Semantic Labeling of Images

Ashish Gupta
Dept. of Computer Science and Engineering
University at Buffalo, State University of New York
agupta28@buffalo.edu

Abstract

Humans have this amazing ability of classifying even the most complex of objects in an image in milliseconds given they have seen the object in the past. Exploiting the ability of humans to learn, I present a method in this paper in which the model learns about the shape, color, texture and the perspective of labeled oversegmented superpixels of an image by extracting the low level features from the training set of 572 images (~ 300,000 superpixels) and uses the learned model to classify superpixels from the test set of 143 images (~80,000 superpixels).

The method described in this paper extract low level features such as edges, textures, RGB values, HSV values, location , number of line pixels per superpixel etc. to train the model using a Support Vector Machine and semantically label the superpixels in test set with labels such as sky, tree, road, grass, water, building, mountains & foreground objects. The results were then compared with ground truth to evaluate the accuracy of the model. The highest accuracy achieved was 45.25%.

Figure 1. Image of a boat in an ocean. We can differentiate the boat from the water because of its color and the occurrence of strong edges and lines in the boat which are missing in the water.

Figure 2. A scene from a city. We can differentiate water from an urban setting because of the difference in the color. The water is mainly colored with hues of blue with no strong lines.

1. Introduction

How do we humans identify objects? How do we learn to differentiate one object from the other? Consider the images shown in Fig 1 and Fig 2. The y position in the image helps us differentiate between the sky and the land, the unique color of water helps us identify water from other objects, edges help us identify objects in a vast expanse of water, texture can help us identify trees from buildings, occurrence of long lines can help us identify building from rest of the image.

Humans quickly grasp all these features and hence can reason about a scene. Only if we could make machines learn all these features and make them reason about a scene, the machines can then semantically label different objects in a scene and perform much more complex tasks.

The problem I am trying to solve in this paper is to help a machine learn and reason about different objects in the image. The task involves classifying the different regions in an images as sky, tree, foreground etc. The first step in this is to decompose a scene into already identified oversegmented superpixels and then use their y-position to reason about sky or land, the occurrence of lines, the occurrence of edges and their orientation , the variety of colors in a superpixel etc. and use this cues to help convert a superpixel into a mathematical notation and then use this reasoning to semantically label various superpixels relatively based on the model learned from the above mentioned reasoning.

Why I am trying to solve this problem is unambiguous and stems from need of many complex task such as image retrieval, image organization and intelligent image processing. If the machine can understand the scene then it can more intelligently adjust the exposure and color rather than just adjusting it to some fixed value. If the machine can understand the various cues in the scene, image retrieval and organization can then be reasoned and performed using those cues as the query. [1]

Say we want to find and organize all images of tree and forests together. A very strong cue for classifying trees is the occurrence of various hues of green together, strong textural patterns (repeated pattern of leaves) and strong edges at the corners. If we can help a machine understand a tree from a non-tree by extracting the above mentioned features, we could easily group all tree images together and answer user queries for image retrieval.

The proposed superpixel classification approach in this paper extracts many low level features from randomly chosen subset of ~10,000 superpixels from a massive training set (of ~ 300,000 superpixels) . These features are then grouped together to form a feature vector for each superpixel . These feature vectors are then fed to Support Vector Machine to learn 8 one vs all classifiers using a liner kernel for multi class classification problem. At the end I also discuss the use of SIFT features & LBP descriptors which can be used for classification problems like these.

2. Related Work

The work described in this paper touches many aspects of computer vision. Choosing features which best describe a given superpixel or features which produce least number of false positives for a given object is itself a huge task.

Many researchers have done a tremendous job in feature extraction for classifying images [1, 2, 3, 5]. Aditya *et al.* [5] for example used features such as color histogram, edge direction histogram and DCT coefficients for classification of city images vs landscapes in a k-NN classifier and achieved an accuracy of 93.9% on an image database of 2716 images.

Hoeim *et al.* [3] used features such as color, texture gradients, perspective , vanishing points ,shading etc. to recover surface layout from an image and label the image. into geometric classes . I use number of long line pixels per superpixels as one of the perspective feature [11] which turned out to be helpful in classification of foreground objects.

Olivier *et al.* [7] showed that support vector machines (SVM's) can generalize well on difficult image classification problems where the some features are high dimensional histograms. I use histograms such as hue and saturation histograms quantized into bins as some of the features.

More advanced methods based on bag of words have been used by many scientists in the past [6, 8, 9]. Notably , Gabriella *et al.* used vector quantized affine invariant descriptors of image patches with SVMs and Naïve Bayes Classifier to classify 7 semantic visual categories.

3. Feature Extraction

One of the main challenges we face in classification problems is choosing the right set of features which gives as high accuracy as possible. Features should cover low level details like the color distribution, the orientation of the edges, number of long line pixels in the superpixel , location with respect to the horizon , texture response and should preserve the relations between them. The list of features I extracted and used for classification are mentioned in Fig 3. Why each of the feature has been used is explained below.

Location of a superpixel in an image itself provides strong cues about the objects in the superpixel. A superpixel appearing high in the image would correspond to the sky or mountain than the pixels appearing low in the image which might correspond to the road, tree , grass etc. I normalize the y-position of the pixels in the superpixels and create a histogram of the normalized y positions into 8 bins. I also compute the number of pixels in a superpixel and the normalized area of a superpixel. [3]

Labeling Cues – Features Used

Location & Shape

Normalized y-position histogram
Number of pixels in a superpixel
Normalized area of superpixel in an image

Color

Mean RGB values in a segment
HSV representation of the above value
Hue : Histogram with 5 bins
Saturation : Histogram with 3 bins

Texture

LM filters : mean absolute response (15 filters)
LM filters : histogram of maximum responses (15filters)

Perspective

Number of long line pixels in a superpixel

5 extra features added over the default implementation: Number of pixels in a superpixel, Normalized area of superpixel in an image, Hue histogram, saturation histogram, Number of long line pixels in a superpixel

Figure 3 : List of features extracted to capture low level details from superpixels

Next I use the RGB and HSV values in the segment to model the superpixels. Color distribution in a superpixels can help us easily identify what is contained in a superpixel . Grass is normally green in color and the mountains are brown. Sky is either blue or white. I use the mean values of the RGB values of all pixels in a superpixel . I also use the HSV representation of the mean calculated above. HSV representations rearranges the geometry of RGB and provides perceptual details of the color to be captured which are missing in RGB. I have also used the Hue values of a superpixels quantized into 5 bins and the Saturation values quantized into 3 bins.

Next I use mean absolute responses of LM filters and a histogram of max responses for the LM filters. These LM filters capture the textural details of the superpixel. Texture details provide some information about the geometric shape and size of the object contained in the superpixel. For e.g. edge, corner and blob responses are some of the most significant filter responses which help us in reasoning about the shape of the one object from the other. The scale and orientation of the image gradients tell us a great deal about the shape of the object.

I have tried to extract some 3D perspective information by using the number of line pixels per superpixel normalized by the square root of the area of the superpixel as one of the features. Long lines in an image convey information about the vanishing points in an image. Roads and buildings have have more line pixels per superpixel than a grass which will have many small line segments but no long line segments . [3]

I have used Hough transform to fit long lines to the canny edge detector response.[11] The main problem that I faced was removal of false lines. This required tuning of the hough transform to ignore small line segments and not to merge too many small lines together as doing that led to too many false line pixels being introduced in the superpixel as shown in Fig 4.

Figure 4. Many false line pixels can be seen because of many small line segments merged together.

After tuning the Hough transform to ignore false line segments, the total found line segments in an image were recorded. On an average, 20 line segments were identified per image. Each pixel in a superpixel was then tested for occurrence on any of the lines using the equation of the line segment, y = mx + b. The slope was calculated using the two end points of each line segment. The total number of pixels occurring on the lines were then recorded as the the number of line pixels per superpixels which was then divided by the square root of the area of the superpixel. Fig 5 shows the optimal line segments extracted from an image.

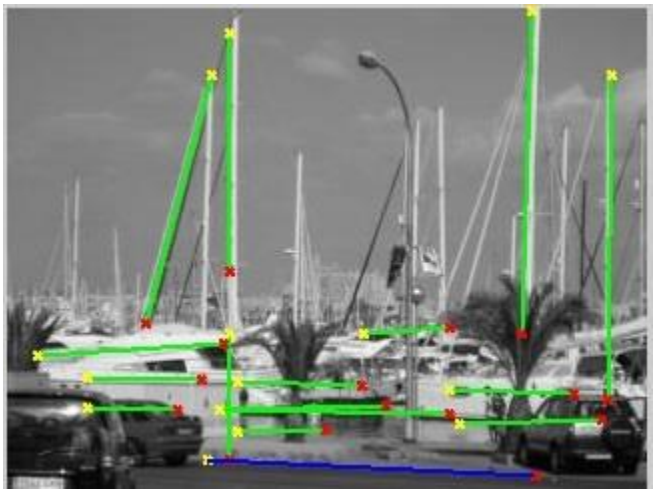

Figure 5. Hough line segments extracted from the images to identify the line segments.

I also tried to explore the opportunities of using SIFT and LBP descriptors using VLfeat for classification which is explained in the material below [10].

4. Classification

I have used LibSVM implementation of a Support Vector Machine to generate models for classification. An SVM model is a representation of the observations as points in space, mapped so that the observations of the separate categories are divided by a clear gap that is as wide as possible. New observations are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on. After all the features were extracted , the feature vector of 1,000-20,000 superpixels was fed to the SVM to generated 8 , one class vs other classification models which were then used to predict ~80,000 superpixels in the test set. Tuning the Radial Basis Function kernel turned out to be fruitless as it performed bad with accuracies ranging from 12-18%. A Linear kernel performed much better producing accuracies of 38-45.2% across all 5 folds.

5. Experimental Results

Legend for classes: 1 - 'Sky' 2- 'Tree' 3 - 'Road' 4 - 'Grass' 5 - 'Water' 6 - 'Bldg' 7- 'Mtn' 8 - 'Fground'

RBF Kernel:

I started with using a radial basis function kernel to train my model on 1000 randomly chosen superpixels. I first used the default set of features which gave accuracy ~12% as shown in Figure 6. After adding 5 more features above the default set as mentioned, the average accuracy slightly improved but were relatively bad and the best performing fold was fold 2 with an average accuracy of 12.5009% as shown in Figure 7. There were more false negatives when extra features were added .

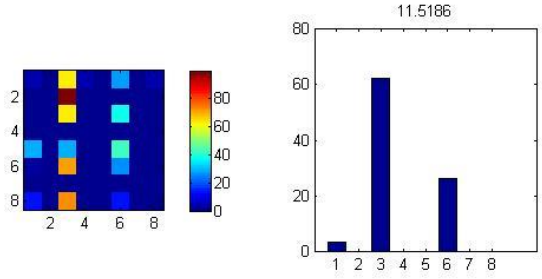

Figure 6: 1000 superpixels , default features , RBF

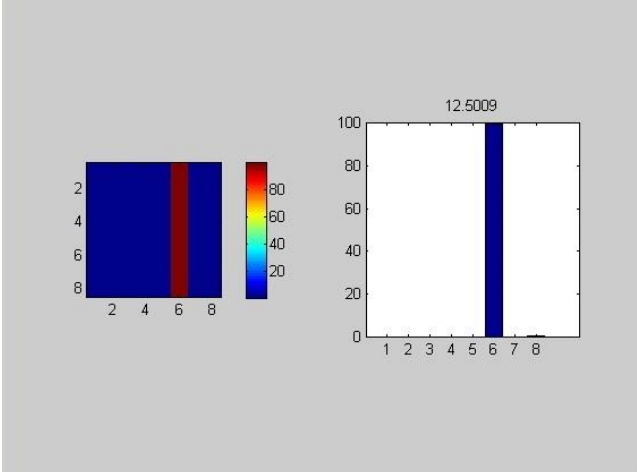

Figure 7: After adding 5 extra features, 1000 superpixels . RBF

I next tried to learn a model by using RBF kernel and using a random set of 20,000 superpixels. Figure 8 shows the average accuracy achieved on fold 4 using the default set of features and the Figure 9 shows the average accuracy achieved on fold 4 using 5 extra features.

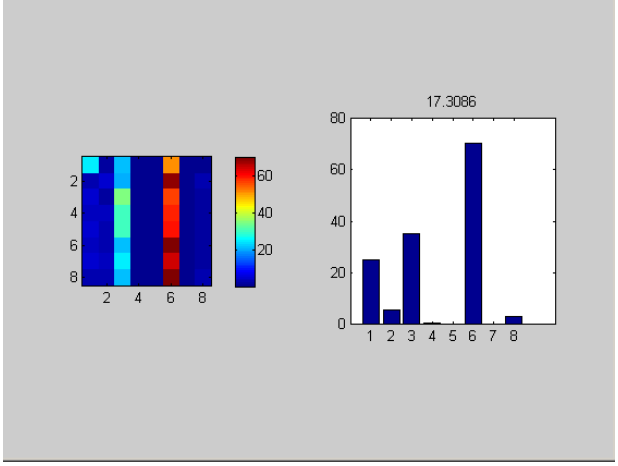

Figure 8: 20,000 superpixels , default features , RBF

Adding extra features did not significantly improve the average accuracy but produced slightly better accuracies for classes 3 and 6 which could be because of the long line pixels per super pixels feature as shown in Figure 9.

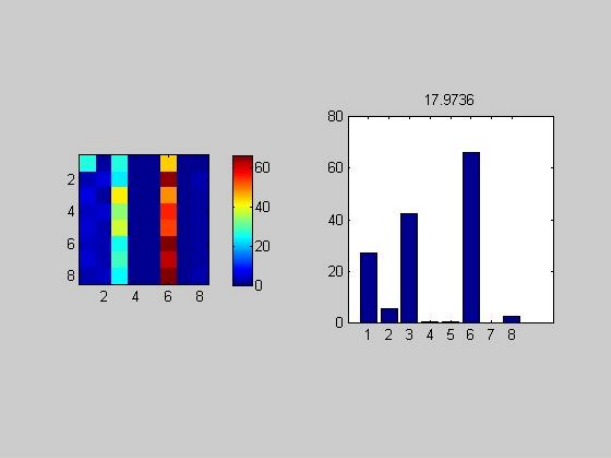

Figure 9 : 5 extra features, 20000 superpixels , RBF

As can be seen from the results, adding more superpixels and adding variety of superpixels helps to train a better model to classify superpixels as the system can better differentiate between the false positives and the real examples. These experiments also show that the random superpixels that were picked or the data set has high occurrences of class 3 and class 6, 'road' and 'building'.

Linear Kernel

After not getting satisfactory accuracies using the RBF kernel , I switched to using libSVMs one vs others classification using a Linear Kernel and the averages accuracy achieved for the default set of features across 5 folds using 5000 superpixels was : 42.28% as shown in Fig 10 with highest accuracy achieved for fold 2 : 44.2027% as shown in the Fig 11.

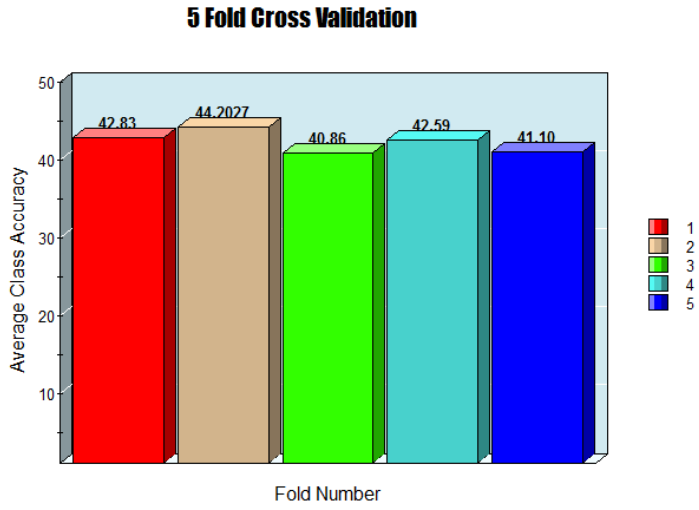

Figure 10 : Default features, Linear Kernel, 5000 superpixels

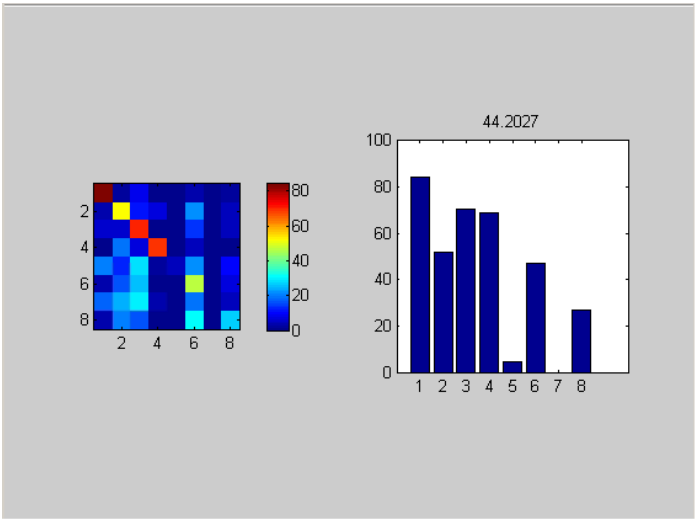

Figure 11 : Default features , Linear Kernel , 5000 superpixels

Then I tried to add some more features as mentioned in Fig 3 over the default implementation. Though the results did not change much, the results achieved were satisfactory across all folds as the learned model showed consistent performance across the 4 folds with an average accuracy of 42.44% across the 4 folds as shown in Fig 12 with highest accuracy achieved for fold 4 at 42,59% as shown in Fig 13.

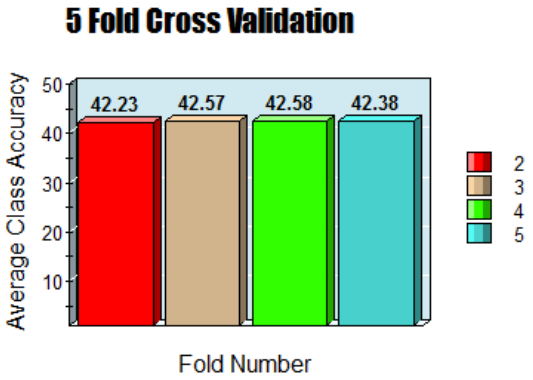

Figure 12 : 5000 superpixels, 5 extra features , Linear kernel

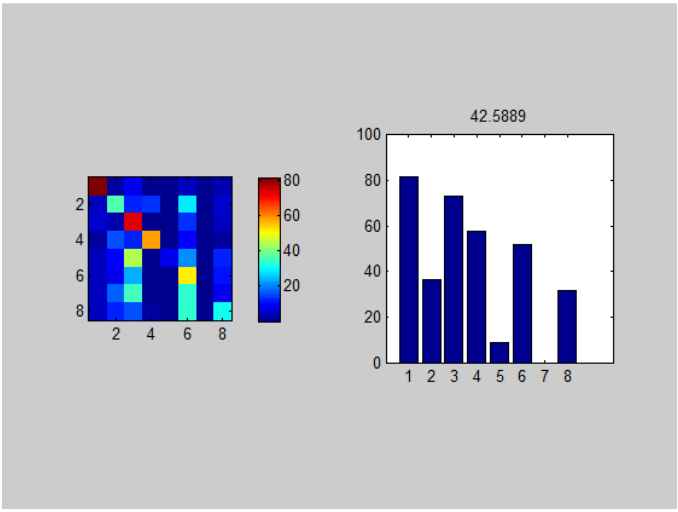

Figure 13: 5 extra features, 5000 superpixels , Linear kernel

Though, adding extra features did not give high accuracy as 44% but performed consistently across all 5 folds and produced a better average accuracy across 5 folds by 0.16 %

The highest accuracy that I achieved was **45.25%** as shown in figure 14. This was achieved with 10,000 superpixels with 5 extra features on the 4th fold using a Linear kernel.

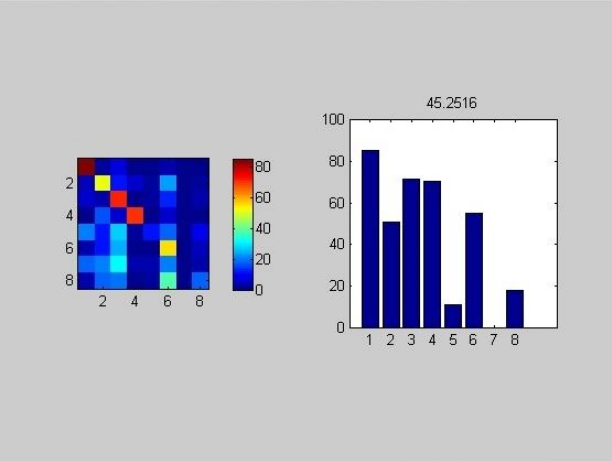

Figure 14: 10,000 superpixels , 5 extra features, Linear Kernel

6. Conclusions and future work

Defining a good set of features to attain higher accuracies and tuning the classifier were two major tasks involved in the work mentioned in this paper. As can be seen from the results, class 7 mountain was rarely recognized as the data set was lacking in the superpixels for mountains. The learned model performed well in not classifying anything as mountain as there were too less of them.

As is evident from the experiments, using a large training set with a mixed set of observations for each class performs better. A classification model learns better when there are more negative results than positive results for a given class.

Also it is evident from the experiments, that the features used did not produce excellent results. Some ways which have potential to improve the accuracy is by using SIFT features[9] and LBP descriptors [8] . After extracting SIFT features from all the images , we can perform quantization of SIFT features into say 200 clusters and represent each cluster as a bin. Each superpixel which has SIFT features can then be expressed as histogram of those 200 bins .

We can do the same with LBP descriptors by clustering all extracted LBP features and clustering them into bins

and expressing the local binary patterns in the superpixel as a histogram of those bins.

Features when considered all on their own convey less information than when looked at in relation with their neighbors. For e.g. a car may span may span many superpixels, if we can make a relationship between the neighboring superpixels we can better identify the labels of each superpixel.

7. System Specifications

Personal machine :
AMD A6 2.9 GHz
4GB RAM

UB Virtual Computing Lab

The average running time for 1000 superpixels was 5 mins, for 5000 superpixels it was 30 mins for RBF kernel and 1.5 hours for a linear kernel, for 10,000 superpixels it ranged from 4-6 hours for each fold on both the UB VCL and my personal system, for >= 20,000 superpixels , the running time ranged from >9 hours on both the systems used.

8. References

[1] Martin Szummer and Rosalind W. Picard : Indoor-Outdoor Image Classification, MIT Media Lab, Cambridge, MA 1998
[2] Robert M. Haralick, K. Shanmugam, It'shak dinstein : Textural features for Image Classification , IEEE Transactions on Systems, Man, and Cybernetics, 1973
[3] Derek Hoeim, Alexela A. Efros, Martial Hebert: Recovering Surface Layout from an Image , Carnegie Mellon University , 2006
[4] Stephen Gould, Richard Fulton, Daphne Koller , Stanford University : Decomposing a scene into Geometric and Semantically Consistent Regions , 2009
[5] Aditya Vailaya, Anil Jain, Hing Jiang Zhang : On Image Classification : City Images vs Landscapes, Michigan State University , 1998
[6] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, Cédric Bray: Visual Categorization with Bags of Keypoints , Xerox Research Center , Meylan ,France,2004
[7] Olivier Chapelle, Patrick Haffner, and Vladimir N. Vapnik : Support Vector Machines for Histogram-Based Image Classification , IEEE transaction on Neural Networks , 1999
[8] Timo Ojala, Matti PietikaÈinen, Senior Member, IEEE, and Topi MaÈenpaÈaÈ , Multiresolution Gray-Scale and Rotation Invariant Texture Classification with Local Binary Patterns , 2002
[9] Image Classification with SIFT features and SVM, http://dsp.stackexchange.com/questions/5979/image-classification-using-sift-features-and-svm
[10] VLFeat Computer Vision Library for SIFT and LBP descriptors : http://www.vlfeat.org/overview/sift.html
[11] Extracting Line Segments using Hough Transform , MATLAB
http://www.mathworks.com/help/images/ref/houghlines.html
[12]LibSVM http://www.csie.ntu.edu.tw/~cjlin/libsvm/