SVM Classifier for Recognition of Handwritten Devanagari Numeral

Mahesh Jangid, Renu Dhir, Rajneesh Rani, Kartar Singh
Department of Computer Science and Engineering
Dr. B R Ambedkar National Institute of Technology
Jalandhar, India
mahesh_seelak@yahoo.co.in, dhirr@nitj.ac.in, ranir@nitj.ac.in, kartarster@gmail.com

Abstract—**In this manuscript we recognize the handwritten Devnagari numerals. In our implementation we have used density and background directional distribution features for the zones, in which we divided the numeral samples already. We used the normalized images of samples of varying sizes of 32*32, 40*40 and 48*48. We divided these normalized images into 4*4 (16), 5*5 (25) and 6*6 (36) zones respectively to compute the features for each zone. In all the cases each zone is of size 8*8 pixels. Each zone contains 9 features consisting of one density feature and 8 backgrounds directional distribution features. The zonal density feature is computed by dividing the number of foreground pixels in each zone by total number of pixels in the zone i.e. 64. The other 8 features are based on directional distribution values of background in eight directions. These directional values are computed for each foreground pixel by summing up the value corresponding to neighbouring background pixels given in the specific mask for each direction. For each direction these directional distribution features are summed up for all pixels in each zone. Thus numbers of features finally used for recognition are 144, 225 and 324 for samples of respective sizes in increasing order. For classification purpose we have used SVM classifier with RBF kernel. Our dataset of handwritten Devnagari numerals used is provided by Indian Statistical Institute (ISI), Kolkata. Training data size is 18783 and testing data size is 3763, totally 22546. The optimum 5-fold cross validation accuracies of training data obtained for varying sizes of samples in increasing order are 98.76%, 98.91% and 98.94% respectively. By observing the cross validation results it is conclusive that at the cost of increasing the features size there is only minute increases in the performance. So we recommend 144 sized feature vector to recognize testing samples. The testing accuracy by using 144 features for 32*32 normalized samples observed is 98.51% which is prominent and cost-efficient.**

Keywords- Devanagari Numerals, Zoning density, Background directional distribution, SVM classifier

I. INTRODUCTION

Optical Character Recognition (OCR) is a field of research in pattern recognition, artificial intelligence and machine vision. OCR is a mechanism to convert machine printed or handwritten document file into editable text format. This field is broadly divided into two Online and offline character recognition. Off-line Character recognition further divided into two machine printed and handwritten character recognition. In handwritten Character Recognition, there are lots of problems as compare to machine printed document because of the different peoples have different writing styles, the size of pen-tip and some people have skewness in their writing. All this challenges make the researches to solve the problems.

Devnagari Script is an oldest one that is used to write many languages such as Hindi, Nepali, Marathi, Sindhi and Sanskrit where Hindi is the third most popular language in the world and it is the national language of the India [1]. 300 million people use the Devnagari Script for documentation in central and northern parts of India [2].

First research report on handwritten Devanagari Characters was published in 1977[3] after which many researchers have done the work on the Devanagari Script with different feature extraction algorithms and different classifiers. G S Lehal and Nivedan Bhatt [4] have proposed a contour extraction technique and obtained 89% accuracy. Reena Bajaj et al. [5] have employed three different kinds of feature namely, density features, moment features and descriptive features for classification of Devanagari Numerals and obtained 89.68% accuracy. R J Ramteke et al [6] have proposed a method based on invariant moments and the divisions of image for the recognition of numerals and achieved 92% accuracy. U. Bhattacharya et al. [7] have used a combination of ANN (Artificial Neural Network) and HMM (Hidden Markov Model) classifier on 16273 samples of Handwritten Devanagari Numerals and obtained 95.64% accuracy. Pal et al. [8] used 23,340 handwritten Numerals samples and obtained 98.41 % recognition rate by getting the gradient feature and applied MQDF classifiers. N Sharma et al. [9] have proposed a quadratic qualifier based technique and used 22546 samples for his experiment and achieved 98.86% accuracy.

II. DATABASE

The database is provided by the ISI (Indian Statistical Institute, Kolkata) [14]. Initially Devanagari script was developed to write Sanskrit but was later adapted to write many other languages such as Hindi Marathi and Nepali. The printed Devanagari Numerals are shown in figure 1 and it is seen that there are variations in the shapes of numerals 5, 8 and 9 in their printed forms. In figure 2, there are shown the samples of the Handwritten Devanagari Numerals database. The distributions of training data and testing data are shown in table 1.

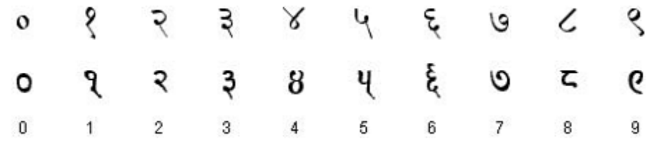


Figure 1: Devanagari Numerals

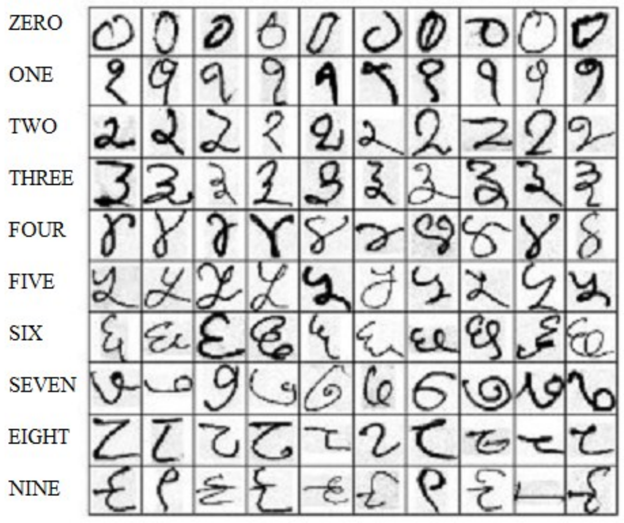


Figure 2: Handwritten Devanagari Numerals Samples

Table 1: Distribution of numerals in Devanagari Database

Digits	Training Set	Test Set	Total
0	1843	369	2212
1	1891	378	2269
2	1891	378	2269
3	1882	377	2259
4	1876	376	2252
5	1889	378	2267
6	1869	374	2243
7	1869	378	2247
8	1887	377	2264
9	1886	378	2264
	18783	3763	22546

III. PROPOSED METHODOLOGY

In our proposed methodology of handwritten Devanagari Numerals recognition we have considered 10 basic classes of Devanagari Numerals for our experiment. We have used zoning density and background directional distribution to extract total of 144, 225 and 324 features for varying normalized sizes of samples. In following sections we will study our proposed method in details.

A. Preprocessing

Originally entire handwritten dataset is in the form of isolated image of each sample. All images are in gray scale format. This standard dataset is obtained from ISI, Kolkata.

i. We converted the image into binary image by choosing Otsu gray threshold value + 0.1.

ii. We removed all isolated components (objects) that have fewer than 30 pixels

iii. We applied median filtering, a nonlinear operation often used in image processing to reduce "salt and pepper" noise

iv. We normalized the image into 32*32, 40*40 and 48*48 sizes for each of three cases of our experiment.

B. Feature Extraction

Two types of features for our experiment- *zoning density* (ZD) and *Background Directional Distribution* (BDD) have been used.

1) Zoning Density (ZD) Features

We have created 16 (4*4) zones of our 32*32 sized sample. By dividing the number of foreground pixels in each zone by total number of pixels in each zone i.e. 64 we obtained the density of each zone. Thus we obtained 16 zoning density features. (Fig.3)

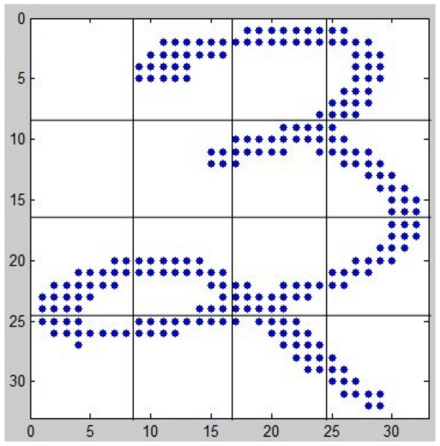


Fig. 3(a): 16 zones of 32*32 normalized Devnagari numeral '3'.

2) Background Directional Distribution (BDD) Features

For these features we have considered the directional distribution of neighboring background pixels to foreground pixels. We computed 8 directional distribution features. To calculate directional distribution values of background pixels for each foreground pixel, we have used following masks for each directional values (Fig.4). The pixel at center 'X' is foreground pixel under consideration to calculate directional distribution values of background.

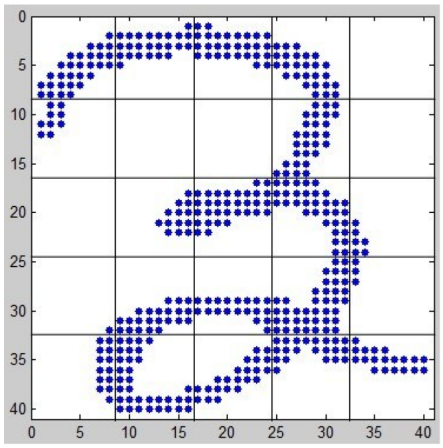


Fig. 3(b): 25 zones of 40*40 normalized Devnagari numeral '3'.

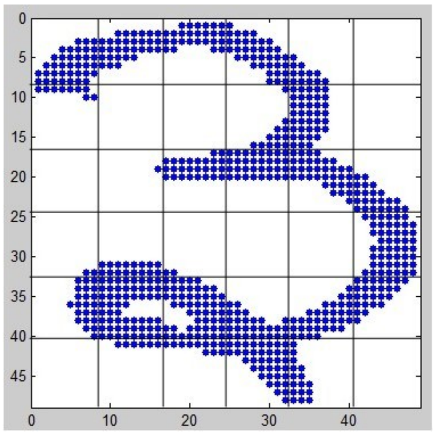
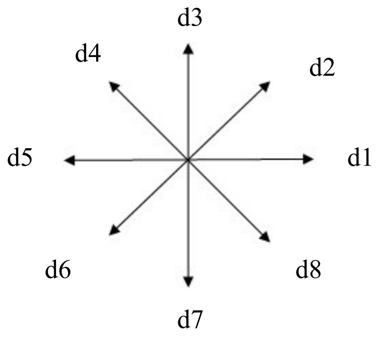
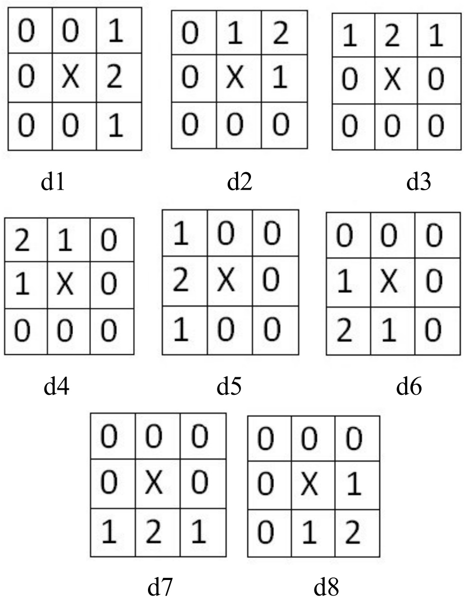


Fig. 3(c): 36 zones of 48*48 normalized Devnagari numeral '3'.



(a)



0	0	1
0	X	2
0	0	1

d1

0	1	2
0	X	1
0	0	0

d2

1	2	1
0	X	0
0	0	0

d3

2	1	0
1	X	0
0	0	0

d4

1	0	0
2	X	0
1	0	0

d5

0	0	0
1	X	0
2	1	0

d6

0	0	0
0	X	0
1	2	1

d7

0	0	0
0	X	1
0	1	2

d8

(b)

Fig.4. (a) 8 directions used to compute directional distribution, (b) Masks used to compute directional distribution in different directions.

To compute directional distribution value for foreground pixel 'X' in direction d1, for example, the corresponding mask values of neighboring background pixels will be added. Similarly we obtained all directional distribution values for each foreground pixel. Then, we summed up all similar directional distribution values for all pixels in each zone, described earlier in zoning density features description. Thus we finally computed 8 directional distribution feature values for each zone.

We combined both types of features extracted from zoning density and directional distribution. SVM classifier uses these features for classification.

IV. CLASSIFICATION

Support Vector Machine is supervised Machine Learning technique. It is primarily a two class classifier. Width of the margin between the classes is the optimization criterion, i.e. the empty area around the decision boundary defined by the distance to the nearest training pattern. These patterns called support vectors, finally define the classification function.

All the experiments are done on LIBSVM 3.0.1[12, 13] which is multiclass SVM and select RBF (Redial Basis Function) kernel. A feature vector set fv(x_i) i=1…m, where m is the total number of character in training set and a class set cs(y_j) j=1…m , cs(y_j) ϵ { 0 1 ….9} which defines the class of the training set, fed to Multi Class SVM.

LIBSVM implements the "one against one" approach (Knerr et al .., 1990) [10] for multi-class classification. Some early works of applying this strategy to SVM include, for example, Kressel (1998) [11]. If k is the number of classes, then k (k-1)/2 classifiers are constructed and each one trains data from two classes. For training data from the ith and jth

classes, we solve the following two class classification problem:

In classification we use a voting strategy: each binary classification is considered to be a voting where votes can be cast for all data points x - in the end a point is designated to be in a class with the maximum number of votes.

$$\min_{w^{ij}, b^{ij}, \xi^{ij}} \quad \frac{1}{2}(w^{ij})^T w^{ij} + c\sum_t (\xi^{ij})_t$$

$$\text{subject to } (w^{ij})^T \phi(x_t) + b^{ij} \geq 1 - \xi_t^{ij},$$

$$\text{if } x_t \text{ in the } i^{th} \text{ class}$$

$$(w^{ij})^T \phi(x_t) + b^{ij} \leq -1 + \xi_t^{ij},$$

$$\text{if } x_t \text{ in the } j^{th} \text{ class},$$

$$\xi_t^{ij} \geq 0.$$

In case that two classes have identical votes, though it may not be a good strategy, now we simply choose the class appearing first in the array of storing class names.

LIBSVM is used with Radial Basis Function (RBF) kernel, a popular, general-purpose yet powerful kernel, denoted as

$$K(x_i, x_j) \equiv \exp(-\gamma\|x_i - x_j\|^2)$$

C is the cost parameter of SVM and γ is RBF kernel parameter, which need to refine to get optimal results.

V. Experiments and Results

We have experimented with three sets of feature vectors. These feature vectors are of sizes 144, 225 and 324 while using sample images of size 32*32, 40*40 and 48*48 respectively. These sample images are divided into 4*4 (16), 5*5 (25) and 6*6 (36) zones. The size of each zone is 8*8 pixels. Each zone produces 8 background directional features and one density feature. The recognition results of our test data with different normalized size of images are shown in table 2. X-coordinate represents image size and Y-coordinate represents the value of RBF kernel parameter **γ.** The results do not vary significantly by changing cost parameter C. Our optimized results are obtained with range of 8 to 12 values of C. With larger values of C the results decrease slightly. The out of range of values of γ are also tested, but lower recognition rate is observed, with one exception of higher accuracy of 97.16% in case of 48*48 normalized image at γ = 0.08.

The confusion matrix of our optimized test result with accuracy 98.62% is shown in table 3.

Table 2. Performance variation with image size and gamma value

Image Size/ gamma (γ)	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**	**0.8**	**0.9**	**1**
32*32 (144)	98.41	98.41	98.54	98.62	98.38	98.27	98.14	97.98	97.90	97.74
40*40 (225)	98.46	98.4	98.30	98.14	98.01	97.79	97.69	97.56	97.56	96.70
48*48 (324)	97.10	97.08	97.05	96.92	96.70	96.60	95.83	94.53	92.51	90.38

Table 3. Confusion Matrix of recognition of 32*32 test samples with γ = 0.4

Numeral	**No. of samples**	**Classified or misclassified as**										**Accuracy (%)**
		૦ (0)	૧(1)	૨ (2)	૩ (3)	૪ (4)	૫ (5)	૬ (6)	૭ (7)	૮ (8)	૯ (9)	
૦ (0)	369	**368**	1	0	0	0	0	0	0	0	0	99.73
૧ (1)	378	0	**370**	2	0	1	2	0	0	2	1	97.88
૨ (2)	378	0	1	**372**	1	0	1	0	0	2	1	98.41
૩ (3)	377	0	0	3	**371**	0	0	3	0	0	0	98.41
૪ (4)	376	1	0	0	1	**374**	0	0	0	0	0	99.47
૫ (5)	378	0	0	1	1	5	**370**	1	0	0	0	97.88
૬ (6)	374	0	0	1	2	0	0	**368**	0	0	3	98.40
૭ (7)	378	4	0	0	0	0	0	3	**371**	0	0	98.15
૮ (8)	377	0	0	2	0	0	1	0	0	**373**	1	98.94
૯ (9)	378	0	0	3	0	0	0	1	0	0	**374**	98.94
Average Test Accuracy												**98.62**

We also performed 5-fold cross validation on whole dataset of size 22546 including training and testing data. Contradictory to our test data results in cross validation performance is observed to be increased with increase in features (or image size) but it is not observed increasing in the test results similarly. To conclude these optimized results either in testing or cross validation, we observed all results by varying the values of C and γ parameters and optimized results are refined. The optimized cross validation results we observed are listed in table 4 for varying size of features.

Thus we got optimum test result as 98.62% with 144 features (32*32 size) while optimum cross validation result as 99.05% accurate with 324 features (40*40 size). Due to increase in size and hence number of features, the computing complexity increases enormously. Hence we have found and recommend the 32*32 normalized size of samples with 144 features having relevant efficiency in both perspectives of good accuracy and less time consuming.

Table 4: 5-fold Cross validation results of whole dataset

S.No.	**Sample Size**	**No. of Features**	**Cross validation Accuracy**	**Classifier parameters**
1.	32*32	144	98.94%	C=12, γ=0.35
2.	40*40	225	98.99%	C=12, γ=0.13
3.	48*48	324	99.08%	C=12,γ=0.12

In the Table 5 the comparison of our proposed method with some earlier approaches is presented.

Table 5: Comparison of accuracy obtained by different methods

S.No	**Method proposed by**	**Data Size**	**Accuracy Obtained**
1	R. Bajaj et al [5]	400	89.6 %
2	R. J. Ramteke et al [6]	169	92.68 %
3	U. Bhattacharya et al. [7]	16274	95.64 %
4	U.Pal et al [8]	23,340	98.41 %
5	N. Sharma et al. [9]	22,546	98.86 %
6	Proposed System	22,546	99.08 %

CONCLUSION

In this paper a feature extraction algorithm has been proposed to recognize handwritten Devanagari Numerals. The results for recognition of test data and 5 fold cross validation of whole dataset are observed. 32*32 normalized size of samples producing 144 features is efficient producing 98.62% test recognition rate and 98.94% cross validation accuracy. For improvement multiple classifiers can be combined with additive recognition rate with diverse features.

REFERENCES

[1] U.Pal and B B Choudhuri, "Indian script character recognition: A survey" Pattern Recognition ,Vol 37,pp 1887-1899,2004

[2] R M K Sinha, " A journey from Indian scripts processing to Indian language processing ", IEEE Ann. Hist. Computer, vol 31, no 1, pp 831, 2009

[3] I.K, Sethi and B. Chatterjee, "Machine Recognition of constrained Hand printed Devanagari", Pattern Recognition, Vol. 9,pp. 69-75, 1977.

[4] G S Lehal, Nivedan Bhatt, "A Recognition System for Devnagri and English Handwritten Numerals", Proc. Of ICMI, 2000.

[5] Reena Bajaj, Lipika Day, Santanu Chaudhari, "Devanagari Numeral Recognition by Combining Decision of Multiple Connectionist Classifiers", Sadhana, Vol.27, Part-I, 59-72, 2002.

[6] R.J.Ramteke, S.C.Mehrotra, "Recognition Handwritten Devanagari Numerals", International journal of Computer processing of Oriental languages, 2008.

[7] U. Bhattacharya, S. K. Parui, B. Shaw, K. Bhattacharya, "Neural Combination of ANN and HMM for Handwritten Devnagari Numeral Recognition".

[8] U.Pal, R.K.Roy and F. Kimura,"Indian muilti-script full pin-code string recognition for postal automation" in Proc. 10 th conf. Document Analysis Recognition, 2009, pp 456-460

[9] U. Pal, T. Wakabayashi, N. Sharma and F. Kimura, "Handwritten Numeral Recognition of Six Popular Indian Scripts", Proc. 9th ICDAR, Curitiba, Brazil, Vol.2 (2007), 749-753.

[10] S. Knerr, L. Personnaz, and G. Dreyfus. Single-layer learning revisited: a stepwise procedure for building and training a neural network. In J. Fogelman, editor, Neu-rocomputing: Algorithms, Architectures and Applications. Springer-Verlag, 1990.

[11] U. H.-G. Kressel. Pairwise classication and support vector machines. In B. Scholkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods { Support Vector Learning, pages 255{268, Cambridge, MA, 1998. MIT Press.

[12] Chih-Chung Chang and Chih-Jen Lin, LIBSVM: a library for support vector machines, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[13] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin, "A Practical Guide to Support Vector Classification", [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[14] http://www.isical.ac.in/~ujjwal/download/database.html.

[15] Gonzalez, Woods and Eddins,"a book on Digital Image Processing Using MATLAB".