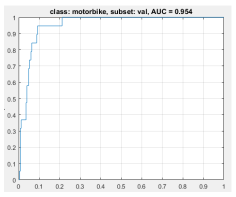
(a) Cat



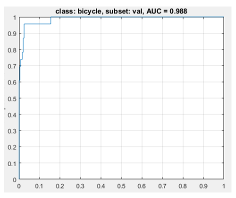
(b) Sheep



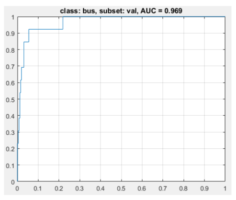
(c) Person



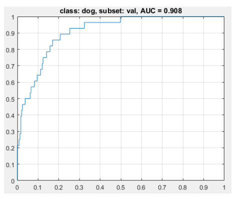
(d) Motorbike



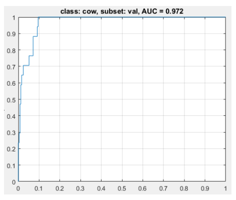
(e) Bicycle



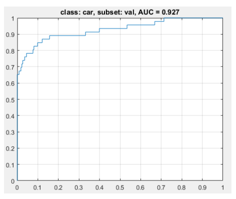
(f) Bus



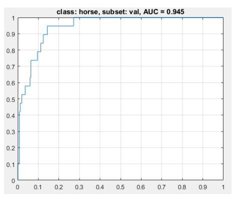
(g) dog



(h) cow

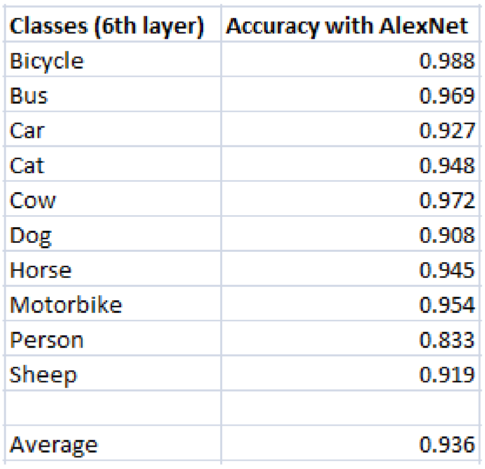


(i) Car



(j) Horse

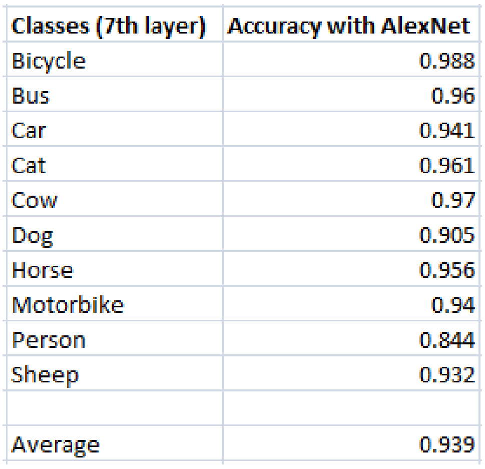
Figure 25: ROC curve AlexNet



Classes (6th layer)	Accuracy with AlexNet
Bicycle	0.988
Bus	0.969
Car	0.927
Cat	0.948
Cow	0.972
Dog	0.908
Horse	0.945
Motorbike	0.954
Person	0.833
Sheep	0.919
Average	0.936

Figure 26: Result - 6th Layer

- Below in the figure 27 and 28 are the results and accuracy obtained when we extract the features from 7th layer of the AlexNet. As we observe when the features are extracted from the 7th layer, the accuracy is 0.939 , which is more than the 6th layer. The time taken is around 20 min to extract the features from the layer on a i3 computer when the model is trained for the first time. When the features have been extracted and then used than the time taken is around 3 min to create the ROC.



Classes (7th layer)	Accuracy with AlexNet
Bicycle	0.988
Bus	0.96
Car	0.941
Cat	0.961
Cow	0.97
Dog	0.905
Horse	0.956
Motorbike	0.94
Person	0.844
Sheep	0.932
Average	0.939

Figure 28: Result - 7th Layer

12 Conclusion

Traditional approach was utilizing Bag of Words to extract features and then transfer the same to SVM classifier. In the first part of the report, small set of data was used for the testing and validation purpose. Training data was divided for both the purpose hence results reported had high accuracy. But in this part different set of data was used for both training and validation purpose as provided to us. Table. 6 shows the number of elements from each class present in the training, testing and validation purpose. Analysis was done using different techniques like changing the parameters of bagofFeatures or creating a SVM classifier or using the MATLab defined techniques. But all these did not bring remarkable change in the performance of the algorithm as desired. Others techniques where the extracted features were given as input to the SVM classifier has a far greater performance wrt the one obtained this way. We would recommend not to use this inbuilt function for the classification purpose as the results were not upto the marks and could mislead.

The result when the neural network is used, increases the average accuracy by around 10%. It can be said that when the neural network to extract the features the results are better than the traditional techniques. If different net are used in it, then the results might change and even provide a better accuracy. The training time to extract features from the training set is increased from the traditional methods. So if the image set is of more than 10000 images the time taken will increase exponentially and it is a case of trade off between accuracy and time taken. Thus, it can be said that with the current dataset it is better to use neural networks than traditional methods to solve the given problem.

References

[1] Wikipedia,
`https://en.wikipedia.org/wiki/Bag-of-words_model_in_computer_vision`

[2] MATLab Central,
`https://es.mathworks.com/help/vision/ug/image-classification-with-bag-of-visual-words.html`

[3] MATLab Central Functional Help,
`https://es.mathworks.com/help/vision/ref/bagoffeatures-class.html`

[4] MATLab Central Example Implementation,
`https://es.mathworks.com/help/vision/examples/image-category-classification-using-bag-of-features.html`

[5] MATLab Central,
`https://es.mathworks.com/help/stats/classification partitionedmodel-class.html`

[6] MATLab Classification using SVM,
`https://es.mathworks.com/help/stats/classificationsvm.crossval.html`

[7] Template SVM Arguments,
`https://es.mathworks.com/help/stats/templatesvm.html #namevaluepairarguments`

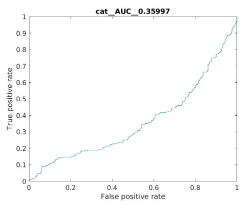
[8] Template SVM,
`https://es.mathworks.com/help/stats/templatesvm.html`

[9] VLFeat Pretrained Network,
`http://www.vlfeat.org/matconvnet/quick/`

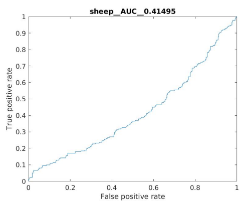
[10] VLFeat Train-Test function for SVM,
`http://www.vlfeat.org/matlab/vl_svmtrain.html`

[11] ImageNet class defination,
`https://datascience.stackexchange.com/questions/27694/is-there-a-person-class-in-imagenet-are-there-any-classes-related-to-humans?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa`

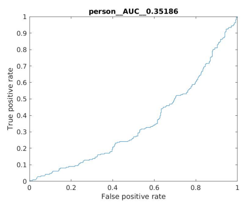
[12] GitHub repository,
`https://github.com/rbgirshick/rcnn`



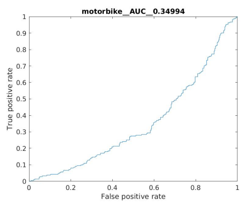
(a) Cat



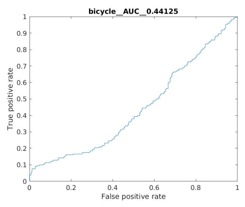
(b) Sheep



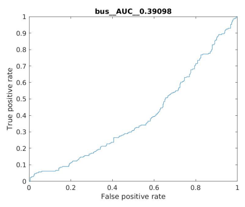
(c) Person



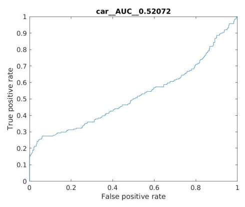
(d) Motorbike



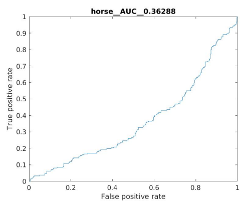
(e) Bicycle



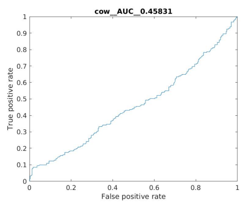
(f) Bus



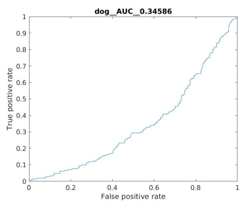
(g) Car



(h) Horse

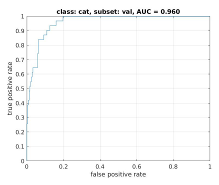


(i) Cow

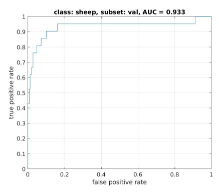


(j) Dog

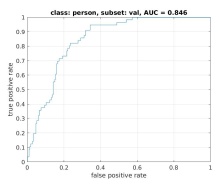
Figure 24: ROC curve L1QP with order 50



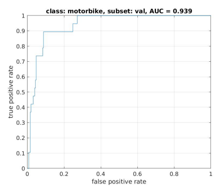
(a) Cat



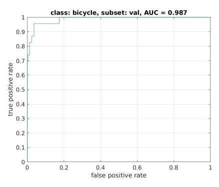
(b) Sheep



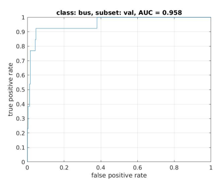
(c) Person



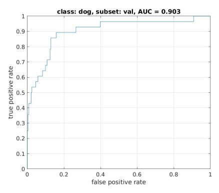
(d) Motorbike



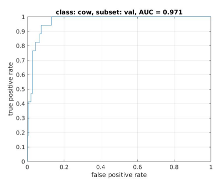
(e) Bicycle



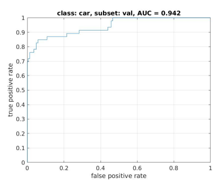
(f) Bus



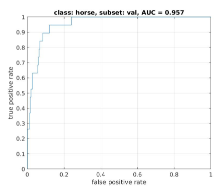
(g) dog



(h) cow



(i) Car



(j) Horse

Figure 27: ROC curve AlexNet