Handwritten Digit Recognition Using Support Vector Machine

By Ji-won Choi, Kyle Wensell, and Ludvik Alkhoury

I. INTRODUCTION

Handwritten digit recognition problem is addressed by machine learning algorithms that learn based on unique features from each digit and assign it to one of the 10 classes that correspond to digits from 0 to 9. In this paper we employed a multi-class SVM to classify a set of handwritten digits. The "Histogram of Oriented Gradients" (HOG) features were extracted from each digit, and used as feature vectors. As a conclusion, SVM method that uses HOG features is a valid method to classify handwritten digits.

II. METHODS

A. Histogram of Oriented Gradients (HOG)

The Histogram of Oriented Gradients (HOG) is a method proposed by Dalal et al. [1] that aims to describe an image by a set of local histograms. These histograms count occurrences of gradient orientation in a local part of the image [2]. At each pixel, the gradient magnitude and the direction is computed. The Histogram of Oriented Gradients shows the direction of the gradient vector (on the x-axis), and the magnitude of the gradient vector (on the y-axis). The x-axis is divided into "m" bins. Usually, HOG utilizes 9 bins to represent all possible directions/angles ([0^o-20^o], [20^o-40^o], [40^o-60^o], [60^o-80^o], [80^o-100^o], [100^o-120^o], [120^o-140^o], [140^o-160^o], [160^o-180^o]). The HOG is calculated for each block of $n \times n$ pixels and all calculated diagrams (for the whole image) are concatenated to form the feature vector. A HOG descriptors calculation example is shown in figure 1.

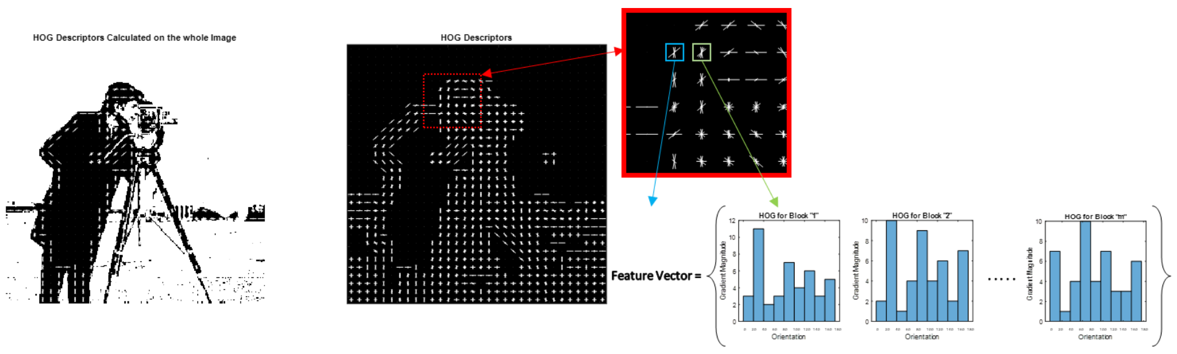


Figure 1. illustration of HOG feature vector calculation

Unlike edge detection algorithms, HOG descriptor provides more information on the image features. For example, edge detection algorithms can detect a circular shape but cannot differentiate between a coin and a button. By extracting unique features from an image, HOG can tell the difference between two circular objects.

B. Support Vector Machine (SVM)

The Support Vector Machine (SVM) is a linear classifier that can be viewed as an extension of the Perceptron developed by Rosenblatt in 1958. The Perceptron finds a hyperplane (if it exists) to separate data points of a given dataset. The SVM finds the maximum margin separating hyperplane. In other words, SVM maximizes the distance to the closest data points from both classes [4]. The SVM optimization problem is formulated as follows:

$$min_{w,b} \; ||\boldsymbol{w}||$$

$$\text{Subject to : } y_i \; (w_i . x_i + b) \geq 1$$

$$\text{For all i: } i = 1, \dots, n$$

Where w and b are the parameters of the hyperplane separating the two classes. The formulated minimization problem can be solved by applying simple convex optimization techniques, giving SVM advantage on other classification algorithms. We employed the *one-versus-the-rest* approach. In this approach trains K separate binary classifiers, with the kth classifier operating on the examples relative to class C_k against the examples from all other classes [5].

C. Kernel Method

Kernel Methods use Kernel functions to transform a set of inseparable data into another space where data are clearly separated. Kernel functions are widely used in SVM since for most of the time, data-points are not linearly separable. In this paper, we used three different Kernel functions; (1) Linear Kernel, (2) Quadratic Kernel, and (3) Gaussian Kernel.

$$\varphi(x) = [1 \quad \boldsymbol{x}] \tag{1}$$

$$\varphi(x) = [1 \quad \boldsymbol{x} \quad \boldsymbol{x}^2] \tag{2}$$

$$\varphi(x) = \left[1 \quad \frac{1}{\sqrt{2\pi}} e^{\frac{-\boldsymbol{x}^2}{2}}\right] \tag{3}$$

D. Additive Noise in Testing Set

It is possible to simulate the effects of noisy test data by adding noise to the handwritten images. We decided to test the effects of two different types of additive noise: Salt-and-Pepper (henceforth SNP) noise, and Gaussian noise. The main difference between the two is the type of pixel added to the image. For SNP noise, the intensity of the pixels added to the greyscale image are binary, i.e., black (0 intensity) or white (1 intensity) pixels, assuming an intensity scale that is normalized to be between 0 and 1. On the other hand, Gaussian noise can take any value of intensity, taken from a Gaussian distribution. The intensity of the pixel that the noise will be added to is taken as the mean of the Gaussian distribution, thus ensuring the added noise will only be a marginal amount for a small given noise variance. Figure 2 illustrates the difference between the two noise styles on an image from our dataset.

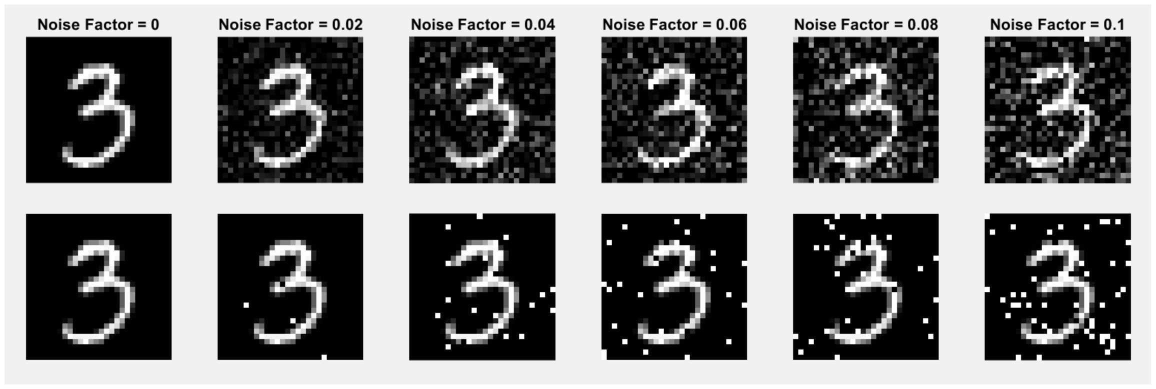


Figure 2. two styles of noise added to handwritten digit from data set

The parameter that defines the amount of noise is slightly different for the two styles. SNP noise has what is known as *noise density*, also called *flip probability*. This defines the probability for each pixel to be exchanged for a salt-or-pepper noise pixel, and thus, also defines the relative amount of noise added to the image as a proportion. Gaussian noise, as mentioned, instead relies on the *variance* of the noise distribution; a higher variance results in a wider range of possible noise values added, thus making the image more and more obscured. The first row and the second rows of figure 2 illustrate the contamination of an image with Gaussian and SNP noise, respectively. We expect this difference in noise distribution to have slightly different effects on the SVM's success, and thus, we felt it would be useful to test both, to analyze the robustness of our chosen algorithm.

III. TESTED SCENARIOS

We conducted three experiments to test SVM under three different scenarios:

1- Experiment 1: We evaluated the performance of SVM for three kernels (Linear, Quadratic, and Gaussian kernel). The HOG is calculated for each block of $n \times n$ pixels and all calculated diagrams (for the whole image) are concatenated to form the feature vector. In this experiment, the testing set is not contaminated with noise. We used a block of 6×6 pixels for this experiment. The results in this case provide an upper bound for the expected performance of the SVM for the three abovementioned kernels.

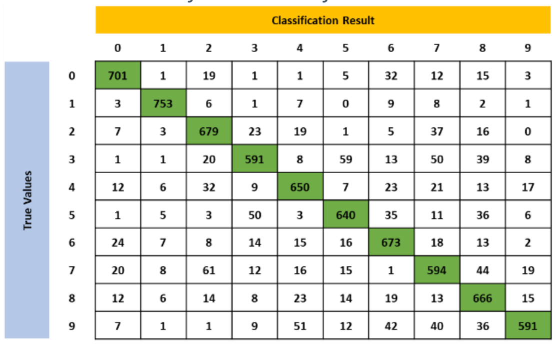
2- Experiment 2: We tested the performance of SVM for Linear, Quadratic, and Gaussian kernel, when HOG is calculated from a block of 4×4 and 6×6 pixels, respectively. We tested this scenario with a clean testing set. In this case, we compared the performance of SVM for different block size.

3- Experiment 3: We ran SVM for three kernels (Linear, quadratic, and Gaussian) on a testing set contaminated with either Salt-and-Pepper or Gaussian noise. We calculated HOG for 6×6 pixels blocks. In this experiment, we intended to see the effect of noise on the accuracy of SVM.

IV. RESULTS

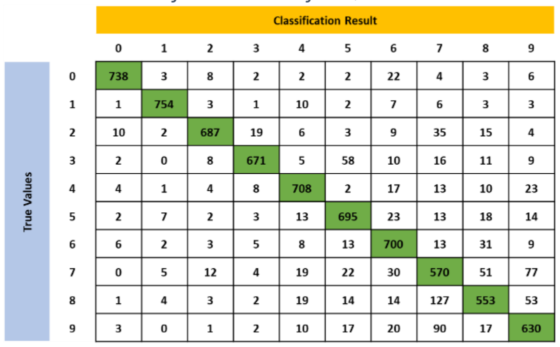
We train SVM with a training set of 1000 images (100 images for each digit from 0 to 9). The testing set consists of 7900 images (790 images for each digits from 0 to 9). We presented our results as; (1) Confusion Matrices, and (2) success rate tables. In a Confusion Matrix, rows and columns represent the "true values/labels" and "classified results", respectively. The diagonal shows the frequency of correct digit classification. Table 1, 2, and 3 are the Confusion Matrices for the classification results of the first experiment for Linear, Quadratic, and Gaussian kernel, respectively. Additionally, table 4 encloses the success rate for digit classification for each of the latter kernels.

Table 1. Confusion Matrix for Linear Kernel



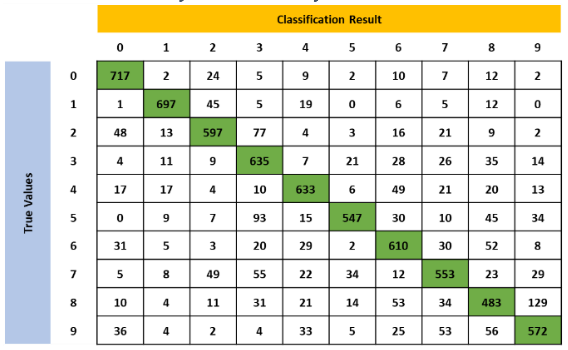
		Classification Result									
		0	1	2	3	4	5	6	7	8	9
True Values	0	701	1	19	1	1	5	32	12	15	3
	1	3	753	6	1	7	0	9	8	2	1
	2	7	3	679	23	19	1	5	37	16	0
	3	1	1	20	591	8	59	13	50	39	8
	4	12	6	32	9	650	7	23	21	13	17
	5	1	5	3	50	3	640	35	11	36	6
	6	24	7	8	14	15	16	673	18	13	2
	7	20	8	61	12	16	15	1	594	44	19
	8	12	6	14	8	23	14	19	13	666	15
	9	7	1	1	9	51	12	42	40	36	591

Table 2. Confusion Matric for Quadratic Kernel



		Classification Result									
		0	1	2	3	4	5	6	7	8	9
True Values	0	738	3	8	2	2	2	22	4	3	6
	1	1	754	3	1	10	2	7	6	3	3
	2	10	2	687	19	6	3	9	35	15	4
	3	2	0	8	671	5	58	10	16	11	9
	4	4	1	4	8	708	2	17	13	10	23
	5	2	7	2	3	13	695	23	13	18	14
	6	6	2	3	5	8	13	700	13	31	9
	7	0	5	12	4	19	22	30	570	51	77
	8	1	4	3	2	19	14	14	127	553	53
	9	3	0	1	2	10	17	20	90	17	630

Table 3. Confusion Matrix for Gaussian Kernel



True Values	Classification Result: 0	1	2	3	4	5	6	7	8	9
0	717	2	24	5	9	2	10	7	12	2
1	1	697	45	5	19	0	6	5	12	0
2	48	13	597	77	4	3	16	21	9	2
3	4	11	9	635	7	21	28	26	35	14
4	17	17	4	10	633	6	49	21	20	13
5	0	9	7	93	15	547	30	10	45	34
6	31	5	3	20	29	2	610	30	52	8
7	5	8	49	55	22	34	12	553	23	29
8	10	4	11	31	21	14	53	34	483	129
9	36	4	2	4	33	5	25	53	56	572

Table 4. Success Rate of Linear, Quadratic, and Gaussian Kernel

	Success Rate (in %)
Linear Kernel	82.76%
Quadratic Kernel	84.89%
Gaussian Kernel	76.51%

The Confusion Matrices (table 1, 2, and 3) show that for all the three kernels, a small number of digits was misclassified. For example, in the case of a Linear kernel, 717 out of 790 “zero” digits were correctly classified as a “zero”. Moreover, the Quadratic kernel with a success rate of 84.89% outperforms the Linear (with an 82.76% success rate) and the Gaussian (with a 76.51% success rate) kernel. As a conclusion, when HOG features were calculated on blocks of 6×6 pixels, the SVM with a Quadratic kernel is the most accurate method in classifying the handwritten digits.

For the second experiment, the comparison of the performance of SVM (rate of successfully classify a digit correctly) for Linear, Quadratic, and Gaussian kernel, when HOG features were calculated from 4×4 and 6×6 blocks, is shown in table 5.

Table 5. Success Rate of Linear, Quadratic, and Gaussian Kernel, when HOG was calculated from 4x4 and 6x6 blocks

	Success Rate (in %)	
	6×6 block	4×4 block
Linear Kernel	82.76%	86.24%
Quadratic Kernel	84.89%	80.90%
Gaussian Kernel	76.51%	

Table 5 shows that the success rate became higher for the three kernels, when HOG features were extracted from blocks of smaller size. HOG feature vectors have 324 features for 4×4 blocks, and 1296 for 6×6 blocks. Therefore, it is more computationally complex to train the classifier using a larger block due to the larger number of HOG features. According to table 5, the success rate has not improved dramatically even with a larger feature vectors. Hence, we suggested the use of SVM with a 6×6 block.

On the third experiment, we contaminated the testing data with SNP and Gaussian noise of increasing intensity. Table 6 and 7 enclose the success rate of noise contaminated images classification for the three kernels (Linear, Quadratic, and Gaussian). HOG features were calculated on 6×6 blocks.

Table 6. Success Rate for SNP noise contaminated testing set

	Success Rate (in %)		
	Linear Kernel	Quadratic Kernel	Gaussian Kernel
1% SNP	66.51%	70.61%	59.78%
2% SNP	56.85%	59.15%	49.95%
5% SNP	38.20%	41.73%	37.65%
10% SNP	26.86%	29.91%	31.61%

Table 7. Success Rate for Gaussian noise contaminated testing set

	Success Rate (in %)		
	Linear Kernel	Quadratic Kernel	Gaussian Kernel
2% Gaussian	17.11%	34.27%	48.78%
5% Gaussian	14.89%	29.97%	50.86%
10% Gaussian	13.39%	25.85%	49.39%

SNP and Gaussian noise has a very high impact on digits' classification using SVM method. The introduction of noise was dramatically deteriorated the performance of SVM for all kernels. Even a very small amount of noise was capable of causing an extensive decrease in the success rate. For example, Quadratic kernel had a success rate of 84.89% before any noise addition. When 1% and 2% of the image was contaminated with SNP and Gaussian noise, respectively, the success rate for the Quadratic Kernel has dropped down to 70.61% (for SNP noise) and 34.27% (for Gaussian noise). Conclusively, SVM method is tremendously sensitive to noise.

V. CONCLUSION AND DISCUSSION

In this paper, we employed SVM to classify handwritten digits into 10 classes (from 0 to 9). We adopted the HOG descriptor to extract unique features from each digit. We tested and analyzed the output of our SVM classifier by inputting HOG feature vectors generated from different block sizes, and found that the larger feature vector size leads to an increase in computation time. However, this increase in computational complexity does not guarantee an increase in success rate, as shown in table 5. The success rate for the randomly-chosen data set we used for testing was dependent on the kernel used for training. For the 6x6 block size, the Quadratic kernel has given a higher success rate as compared to the Linear and Gaussian kernel, but the opposite is true for the 4x4 block size. Furthermore, we added two types of noise to the testing set: (1) Salt-and-Pepper noise, and (2) Gaussian noise. It was clear that when noise was added, the success rate has tremendously decreased, divulging the sensitivity of SVM to noise. Surprisingly, for the Gaussian noise, the Gaussian kernel does not experience the same decrease in performance that the other kernels do. In this paper, we have used a relatively small training set (1000 data points). The increase in training set size may improve the success rate of the SVM classifier. Additionally, a larger test set may show a more complete picture of the success rates of each method. Overall, we have proven the validity of using HOG vectors for handwritten digit classification, and shown that the relative accuracy of the method is maintained even when using a 6x6 block size to generate a more manageable feature vector.

VI. REFERENCES

[1] Dalal, N., & Triggs, B. (2005, June). Histograms of oriented gradients for human detection. In International Conference on computer vision & Pattern Recognition (CVPR'05) (Vol. 1, pp. 886-893). IEEE Computer Society.

[2] Suard, F., Rakotomamonjy, A., Bensrhair, A., & Broggi, A. (2006, June). Pedestrian detection using infrared images and histograms of oriented gradients. In 2006 IEEE Intelligent Vehicles Symposium (pp. 206-212). IEEE.

[3] Jacobs, D. (2005). Image gradients. Class Notes for CMSC, 426.

[4] http://www.cs.cornell.edu/courses/cs4780/2017sp/lectures/lecturenote09.html

[5] Simeone, O. (2018). A brief introduction to machine learning for engineers. *Foundations and Trends® in Signal Processing*, *12*(3-4), 200-431.