Labeling Superpixels of Datasets through Feature Extraction and Classification of Images with Attempted SIFT Techniques

Danny Peng
CSE 473
CVIP – dannypen@bufalo.edu
50031597

Abstract

The goal of the following discussed techniques is to perform semantic labeling upon a set of training images and testing images. By utilizing a built-in SVM for classification and attempting to use a SIFT keypoint detector, a moderate accuracy evaluation can be achieved.

1. Introduction

The objective of the project is to achieve a high accuracy evaluation with labeling superpixels of images from a benchmark dataset; the ideal end result of a successful procedure to do so would be a correct assignment of labels to superpixels for every testing image, which include foreground and background. The eight labels that are assigned are: sky, tree, road, grass, water, building, mountain, and foreground. As the dataset of images has already been preprocessed for superpixels, the remaining components of the project is extracting features from the superpixels and training a classifier on the eight labels.

The significance of assigning labels to these segmented images lies in the spectrum of automatous work. While a human would be able to classify and label an image in a short amount of time, if an algorithm was conceived to perform the same scope of labor with the same accuracy without human intervention, then images could be classified and labeled automatically at a rate thousands of times faster than a human would be able to work. The result would benefit a myriad of fields, with emphasis on those fields focused on frame-by-frame video capture, such as security and traffic. If a sufficiently fast algorithm was built that was able to autonomously classify only upright humans in a single image, this would allow real-time tracking and detection without having a human supervisor watching a video feed.

The expected results from this project would allow a keen focus on what allows for a successful classifier to obtain a high accuracy rate. They would display which feature extracting techniques are effective with large enough datasets of superpixels. They would also show how accuracy rates change with increasing number of superpixels.

2. Related Work

One such related work is Darwin [1], a software package that generates superpixels within images. It performs this task through the SLIC superpixels and graph-cut superixels algorithms.

Another related work is image segmentation by Kovesi [2] through using the SLIC superpixels algorithm alongside DBSCAN clustering. These two techniques are able to generate a single segmented image in a relatively fast manner.

A branch of research dealing with the same techniques but used in a specific application is the Urban Feature Extraction project [3]. This project uses automated feature extraction from multi-sensor, multi-resolution, etc. in order to create 3D images of urban areas.

Another work that involves feature extracting is performed by Vadim Kantorov [4], in his Computer Vision and Pattern Recognition paper. His project focuses on providing state-of-the-art performance for recognition. While the accuracy of image recognition has been improved upon throughout its years of research, the low speed due to the bulky algorithm has been a problem in preventing real-time recognition from becoming a viable option. His project focuses on improving the speed of video feature extraction by creating a better and conceivable approach.

3. Approach/Algorithm

In brainstorming an approach, I sought out pre-classifier methods first instead of feature extraction methods. This is due to that there was already a sizable chunk of code that was dedicated to feature extraction within the stock code.

Before researching any topics, I compiled the stock code and ran the runfile, through all five folds of cross-validation. These results were exceptionally poor and varied from 10% to 13% in accuracy rating.

I then sought out to see if any improvements could be made within the SVM classifiers used in the stock code. Within the classifier parameters, I found that it had an argument relating to a kernel function. Through research, I discovered that the kernel function is a set of algorithms used for pattern analysis and are commonly used within SVMs. They perform mapping of data into higher-dimensionally spaces where the data can become easier to distinguish so that other techniques may be performed on it.

I saw that the current argument in the SVM classifier was the rbf kernel function, which is a family of function whose most commonly-used member is the Gaussian kernel function. The Gaussian kernel function uses the adjustable parameter gamma, which must be delicately adjusted with respect to its application [5]. If it is not, then the function will lack regularization and the results will be moreso clouded by the presence of noise in the images.

Instead of using the Gaussian kernel function which must be fine-tuned, I decided to try the linear kernel function instead. The linear kernel function is a simple kernel function that is given by the inner product <x,y> with an optional constant c.

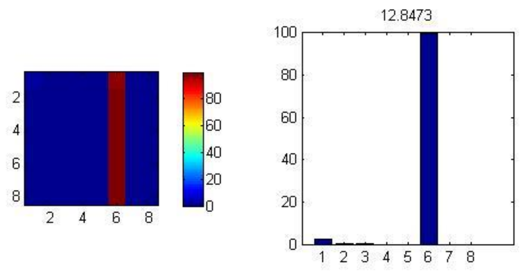
When I replaced the Gaussian kernel function with the linear kernel function, and then ran the runfile, the results were much more improved. With the linear kernel function, the accuracy rating jumped up 30% and higher with no notable difference in runtime.

Afterwards, I decided to try out a SIFT keypoint detector program [6]. Within this program, it utilizes the SIFT technique to extract features from images and create meaningful matches between images and then uses a bag-of-words methodology to use a SVM to learn a classifier. More specifically, the features would be extracted from training images and then k-means would be computed over the entire set of training images to then obtain cluster centers.

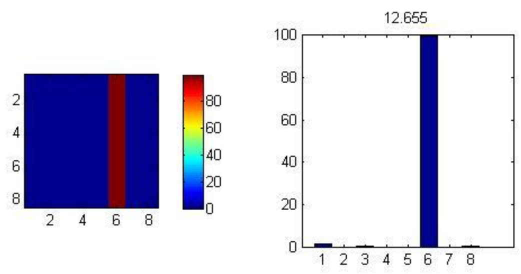
However, I did not succeed in doing so, due to a general confusion about how to utilize the SIFT program and where/when it should be incorporated within the current runfile. Therefore, I went on directly to compute various sizes of superpixels with my current code.

4. Experimental Analysis & Results

First, the labeling process was run with the default classifiers and no code changed. It was monitored at 1000 superpixels over 5-fold cross-validation. The initial accuracy for the first fold was 12.8473%, as documented in the figure below.



Over the course of five folds, the end result is shown below, with an accuracy of 12.6550% :

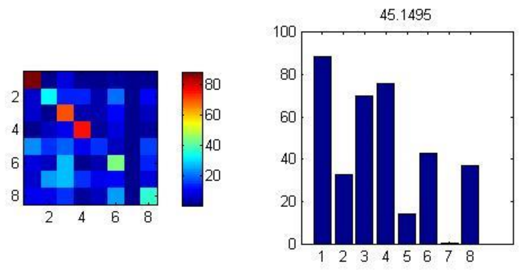


The following is a table of the accuracy results for each fold. The first column represents the fold number; the second column represents the accuracy results for the corresponding fold.

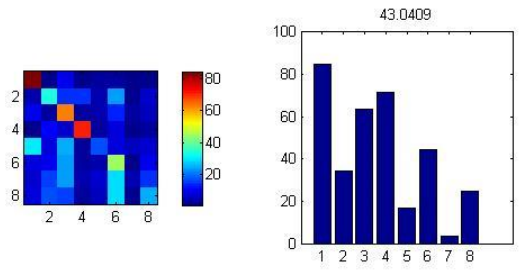
1	12.8473%
2	12.7162%
3	13.0093%
4	12.7144%
5	12.6550%

As these results are obtained with no changes to the stock code, they are clearly inferior and in need of improvements.

Then, the labeling process was run with the same feature extraction tools but with a linear kernel function instead. Below is the data obtained for the first fold at 1000 superpixels. The accuracy is improved greatly, rising up from 12.8473% to 45.1495%.



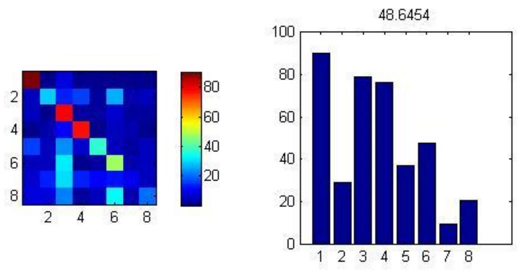
Over the course of five folds, the end result is shown below for the linear kernel function, with an accuracy of 43.0409% :

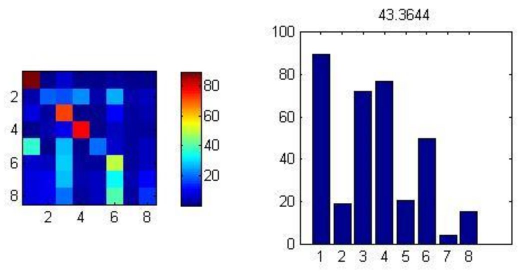


The following is a table of the accuracy results for each fold with the linear kernel function replacing the Gaussian kernel function. The first column represents the fold number; the second column represents the accuracy results for the corresponding fold.

1	45.1495%
2	43.5321%
3	40.7567%
4	44.6341%
5	43.0409%

Afterwards, I repeated the above runs but instead of 1000 superpixels, I increased the load by tenfold, increasing it to 10000 superpixels. The initial accuracy and end accuracies are shown below, at 48.6454% and 43.3644% respectively.





The following is a table of the accuracy results for each fold with the linear kernel function at 10000 superpixels. The first column represents the fold number; the second column represents the accuracy results for the corresponding fold.

1	48.6454%
2	47.6592%
3	45.6486%
4	44.0097%
5	43.3644%

The following is a table averaging up the results for each run: the stock labeling with default code, the linear kernel function at 1000 superpixels, and the linear kernel function at 10000 superpixels.

Stock Labeling	12.78844%
Linear Kernel – 1000	43.42266%
Linear Kernel – 10000	45.86546%

5. Discussion and Conclusions

Judging from the previous table, it is clear that the linear kernel function is superior to the rbf kernel function in terms of accuracy. In terms of runtime, there was no notable difference between the two runs.

Overall, these results show that the Gaussian kernel function must be fine-tuned to whatever application it is being used for; otherwise, the results will be subpar compared to using a linear kernel function for the SVM classifier.

Additionally, the accuracy rating increased by a total of approximately 5.6% from running the labeling process over 1000 superpixels to 10000 superpixels. This gain in accuracy shows that there is perhaps a correlation between the number of superpixels and the resulting accuracy percentage; at the very least, it shows that there is more consistency with a higher number of superpixels than a lower number.

My findings show that there is an extreme difficulty in this field of work and that the related works are built on top of previous layers of works; it is inconceivable to create a framework to perform image labeling and feature extraction from scratch.

The study's limitations are focused mostly on a lack of research due to time restraints and unintelligible resources; the information that can be found online relating to SIFT classifying is extremely technical and very difficult to understand fully so that it is challenging to incorporate it within another large project.

This study raises up questions regarding the runtime of image segmentation and feature extracting. In order for a successful environment where real-time feature extracting can occur to track live video feeds, the algorithm must be light-weight and extremely fast. In this study, the runtime has shown that in order for this environment to become more widespread, smarter algorithms need to be developed.

6. References

[1] Gould, S. Superpixel Graph Label Transfer Project, accessed 2014. http://drwn.anu.edu.au/drwnProjNNGraph.html

[2] Kovesi, Peter. Image Segmentation using SLIC Superpixels and DBSCAN Clustering, accessed 2014. http://www.peterkovesi.com/projects/segmentation/

[3] Zhang, Chunsun. Urban Feature Extraction, accessed 2014. http://www.crcsi.com.au/Research/2-Feature-Extraction/2-02-Feature-Extraction

[4] Kantorov, Vadim. Efficient feature extraction, encoding and classification for action recognition, accessed 2014. http://www.di.ens.fr/willow/research/fastvideofeat/

[5] Genton, Marc G. Classes of Kernels for Machine Learning, accessed 2014. http://crsouza.blogspot.com/2010/03/kernel-functions-for-machine-learning.html

[6] Lowe, David. Keypoint Detector, accessed 2014. http://www.cs.ubc.ca/~lowe/keypoints/