# Object Detection and Localization using Bag of Words Representation and Multi-Class Classification

Amitesh Sah
Electrical and Computer Engineering
NY, USA
aks629@nyu.edu

Neel Dey
Electrical and Computer Engineering
NY, USA
nsd291@nyu.edu

Shashank Garg
Electrical and Computer Engineering
NY, USA
sg4437@nyu.edu

***Abstract*— The project proposes using a Machine Learning based approach to the Computer Vision challenge of detecting and localizing objects within an image and categorizing the objects found in that image using a multi-class classification approach. A Bag of Visual Words approach which usually ignores spatial information in an image is adopted for the initial classification, and later extended for the localization. This also includes intermediate steps such as Feature Extraction using Patch Descriptors such as the Scale Invariant Feature Transform, Clustering to generate the Codeword Dictionary Generation, training the classifier and discriminative classification for the object classification and Hierarchical Pyramid Segmentation for object localization.**

**Keywords—Object Detection, Object Localization, Object Classification, Multi-class classification, Computer Vision**

## I. Introduction

The project addresses the classic problem in the field of Computer Vision of Object Classification, and Localization within an image. Object classification involves the classification of images based on their visual content. A simple example being image tagging. It differentiates between images having different objects. For example, when Figure 1(a) is fed to an object detector classifier, it concludes that there is a cat in the image based on its training with various categories, one of which would be this cat. Localization involves finding the object within the main image. The red bounding box in Figure 1(b) shows the location of cat in the image. Object Localization in our context will mean to determine the number of objects and their location in an image.

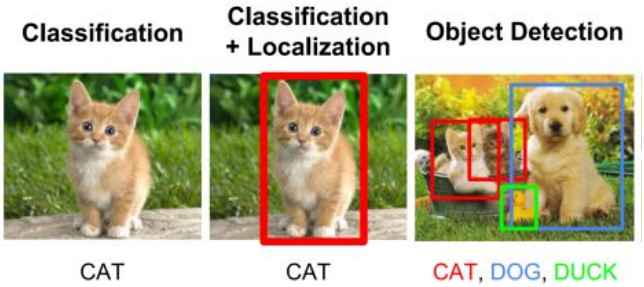


*Figure 1: Introduction of Image Classification and Localization*

## II. Motivation

With an absolute explosion in the amount of images generated within the past decade, the task of image tagging is more important than ever with millions of dollars in research funding coming in from the Government and Industry titans like Facebook, and Google. Computer Vision and Machine Learning have have a revolution underway due to the availability of open source data, and large computing clusters.

Image classification is currently used in various industries like facial recognition for security or in the realm of social media, the military for aerial monitoring, image search applications, etc.

One approach is to train a neural network classifier for the classification and localization, however this requires enormous amounts of training data and large scale computation. We have implemented a Bag of Words model for image features instead due to their relative ease of training and good accuracy, and extended the overall Bag of Words model to deal with the problem of localization. Coupling powerful approaches within Computer Vision with Machine Learning

lead to powerful algorithms that help serve the above applications better.

## III. Related Work


[1]. G. Dorko and C. Schmid. Object class recognition using discriminative local features. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004.

[2]. L. Fei-Fei, A. Torralba, and R. Fergus, A tutorial on Recognizing and Learning Object Categories, 2007.

[3]. S. Lazebnik, C. Schmid, and J. Ponce, Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories, CVPR 2006.

[4] C.H. Lampert, M.B. Blaschko, T. Hofmann, Efficient Subwindow Search: A Branch and Bound Framework for Object Localization, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009.

[5]. D. G. Lowe, Object Recognition from Local Scale Invariant Features, ICCV 1999.

[6]. D. Schmitt, N. McCoy, Object Classification and Localization using SURF Descriptors, 2011.

[7]. Zhang, Jian, Marszalek, Marcin, Lazebnik, Svetlana, and Schmid, Cordelia. Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. IJCV, 73(2):213–238, June 2007.

[8] Breiman, L. "Random Forests." *Machine Learning*. Vol. 45, pp. 5–32, 2001.


## IV. Data

We have used the Caltech101 and Caltech 256 datasets. In Caltech 101, there are objects belonging to 101 categories and each category has about 31 to 800 images, hence the total number of images amount to 8677. In CalTech 256, there are object belonging to 256 categories and each category has about 80 to 827 images and the total images in this dataset is 29780.

| *Dataset* | *Released* | *Categories* | *Pictures* | *Pictures Per Category* | | | |
|---|---|---|---|---|---|---|---|
| | | | *Total* | *Min* | *Med* | *Mean* | *Max* |
| Caltech-101 | 2003 | 102 | 9144 | 31 | 59 | 90 | 800 |
| Caltech-256 | 2006 | 257 | 30608 | 80 | 100 | 119 | 827 |

*Table: Statistics of Caltech 101 and 256. The original release of the datasets included an extraneous class of images, which was removed, thus reducing the size from 102 to 101 classes.*

A portion of each dataset will be used for testing the performance of our algorithm. The data was split into 60% training images and 40% testing images randomly for each category. Hence, for Caltech101, the total training and test images are 5206 and 3471 respectively. Similarly for Caltech 256 dataset, the total training and test images are 17868 and 11912 respectively. Also, images belonging to 4 specific categories of data from Caltech 101 such as airplanes, car-sides, motorbikes, and faces were trained and tested, were focused on.

In our project, Caltech 256 was used more sparingly, due to the massive computational requirements described later on.

## V. Algorithms used

Our algorithm for object detection can be summarized as:

- Split the data into 60% training and 40% testing sets.
- Generate Feature Vectors for all of the images
- Train a classifier using the feature vectors from the training images
- Use the trained classifier to predict the labels of the testing data.

The following elaborates on the above algorithm which will be used to address the problems mentioned above.

### *1) Feature Extraction*

We are using the SIFT (Scale Invariant Feature Transformation) algorithm to generate interest points and features. The **Scale Invariant Feature Transform** (SIFT) was introduced by **David Lowe** in 1999, [5] and provided a robust solution to the problem of detecting key features invariant to image translation, scaling, and rotation, partially invariant to illumination changes and robust to local geometric distortion. The interest points generated by the algorithm are points in the space of 128 dimension and for each interest point we generate feature vectors of length 128. So for an image with N interest points we will have N X 128 dimensional vectors describing the different features of the image.

For the Caltech 101 dataset which has 8677 images, we generated 8677 feature vectors (with a concatenated length of 1896071). For Caltech 256 dataset which contains a much larger number of images with more categories we generated 29780 feature vectors from 29780 images (with a concatenated length of 15846496). Now, the next step would be to cluster the feature vectors to generate codewords. These codewords are in analogues to the approach of "bag of words" in natural language processing. But in our case the words would rather be visual words. The next section explains in detail about the codeword dictionary generation.

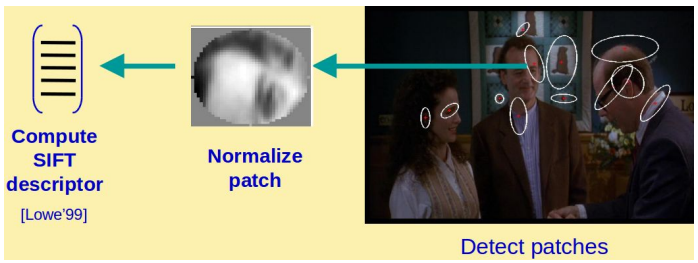


*Figure 2: Feature Extraction. Credit: Josef Sivic*

*2) Codeword Dictionary Generation*

Once we have multiple descriptors from the images as explained in the last section, we use K-means clustering to cluster our descriptors/codewords into groups to generate visual codewords [6], [7]. The K in the algorithm is the the number of clusters you want your data to be grouped into. In our case we ran the clustering algorithm for different k values and calculated the energy for each k.

For the smaller dataset, with only four categories from the Caltech 101 dataset (Airplanes, Faces, Motorbikes and Cars), the optimal number of clusters were very less as compared to while running for the whole dataset. This comes in accordance with the fact that with less categories we will have less less distinct codewords. A performance analysis on the data with different values of K has been shown below in the results section.

After clustering, we generate a histogram of visual words in each image. Which means that for K clusters we will have K codewords and we map each feature vector present in an image to the cluster it belongs and count the number of such features for every image. This generates a histogram of visual words which basically is a sparse matrix with number of rows equal to the number of images in the dataset and number of columns equal to the number of clusters, i.e. K. Figure 4 shows the generated histogram for the whole dataset (CalTech 101).

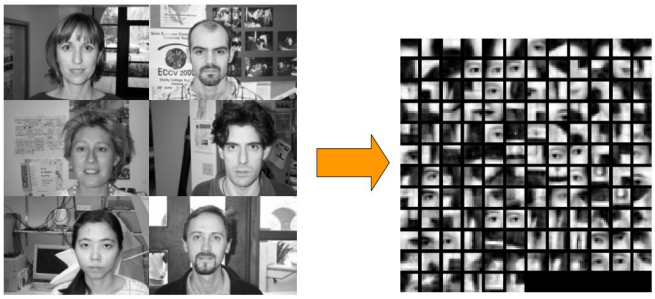


*Figure 3: Representation of Visual Word Extraction*

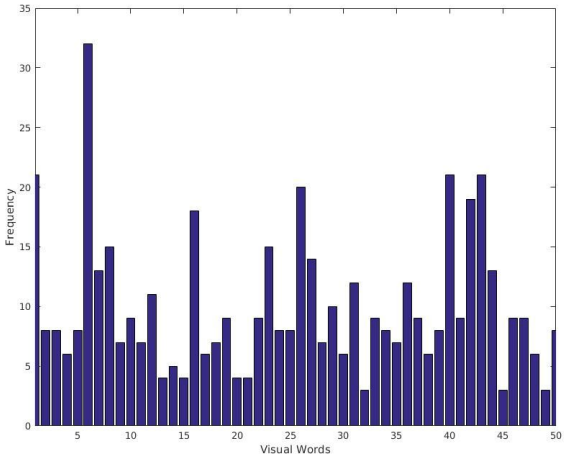


*Figure 4: Histogram/Codeword Dictionary Generation for 50 clusters*

*3) Training the Classifier and Discriminative Classification*

We consider each training image to be a vector which is the Histogram generated previously. We used Random Forests and Naive Bayes classifiers for the classification and compared the results.

*RANDOM FORESTS:*

Random forests is a subclass of decision trees which is a very popular model of classification in machine learning. [8] Each node in the tree is point where we classify and the split is generally a binary split of whether the subset of data being considered belongs or does not belong to the particular tag. We keep splitting the nodes till finally we arrive at a point where we have all true classified samples. Main benefits of these type of algorithms is that they are invariant under scaling and various other transformations of feature values and also is robust to inclusion of irrelevant features and produces inspectable models.

The traditional approach of decision trees however, has a problem that it can overfit the training sets as the tree gets deep. This basically happens because we stop the algorithm only when we have all true classes. This makes the model have highly irregular patterns and although the bias for the trained model will be very low the model on the other hand will have high variance which is highly undesired. To tackle this issue people came up with different version of classifiers based on the decision tree approach as described below.

Firstly, Bagging (Breiman, 1996) is one algorithm that fits many large trees to bootstrap-resampled versions of the training data, and classify by majority vote. Another algorithm is Boosting which basically fits many large or small trees to

the reweighted versions of the training data and classify by weighted majority vote. Random forests on the other hand is an extension of bagging approach. In general, Random forests is the most effective and widely used algorithm and gives better results than other decision tree algorithms.

As mentioned in the previous section, random forests is an extension of bagging, we will describe bagging first. Bagging averages a given procedure of tree fitting over many samples picked without replacement, to reduce its variance. The algorithm works as follows. Given a training set $X = \{x_1, ..., x_n\}$ with responses $Y = \{y_1, ..., y_n\}$, bagging repeatedly B times selects random samples with replacement of the training set and fits trees to these samples. After training, predictions for unseen samples $x'$ can be made by averaging the predictions from all the individual regression trees on $x'$. Although this removes correlation between the samples to a great extent, random forests go a step ahead to further reduce the correlation. To achieve that, rather than randomly taking samples from the training data, at each split, it randomly selects a subset of features. This is typically beneficial when one of the few features are very strong predictors for the response variable, then these features will also be selected in any of the B trees, causing them to become correlated. Random forests helps us solve this issue.

*NAIVE BAYES:*

We also applied Naive Bayes classifier to our data to compare the results with the random forests. Naive Bayes is a probabilistic method and can be described by the following equations. The probability $\phi_{y=c}$ that an image fits into a classification c is given by

$$\phi_{y=c} = \frac{1}{m} \sum_{i=1}^{m} 1\{y^i = c\}$$

Additionally, the probability $\phi_{k|y=c}$, that a certain cluster centroid, k, will contain a word count, xk, given that it is in classification c, is defined to be

$$\phi_{k|y=c} = \frac{(\sum_{i=1}^{m} 1\{y^{(i)} = c\} x_k^{(i)}) + 1}{(\sum_{i=1}^{m} 1\{y^{(i)} = c\} n_i) + N}$$

Based on this, the naive bayes method can be applied to our bag of word model.

### 4) *Hierarchical Segmentation*

This form of image classification would work well for images with one one prominent central object . However, that is rarely the case in real images which comprises of multiple objects. So, if the image contains a subset of many objects, this classification algorithm would fail. For this, we take a hierarchical segmentation approach [6].

We first extract the descriptors D from the image using SIFT. We then segment the image into several pyramid levels of our choosing L. The higher the levels, the better the granularity of localization search. At each level, the entire image descriptors are segmented. This develops a system where each pixel values votes for belonging to a certain category. The pixel from lower levels has more voting weights than the votes from a pixel at higher level. The weights are defined as $(\frac{1}{2})^{L-l}$. So the summed weighted votes for each class are mapped into N*1 map, where N is the different classes given by equation

$$voteMap(c) = \sum_{l=0}^{L-1} 2^{l-1} 1\{label_{pyr}(l,i) = c\}$$

At the end, the label given to the pixel is the class that has the maximum number of votes.

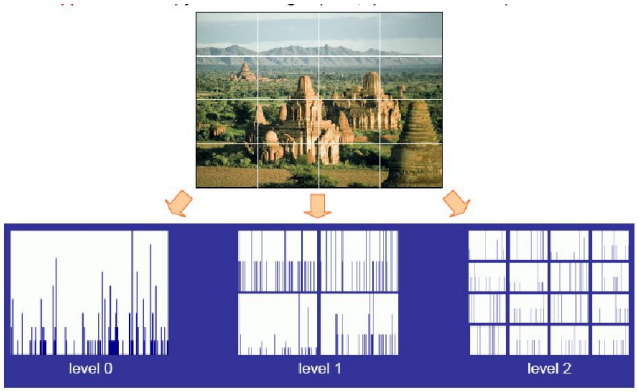


*Figure 5: Hierarchical Descriptor Segmentation. Credit: S. Lazebnik [3]*

## VI. RESULTS

| K | Random Forests | Naive Bayes |
|---|---|---|
| 20 | 91.08 | 86.11 |
| 50 | 89 | 87.85 |
| 100 | 86.68 | 61.57 |
| 500 | 81.82 | 51.97 |

*Figure 6: Percentage Accuracy of Classifier for 4 classes of interest, compiled against the size of the codeword set (K).*

When 4 classes are chosen with a high number of training images, our classifier has excellent accuracy, with Random Forests performing slightly better than Naive Bayes. However, with an increasing size of the codeword dictionary, a tendency to overfit emerges with the percentage accuracy decreasing.

Interestingly, Random Forests is somewhat stable against this, as shown in the data above, but Naive Bayes has a significant performance decrease.

| K | Random Forests | Naive Bayes |
|---|---|---|
| 20 | 31.88 | 25.15 |
| 50 | 31.94 | 27.32 |
| 100 | 28.66 | 25.52 |
| 500 | 21.67 | 4.42 |

*Figure 7: Percentage Accuracy of Classifier for 101 classes of interest, compiled against the size of the codeword set (K).*

This trend continues when the classifier is applied to the entire 101 dataset. As is obvious, the accuracy overall across all classes would be much lower due to some classes having very little data to train on. Naive Bayes again suffers greatly, reducing to just 4.42% at 500 clusters.

| Class | AUC |
|---|---|
| Faces | 0.9983 |
| Motorbikes | 0.99 |
| Airplanes | 0.9602 |
| Cars | 0.9447 |

*Figure 8: Area under the curve for 4 classes using Random Forests.*

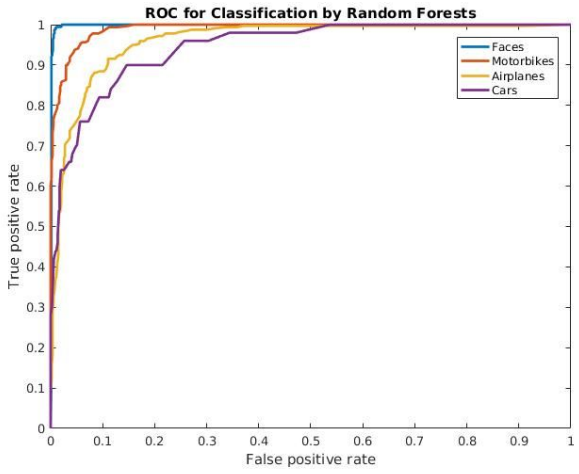


*Figure 9: ROC for 4 classes using Random Forests*

The ROC for the 4 classes of interest are about as accurate as can be expected, with excellent AUC values. This is always the case when there is a lot training data. The algorithm only suffers when the training set for a category becomes small.

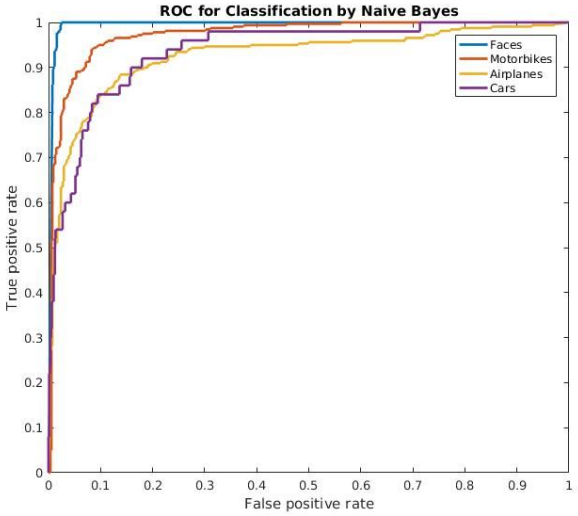


*Figure 10: ROC for 4 classes using Naive Bayes*

The Naive Bayes classifier has similar performance for these 4 classes. Interestingly, it actually does better for cars than Random Forests does.

| Metric | Faces | Motorbikes | Airplanes |
|---|---|---|---|
| Precision | 0.977 | 0.9281 | 0.9125 |
| Recall | 0.9714 | 0.9252 | 0.8134 |
| f1 scores | 0.9742 | 0.9267 | 0.8601 |

*Figure 11: Some metrics of classification for individual classes*

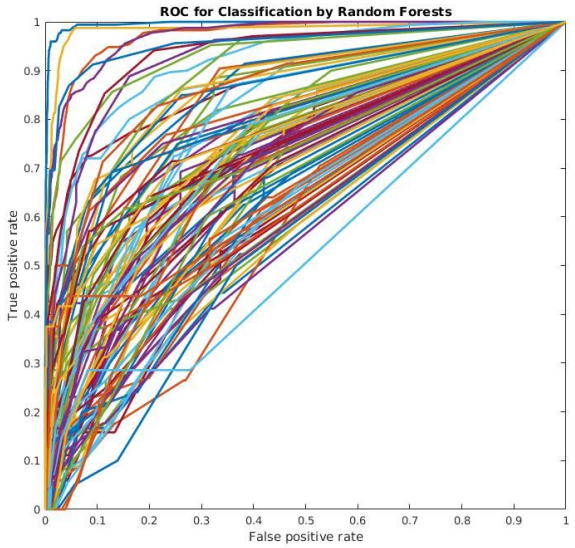


*Figure 12: ROC Curves for the entire dataset using Random Forests.*

As can be seen here, some classes do far better than others in their accuracy. This is again due to the size of the training sets.

| | | | |
|---|---|---|---|
| 171 | 1 | 2 | 0 |
| 0 | 291 | 27 | 2 |
| 7 | 15 | 295 | 3 |
| 0 | 1 | 32 | 17 |

*Figure 13: Confusion Matrix for 4 class classification*

Very similar trends were observed with Caltech 256. However, due to the overall size of the dataset. Extensive experimentation was not computationally possible.

| k | Random Forests | Naive Bayes |
|---|---|---|
| 20 | 10.18 | 7.74 |

*Figure 14: Percentage Accuracy for Caltech 256 with 20 clusters*

When subsets of Caltech 256 were chosen with large swathes of training images, the random forest classifier again performed well with 90% accuracy.


*Figure 15: Object Localization Algorithm Illustration*

When we give our object localization and hierarchical segmentation algorithm the picture above, with just a two level pyramid, it correctly classifies the region as a face and localizes it to the top-left.

## VII. Code

Our code for this project can be found here: https://github.com/shashankgargnyu/Machine-Learning-Project

## VIII. Video Link

Our presentation for this project can be found here: https://youtu.be/chy--M5pPj8

## IX. Evaluation

The classification accuracy for the 4 categories of interest in Caltech-101 are very competitive with most algorithms. This is in part due to having an abundance of training and testing data (2154 images in 4 classes). This accuracy drops off significantly when the object classes start having fewer images within them. This is to be expected, and could be circumvented by only considering the classes with a large amount of training images.

A bag of visual words model is quite computationally expensive to do on a large scale due to the SIFT feature extraction. We could speed up our algorithm significantly if we decided to use a competing image feature such as SURF as in [9]. However, SIFT is the de-facto industry standard in computer vision due to its robustness.

Hierarchical segmentation, while powerful, does have some limitations. The algorithm faces difficulty when the object of interest, lies on the exact lines of division of the pyramid. For example, if a face lies in the center of the image, after segmentation, the face might be split into 4 quarters thus increasing the likelihood of being misclassified.

This problem does have a solution as in [4], where the blocks are made to overlap with each other and a solution is found in a bound and optimized manner. If we had more time, we could have implement overlapping blocks at each level of the pyramid.

## X. Conclusion

Images are hard to characterize. How do you efficiently represent an image? A bag of words classifier has various uses in natural language processing, and can be applied to images by forcing them to be more like documents. However, historically this has always cost us spatial detail within our classification. The algorithm we implement here tackles this issue of spatial recognition, and implements it in a robust manner.