English Digit Recognition

Using HOG features and SVM in Matlab

Ariyan Zarei ▸ 3/15/2017

Abstract

This is the report for the sixth project of the Image Processing Course (2nd Semester of the 95-96) by Dr. Alireza Tavakoli. We implemented a Digit Recognition system using SVM and HOG features in Matlab. The system is a pretty easy example of recognition problem. At the end we examined the effect of applying different HOG window sizes, on the recognition accuracy. You can find the results at the Results and Conclusion section. The best window size for this image set in this project is 2x2 which yields the highest accuracy of 90% in digit recognition.

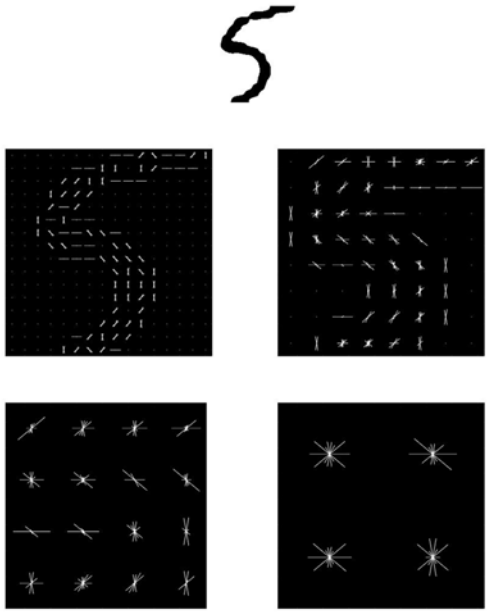

Figure 1 – Extracted HOG features for MainImage.jpg in the 1x1, 2x2, 4x4 and 8x8 HOG windows

English Digit Recognition

Using HOG features and SVM in Matlab

Introduction

This project deals with recognition of English digits placed into the simple images. The digits can either be handwritten or synthetic. As a recognition system, we used SVM or support vector machines and for having feature vectors instead of recognizing the raw images, we took advantage of HOG vectors. In this project we implemented 3 methods in Matlab. One for generating a SVM classifier, one for evaluating the classifier and generating Confusion Matrix and another for testing the classifier for single input images. You can see explanation for each of these functions in the following sections.

SVM Classifier Generator

This method, deals with the important task of generating a SVM classifier based on the input images and their labels. It takes a directory for the input images and a simple number which represents the HOG window size. It should be mentioned that, the input image set should be in correct folder formatting, which is like having 10 folders for the 10 digits and as many as images inside each folder. This method goes through the below steps in order to generate the classifier:

- Defining the Feature and Label Matrices
- For each input image
    - Running a preprocess method in order to transform the images into binary images.
    - Extracting the HOG feature using 'extractHOGfeatures' method of Matlab.
    - Adding the acquired informations into the Feature and Label Matrices
- Generating the SVM classifier using the 'fitcecoc' method.

Evaluating the Classifier

This method, as it can be perceived from its title, evaluates the generated SVM classifier. It simply takes a test image set directory, a classifier to evaluate and a HOG window size. The HOG window size in this step has to be the same size as the window size used for generating the classifier, otherwise the method fails to evaluate the classifier.

This method, then finds the feature and label matrices for the test images, by performing the same procedure as the procedure of the classifier generator method. At the end it compares the predicted labels, acquired by 'predict' function of Matlab, with the labels of the test images and produces a confusion matrix as the returning results of the function. By scrutinizing the confusion matrix, we can evaluate the SVM classifier performance in recognition of digits.

Digit OCR Predict Method

This simple method just takes an image containing a digit and a classifier, and predicts that digit based on the classifiers guess. It is implemented for humans testing the classifier.

Results and Conclusion

In order to test the aforementioned methods, we used digit image set of the Matlab. We generated the SVM classifier using the handwritten images in the set with different window sizes for HOG feature extractor. The sizes are 1x1, 2x2, 4x4 and 8x8. We then used the evaluator method, for each window size and acquired the confusion matrix for each of them. Based on the confusion matrices, the best window size for our image set is 2x2 which yields the highest accuracy of 90%. You can find the confusion matrices in the files 'ConfMatrix_1x1.csv', 'ConfMatrix_2x2.csv', 'ConfMatrix_4x4.csv' and 'ConfMatrix_8x8.csv' and also you can find all of them all together in the 'result.xlsc' file with proper formatting. You can see the confusion matrices with their accuracy of recognition for each window size below.

1x1 HOG

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	71	0	0	22	1	0	0	0	0	7
1	0	101	0	0	0	0	0	0	0	0
2	0	0	97	1	0	0	0	0	3	0
3	0	0	0	81	0	13	0	0	0	7
4	0	0	0	0	99	0	2	0	0	0
5	0	0	0	2	0	99	0	0	0	0
6	52	0	0	6	0	29	14	0	0	0
7	0	0	7	0	0	0	0	92	2	0
8	0	0	0	3	0	0	0	0	31	67
9	0	0	0	8	0	26	0	0	1	66

Correct 0.7

2x2 HOG

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	97	0	0	2	0	0	0	0	0	2
1	0	101	0	0	0	0	0	0	0	0
2	0	0	101	0	0	0	0	0	0	0
3	0	0	0	98	0	0	0	0	3	0
4	0	0	0	0	101	0	0	0	0	0
5	0	0	0	0	0	101	0	0	0	0
6	3	0	0	0	0	28	67	0	3	0
7	0	0	3	0	0	0	0	53	45	0
8	0	0	0	0	0	0	0	0	101	0
9	4	0	0	39	0	1	0	0	0	57

Correct 0.9

4x4 HOG

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	92	0	0	0	0	0	0	0	9	0
1	0	75	0	0	24	0	0	0	2	0
2	0	0	101	0	0	0	0	0	0	0
3	0	0	0	10	0	0	0	0	84	7
4	0	0	0	0	101	0	0	0	0	0
5	0	0	0	0	0	94	0	0	0	7
6	8	0	0	0	0	17	67	0	9	0
7	0	0	0	0	0	0	0	93	8	0
8	0	0	0	0	0	0	0	0	101	0
9	0	0	0	0	0	0	0	0	10	91

Correct 0.8

8x8 HOG

Actual \ Predicted	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	101	0
1	0	4	0	0	83	0	0	0	14	0
2	0	0	0	16	0	0	0	0	85	0
3	0	0	0	0	0	0	0	0	101	0
4	0	0	0	0	83	0	0	0	18	0
5	0	0	0	4	0	0	0	0	97	0
6	0	0	0	0	0	0	4	0	97	0
7	0	0	0	0	0	0	0	101	0	0
8	0	0	0	0	0	0	0	0	101	0
9	0	0	0	1	0	0	0	0	100	0

Correct 0.3