Semantic Labeling of Images: Design and Analysis

Aayush Uppal, 50134711
The State University of New York, University at Buffalo
aayushup@buffalo.edu

Abstract

The process of analyzing a scene and decomposing it into logical partitions or semantic segments is what semantic labeling of images refers to. The capability is inherent in human beings, when we see an image we are readily able to classify every part of it as either a person, building, sky etc. depending upon our learning of surroundings and objects. In this paper we discuss the approach to extend this capability to Computer Vision and analyze how some features, intrinsic to a scene impact our understanding and classification of labels.

1. Introduction

The basic understanding of an image from a human perspective lies in the broader yet much more intensive classification with respect to real world objects and surroundings. This process is at a higher level and much like any other machine-human interaction scenario we will need to upscale from components at a lower level that fit into computer vision analysis parameters directly to interpret these higher level features.

In order to decompose features at a higher and more holistic level we need to move from pixel level to a more grouped and unified basic unit for image understanding furthermore this also helps us reduce computational intensiveness widely, to implement this we use a superpixel as a basic block for scene understanding. For our image data we are provided input images pre segmented in superpixels. It is worth mentioning here the basic metric behind superpixel calculation is an adaptive clustering technique based on color and image plane space so it does not compromise on pixel information data.

Our benchmark dataset is the “Stanford Background Dataset”, a set of 715 benchmark images from urban and rural scenes each image size of 320*240 pixels. Each image here has at least one foreground object and has the horizon positioned within the image. Each image has a different number of superpixels as decided by the algorithm employed to calculate superpixels which ultimately renders the initial pixel size image measure obsolete and our ultimate processing comes down to number of superpixels.

The crux component of our approach involves pixel to superpixel generation and then realizing a feature space, as mentioned earlier the feature space parameters that fit directly in computer vision analysis parameters and hence we generate the high level classification. In our approach the feature space is composed of RGB color space values, HSV values and there manipulations as mean and maximum values, mean texture response, maximum texture response and superpixel position respective to a scene. The RGB and HSV color space parameters have been extracted from image data provided in the input dataset while to generate mean and maximum texture response we use LM (Leung-Malik) Filter bank which consists of first and second derivatives of Gaussians at 6 orientations and 3 scales making a total of 36; 8 Laplacian of Gaussian (LOG) filters; and 4 Gaussians. LM filter bank ensures a comprehensive texture output but its relevancy to ensure accurate classification shall be discussed in the paper later. The position component is decided by row and column value and is expressed in a relative scenario to the scene for a superpixel. Furthermore optimal feature has been decided based upon the concept of Markov Random Field by setting edge relations between neighborhoods generated from adjacency matrix and determining the most common feature value and maximizing the same, this is a very basic implementation based on the concept of estimation via Markov Blanket for a superpixel that is the node in our case. All these combined give up to 52 dimensional feature vector space to start our analysis of image labeling. The feature vector space has been heavily modified and analyzed in the process of understanding of effective image classification and accurate labels. On the basis of this available vector space comparative analysis has been done for added / removed features and its impact on specific classes.

This feature space generated for the entire dataset is used to train out model based on Support Vector Machine, again in SVM analysis was done for several tuning parameters to improve accuracy of classification and labeling. Finally a test metric has been defined to set up a confusion matrix, this matrix represents how many of the existing superpixels in a particular class are correctly classified for the given input fold.

2. Related Work

My approach here encompasses several aspects of Computer Vision & Image Processing and mathematical analysis paradigms, plenty of research has been done in each of these areas individually. To start with, the problem of generation of superpixels has been done via several algorithms and addressed in very optimal ways [1, 2], the generic approach though has been based on classification via image and color plane features along with texture based clustering to choose superpixel size [3].

For feature generation and analysis of texture extraction a basic approach based on LM Filters has been done [4]. In LM filters approach in our case we analyze histogram response and mean response based on output from all the filters in the filter bank this generates 30 dimensions in the feature vector space.

For the concept of Markov Blanket based prediction using Adjacency Matrix [5] to select most weighted feature vector dimension and then furthermore to enhance test output labels. Another source of reference that greatly supports the use of a neighbor approach [6]. The work done here in comparative analysis of use of nearest neighbor based image classification against state of the art image classification methods provides a strong point to pursue the approach.

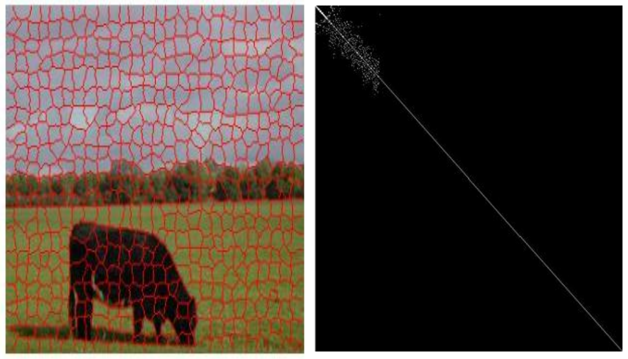
The importance of color based features is inherent in the analysis here we encounter distinctions between classes like ‘grass’ amongst others which rely heavily upon color based features. I have extracted color based features from data set and it’s representation in RGB space and HSV space [7].

The classifier approach to train is one of the trickiest aspects apart from the feature space analysis. We need to classify our features into multiple classes here, 8 classes. A large number of approaches are available to pursue this namely via neural network classifiers, conditional random field classifiers, in my work I have used Support Vector Machine extensively and worked on its tuning parameters which has enabled tremendous performance boost. In particular ‘fitcsvm’ model has been used which is basically an optimized ‘libsvm’ for MATLAB. Now it must be noted that Support Vector Machine is inherently used for linear classification we are able to extend it to 8 classes upon recursive implementation. A good resource and an advocate of use for linear kernel function [8].

3. Approach/Algorithm

The workflow of my approach broadly involves the following steps in respective order. To start with we have the data consolidation followed by feature extraction, model training and testing.

The final aim of the project is to perform semantic labeling o images given a benchmark dataset, the Stanford Background Dataset and then evaluate the labeling procedure. In our data consolidation phase we load the image data for all images stored in a state variable for MATLAB. Each image loaded here has a fixed dimension of 320*240 pixels and more importantly each image is chosen such that there is at least one foreground object and the horizon is positioned within the image. This is entirely essential because we have a label class ‘foreground object’. Now this dataset is further divided into different folds so as to enable us to choose different iteration cases in order to improve accuracy over successive iterations by choosing best fold for a case. A single fold is run during one time execution and each fold basically has 715 images split into 572 training and 143 testing images over which the performance is scored. One of the key and time saving features of the dataset being provided to us is the heavy pre-processing done on it, we have superpixels defined for each image generated by one of the generic methods discussed earlier (not within the scope of discussion in this project), this in particular is very helpful because it reduces our computational intensiveness from pixel level to super pixel level that is to say an image has 320*240 pixels now has only 560 superpixels i.e. on an average a 99% decrease in number of basic block units to be processed, apart from this superpixel gives a more holistic grouping of scene and enables to group images in a continuous and more semantic way.



(i) (ii)

Fig 1: Example of an image represntation from dataset in it’s superpixels segmentation (i) and adjacency matrix representation of the image (ii).

Another highly important data being provided is the Adjacency matrix for each superpixel, this is the matrix representation of all the neighbors of a superpixel in an image and helps me greatly to implement the concepts from “Nearest Neighbor Based Image Classification” combined with concepts of “Markov Blankets for Causal Structure Learning”, The key concept being that the neighbor pixels have a strong and defining influence on the centrally located pixel, the relevance of the concept can be understood from the following example that, say we have a stray grass classified superpixels amidst all sky superpixels it is highly likely that the classification of grass is wrong, so we set a threshold value of probability and chose when to update the grass superpixel to sky superpixel.
(Note: The adjacency matrix supplied has some anomalies for superpixel values greater than approximately 100 which has compromised upon the accuracy of results from expected values.)

Feature space representation and extraction has been one of the key areas of focus in this project for both the design and analysis perspective of our semantic labeling logic. To start with we had the basic feature space vector of 44 dimension consisting primarily of texture, generated via LM Filters’ Histogram and Mean value analysis, Color space representation – RGB color space and HSV color space and spatial position in 2D for superpixels. One of the first basic things to do intuitively was to understand the relevance of Location cues implemented in features, given in the set we had only used ‘y’ position values so instead we run the dataset to generate a feature set now to see what difference x position values make. After this I removed entire location based features to see its impact on particular classes and their accuracies. One of the strongest metric intuitively is color space representation and so I analyzed its impact on classification. Then finally for the last stage of reductional feature analysis I used mean and max texture response from LM filter output in a disjoint fashion.

Moving on to some additional features the first feature added was x position along with y position for stronger Location sense. Upon addition I generated upto 52 dimensional feature vector space. Next up after this I tried to use the concepts of Markov Blanket considering the superpixel in consideration as my node and the edge relations being generated from all those in the strict neighborhood, this was done because it has been proven by work mentioned earlier [6] that Near neighborhood is indeed a very effective metric. To realize this concept I chose all the feature vectors of all neighbors and then calculated the strongest feature, the strongest feature thus generated in each case subsequently was given a double weight in feature vector space.

After this analysis was done on Folds dataset with respect to corresponding accuracy output in each scenario.

The penultimate phase was the training phase in this we train over the feature set generated in previous phase. For our training phase here we use the Support Vector Machine model defined by the function fitcsvm in MATLAB. By definition SVM is used when we have a linear classification problem that is between 2 classes per se a feature belongs to class 1 or class 2. In our case we have 8 classes so we do an intensive process of analyzing for each class individually eg: check feature 1 in class 1 or not, then in class 2 or not and so on until class 8. This is one computationally intensive drawback of approach here.

The fitcsvm model used for initial iteration of training was a radial basis function.

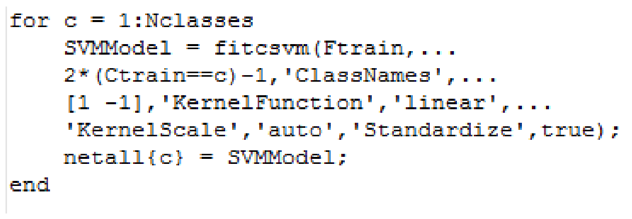
Model = fitcsvm(X,Y)

This returns support vector machine classifier trained by predictors X and class labels Y. The RBF kernel runs on two samples x and x’, the feature vectors from input and is defined as

$$K(\mathbf{x},\mathbf{x'}) = \exp\left(-\frac{||\mathbf{x}-\mathbf{x'}||^2}{2\sigma^2}\right)$$

Based upon understanding and literature review I tried several other kernel function parameters starting with linear kernel function, next I tried polynomial function of the order of 2, 3 and 4. What these kernel functions basically do is they provide solutions to generate hyper plane such that it is at optimum and maximum distance from separate classes identified as the order increases we are able to generate more solutions and better probability of much better hyper plane. I also found out another tuning feature for svm model to train dataset ‘KernelScale’, when set to 'auto', then the software uses a heuristic procedure to select the scale value. The heuristic procedure uses subsampling. Therefore, to reproduce results, set a random number seed using rng before training the classifier.
In our case Linear kernel stands out in practice along with KernelScale set to auto, primarily because we are recursively classifying each class individually from 1 to 8.
Linear kernel is given by the inner product <x,y> plus an optional constant **c**.

$$k(x,y) = x^T y + c$$



```
for c = 1:Nclasses
    SVMModel = fitcsvm(Ftrain,...
    2*(Ctrain==c)-1,'ClassNames',...
    [1 -1],'KernelFunction','linear',...
    'KernelScale','auto','Standardize',true);
    netall{c} = SVMModel;
end
```

It needs to be noted that in training phase we specify the folds for iteration of training learning process and of this it chooses the train set of 572 images.

The final phase is the test phase, though in the basic approach nothing much has to be done in this phase besides choosing a test fold, we choose th the test fold corresponding to the train fold input in previous phase. Over here we use the predict function, the predict function basically scores and sets the class vlaue for a superpixel this too is done iteratively over each superpixel, and it needs to be noted that 1 here represents presence of a superpixel in that class and -1 denotes absence. For better understanding I slightly modified the code and generated a label space of a 8 column matrix where each column corresponds to a class and 1, -1 represent the class of corresponding superpixel mentioned along the rows.

At this stage I tried another optimization technique using markov random blanket concept but due to inadequacy of representation in adjacency matrix not much difference was observed. Upon testing phase after generating the labels and mapping them to corresponding classes I updated the corresponding superpixels based on the value of their neigbors in order to minimize erroneous representations and false classes.

```
for f = 1:numel(imsegs)
    disp(numel(imsegs)-f);
    numspix = imsegs(f).nsegs;

    for i = 1:numspix


      adjcell = find(imsegs(f).adjmat(i,:)== 1);

            kw = numel(adjcell);
            az = zeros(1,kw);

            for op = 1:kw
            nbspix = adjcell(op);
            az(1,op) = imsegs(f).super_pixel_labels(nbspix);
            end
            xt = unique(az);
            ct = numel(xt);
            pct = zeros(1,ct);

            for yu=1:ct
            pct(1,yu) = numel(find(az == xt(yu)));
            end
            sdf = find(pct == max(pct));
            sdf = sdf(1);

            spixlbl = xt(sdf);
            imsegs(f).super_pixel_labels(i) = spixlbl;
    end

end
```

Finally a confusion matrix is generated, a confusion matrix representation is between 8 clasees on both axis and the matrix cells indicate the correct and incorrect classifications along the matrix, the diagonal being the ideal output where each class is classified as itself. This is evaluated over entire set of superpixels in test fold and then over the mean of all diagonal values calculated which represents our accuracy metric.

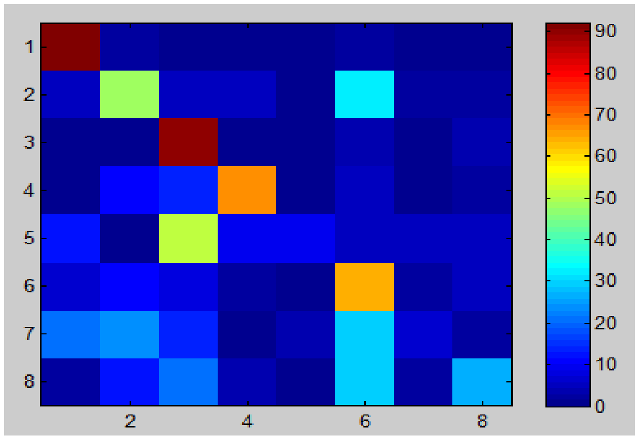
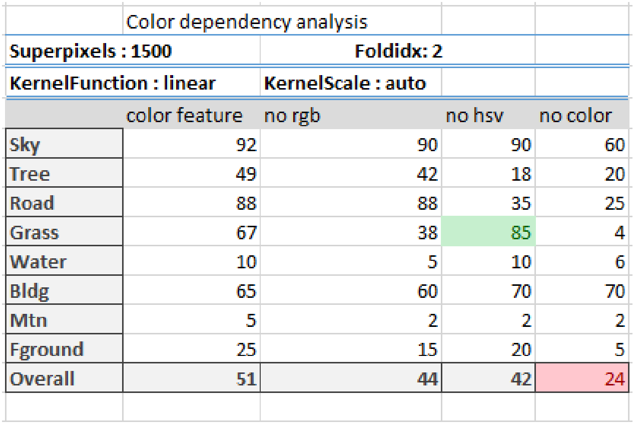


Fig 2: A sample confusion matrix generated for test fold output analysis.

4. Experimental analysis & Results

A series of comprehensive analysis was conducted as discussed, reductional phase feature analysis, additional phase, color dependency, adjacency matrix highest weighted feature & best fold iterations, all of these were in the feature phase implementation. Then another test was done based on test phase optimization using markov blanket dependency concept.

4.1. Color dependency analysis

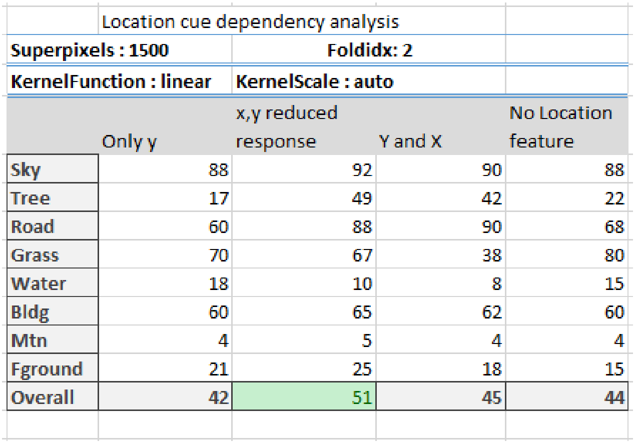


	Color dependency analysis			
Superpixels : 1500		**Foldidx: 2**		
KernelFunction : linear		**KernelScale : auto**		
	color feature	no rgb	no hsv	no color
Sky	92	90	90	60
Tree	49	42	18	20
Road	88	88	35	25
Grass	67	38	85	4
Water	10	5	10	6
Bldg	65	60	70	70
Mtn	5	2	2	2
Fground	25	15	20	5
Overall	51	44	42	24

We infer from this the relevancy of color feature in our image labeling as our accuracy drops from 51% to 24% overall. One striking feature is the increase in accuracy for grass feature classification when HSV is removed this can be attributed to the fact that a large amount was wrongly classified earlier in tree segment and upon removal of HSV it is set right in grass section, notice the rise in grass and considerable decrease in tree section upon comparison between columns 1 and 3.

Furthermore removal of RGB and HSV in general is a degrading factor.

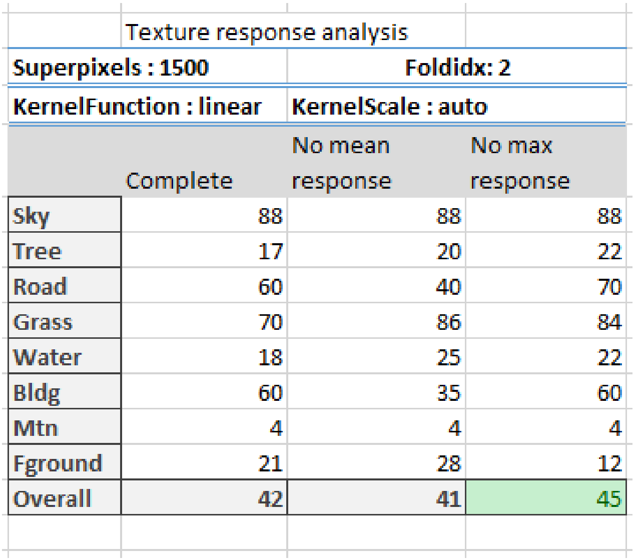
4.2. Location cue dependency analysis



Location cue dependency analysis				
Superpixels : 1500		**Foldidx: 2**		
KernelFunction : linear		**KernelScale : auto**		
	Only y	x,y reduced response	Y and X	No Location feature
Sky	88	92	90	88
Tree	17	49	42	22
Road	60	88	90	68
Grass	70	67	38	80
Water	18	10	8	15
Bldg	60	65	62	60
Mtn	4	5	4	4
Fground	21	25	18	15
Overall	42	51	45	44

The location analysis presents a fairly consistent output over all class labeling metrics, this goes on to highlight the fact from our empirical observation that position in this background data set is not that important a feature vector dimension. The max accuracy output though is obtained for a reduced x, y response and not a no location feature. For our reduced response we use a combined function of x and y over just 2 values instead of 8 feature parameters.

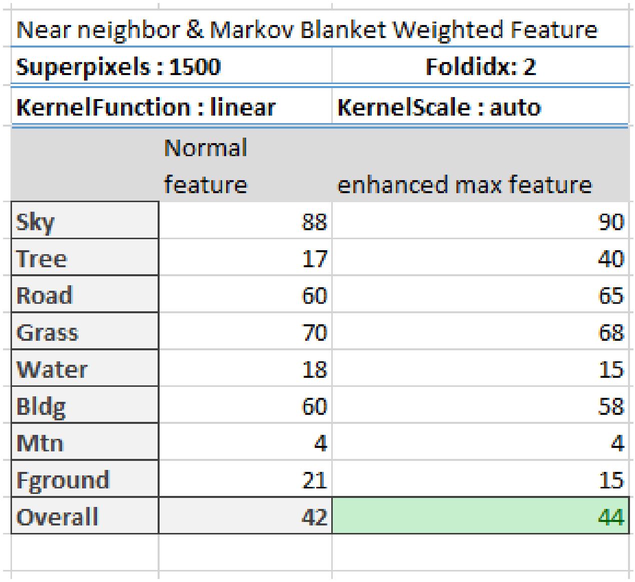
4.3. Texture response analysis



Texture response analysis			
Superpixels : 1500		**Foldidx: 2**	
KernelFunction : linear		**KernelScale : auto**	
	Complete	No mean response	No max response
Sky	88	88	88
Tree	17	20	22
Road	60	40	70
Grass	70	86	84
Water	18	25	22
Bldg	60	35	60
Mtn	4	4	4
Fground	21	28	12
Overall	42	41	45

Texture response analysis displays a comparative study between usefulness of texture as a feature dimension. Texture is a fairly relevant feature and most of all the mean texture response is a better parameter. Finally we realize max response should be removed and mean response should be used.

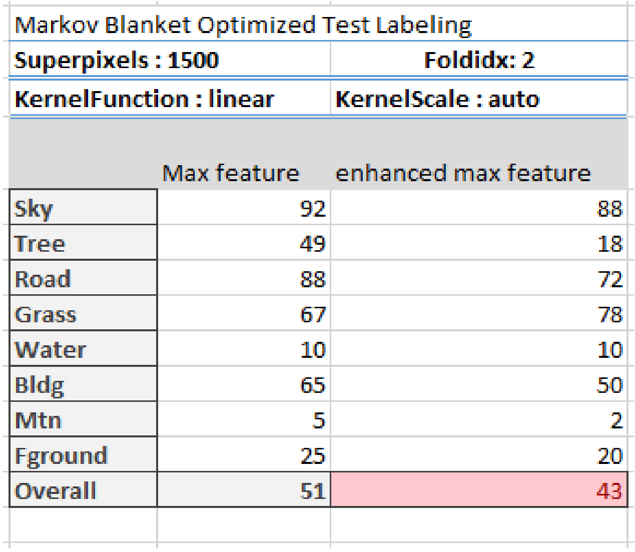
4.4. Near neighbor & Markov Blanket Weighted Feature



Near neighbor & Markov Blanket Weighted Feature		
Superpixels : 1500		**Foldidx: 2**
KernelFunction : linear		**KernelScale : auto**
	Normal feature	enhanced max feature
Sky	88	90
Tree	17	40
Road	60	65
Grass	70	68
Water	18	15
Bldg	60	58
Mtn	4	4
Fground	21	15
Overall	42	44

Near neighbor & Markov Blanket Weighted augmented feature refers to the double weighted maximum feature dimension determined via adjacency matrix. Notice there is a slight increase in the output parameters accuracy but not as much as would be expected or is told by theoretical studies this anomaly is due to the discrepancy mentioned in adjacency matrix set.

4.5. Markov Blanket Optimized Test Labeling



Markov Blanket Optimized Test Labeling		
Superpixels : 1500		**Foldidx: 2**
KernelFunction : linear		**KernelScale : auto**
	Max feature	enhanced max feature
Sky	92	88
Tree	49	18
Road	88	72
Grass	67	78
Water	10	10
Bldg	65	50
Mtn	5	2
Fground	25	20
Overall	51	43

In this final phase optimization I tried to augment labels after test output via using the markov blanket labels generated from predict function and then finding their neighbors from adjacent matrix. I expected a steep increase in accuracy but due to the limitation discussed in adjacency matrix there was no significant increase. I intend to rectify this in future implementation and reconstruct the corresponding neighbors by storing them in an adjacency list which will also help me to reduce computation time.

4.6. SVM Tuning Analysis

In this section we analyzed the impact of svm tuning parameters starting from the first response generated for the overall dataset run and trained over entire superpixel dataset. This figure uses a radial basis function for analysis, in rbf the corresponding mapping is infinite dimensional but it does not serve our purpose because we are recursively linearly classifying / not classifying into one class step by step and hence rbf may be encountering over fitting.

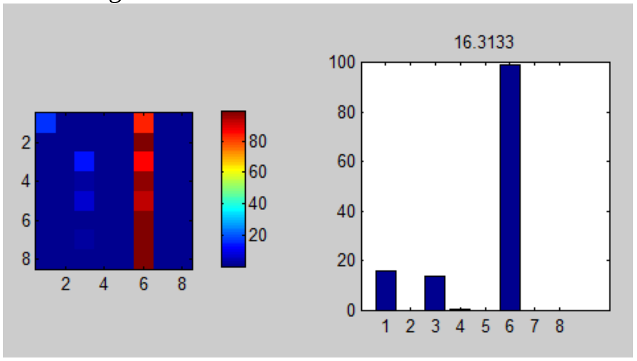


Fig 3: KernelFunction – rbf

In the next snapshot we see a trained output set for a relatively smaller dataset of superpixels over polynomial kernel function of order 3 and scale auto. This is as expected a much improvement from rbf.

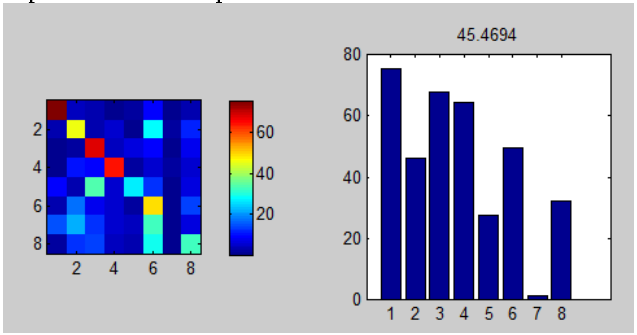


Fig 3: KernelFunction – polynomial, PolynomialOrder – 3, KernelScale – auto

In the final we show the linear kernel function used extensively over auto paramters. Clearly we see a transition from 16% to approx 51% and that too over a smaller dataset analysis.

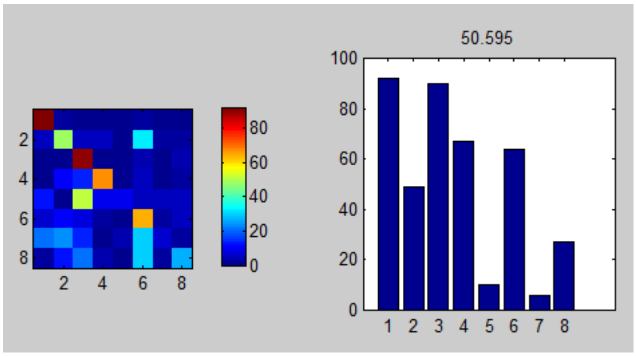


Fig5: Kernel Function – linear, KernelScale - auto

5. Discussions and Conclusions

In summary of this report we have realized the importance of several parameters in feature dataset and understood that even post classification augmentation techniques are possible. I expect from the latter phase a much higher accuracy can be generated for this background data set. I intend to use horizon as feature parameter because that is one of the prominent features in this dataset of images and maybe use of horizon would help us to identify mountains much better in landscape scenarios. To implement horizon I propose to use edge detection followed by Hough transform analysis and then choose the longest line segment in the background scenario only. Apart from this the post test phase adjacency matrix needs to be calibrated and optimized using an adjacency list of superpixels for faster computation and effective implementation of concepts of near neighbor and Markov Blanket. One other key aspect realized from this was the inability to classify mountain features in present implementation, this I believe can be rectified by use horizon feature and higher weighted color response output. On the overall I was able to show an accuracy of 51% for features in the rural and urban scenes also we have learnt about several ways in which this can be optimized and the drawbacks of a few features.

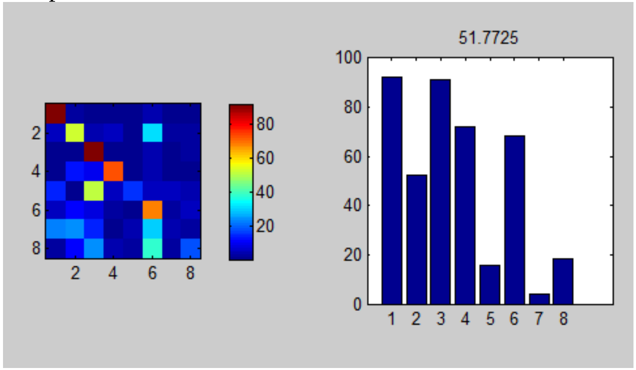


Fig 6: Final output for 5000 superpixels, Accuracy – 51.7%

References

[1] X. Ren and J. Malik. Learning a classification model for segmentation. In Proc. 9th Int. Conf. Computer Vision, volume 1, pages 10-17, 2003.

[2] G. Mori, X. Ren, A. Efros, and J. Malik, Recovering Human Body Configurations: Combining Segmentation and Recognition, IEEE Computer Vision and Pattern Recognition, 2004.

[3] SLIC Superpixels Compared to State-of-the-Art Superpixel Methods, IEEE, 2012.

[4] Manik Varma and Andrew Zisserman, A Statistical Approach to Texture Classification from Single Images, International Journal of Computer Vision, 2005.

[5] Jean-Philippe Pellet and Andre´ Elisseeff, Using Markov Blankets for Causal Structure Learning, Journal of Machine Learning Research, 2008.

[6] Boiman O, Shechtman E and Irani M, In Defense of Nearest-Neighbor Based Image Classification, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008.

[7] R.V.R Chary, Dr. D. Rajya Lakshmi and Dr. K.V.N Sunitha Feature Extraction Methods for Color Image Similarity, Advanced Computing International Journal, 2012.

[8] O Chapelle, P Haffner and V.N. Vapnik, Support Vector Machines for Histogram Based Image Classification, IEEE Transactions on Neural Networks, 1999.