Comparative Performance Analysis of Feature(S)-Classifier Combination for Devanagari Optical Character Recognition System

Jasbir Singh
Department of Computer Science
Punjabi University
Patiala, India

Gurpreet Singh Lehal
Department of Computer Science
Punjabi University
Patiala, India

***Abstract*—this paper presents a comparative performance analysis of feature(s)-classifier combination for Devanagari optical character recognition system. For performance evaluation, three classifiers namely support vector machines, artificial neural networks and k-nearest neighbors, and seven feature extraction approaches viz. profile direction codes, transition, zoning, directional distance distribution, Gabor filter, discrete cosine transform and gradient features have been used. The first four features have been used jointly as statistical features. The performance has also been evaluated by using the combination of these feature extraction approaches. In addition, performance evaluation has also been done by varying the feature vector length of Gabor and DCT features. For training the classifiers, 7000 samples of first 70 classes (out of 942 classes), recognized in the earlier work have been used. Such a large number of classes are due to the horizontal and vertical fusion/overlapping characters. We have chosen first 70 classes as their percentage contribution out of 942 classes has found to be 96.69%. For testing, 1400 samples have been collected separately. A corpus of 25 books has been used for sample collection. Classifiers trained on different features, have been compared for performance evaluation. It has been found that support vector machines trained with Gradient features provide the classification correctness of 99.429%, and there is no significant increase in the performance with the increase in the feature vector length.**

Keywords—Artificial Neural Network; DCT; Directional Distance Distribution; Feature extraction, Gabor; k-Nearest Neighbour; Profile direction codes; Support Vector Machines; Transition; Zoning

I. INTRODUCTION

Optical character recognition is a widely used technique for generating digital counterpart of printed or handwritten text. A lot of work has been done in this field, particularly from Devanagari script point of view. In one of the earlier work, Sinha and Mahabala [1] have used syntactic pattern analysis system with an embedded picture language for recognition of Devanagari script.

Bansal and Sinha[3,4], laid emphasis on the use of various knowledge sources at all levels in Devanagari document processing system. These knowledge sources are mostly statistical in nature. Chaudhuri and Pal [5] have suggested the primary grouping of characters, where each character is assigned to one of the three groups namely basic, modifier and compound character. A feature based tree classifier approach is then used for basic and modifier character recognition. As Devanagari script consists of several basic characters, half form of characters, vowel-modifiers and diacritics, therefore from character recognition point of view only 78 basic character classes are sufficient for the identification of these characters. But in Devanagari the characters fuse with each other and generate new compound characters which are very difficult to separate during segmentation phase of OCR process. These compound characters are commonly known as conjuncts.

Sinha and Bansal [2] have discussed the algorithms that can be used to segment the compound characters into its constituent symbols rather than treating the character as a single unit. But in our work we have considered these compound characters as single recognizable unit, so that the segmentation errors can be reduced. Kompalli, Nayak and Setlur [6] and Kompalli, Nayak and Govindaraju [7] have also discussed the wide range of challenges in Devanagari script including that of compound characters. These compound characters are the result of horizontal or vertical fusion of basic characters. As an example व + ् + य will form व्य and च + ् + च will result in च्च. From these two examples it is very much clear, that it is very difficult to decide from where to separate these compound characters into the constituent basic symbols and therefore treated as single recognizable unit.

The presence of conjuncts is not the only problem in segmentation but sometime the height of constituent characters in a word (to be segmented) is such that it causes segmentation problems, for example in the word समूह the height of constituent character ह is such that it can either lead to, over-segmentation of ह or under-segmentation of मू. Therefore we have also considered such character combinations (मू) as single recognizable units. It should be noted that if the constituent symbols are not connected, then they will be treated as separate recognizable units.

In order to identify the possible classes and their frequency of occurrence we [9] have used a corpus of approximately 3 million words, which comprises of Unicode data. We have identified 864 compound characters (apart from basic 78 characters), which comprise of both horizontally and vertically fused characters, and consonants with lower vowel modifiers which makes a total of 942 recognizable units. As it is very difficult to handle such a large number of classes; therefore coverage analysis has been done to optimize the character class count. The analysis has been done on the basis of their frequency of occurrence. It has found that the first 70 classes contribute to 96.69% of the overall classes, as shown in Table I. Therefore in this work we have taken the first 70 classes to find an optimal Feature(s)-classifier combination. For evaluating the performance of feature(s)-classifier combinations 1400 test samples i.e. 20 samples per class has been collected separately.

TABLE I. PERCENTAGE CONTRIBUTION OF RECOGNIZABLE UNITS

Recognizable units	% contribution
20	82.0185
30	90.1112
40	93.4336
50	95.0826
70	96.6985

This paper is organized as follows. Section II describes the various classification techniques, like Support Vector Machines, Artificial Neural Network and k-Nearest Neighbor. Section III depicts the various feature extraction methods which are used in his work. In section IV the performance of all feature-classifier combinations has been presented, and in section V comparison of all combinations has been done. Conclusion and future scope is described in the section VI.

II. CLASSIFICATION METHODS

The task of classification is to assign an input pattern represented by feature vectors to one of many pre-specified classes. Here we have used three classifiers described here.

A. Support Vector Machines (SVM)

SVM's (Support Vector Machines) [10] are a useful technique for data classification. SVM is a supervised learning classifier. A classification task usually involves separating data into training and testing sets. Each instance in the training set contains one target value (class label) and several attributes (features). The goal of SVM is to produce a model which predicts the target value. Given a training set of attributes-label pairs, $(x_i, y_i),\ i = 1, \dots, l$ where $x_i \in R^n$ and $y \in \{-1,1\}^l$, the support vector machines require the solution of the following optimization problem given by (1):

$$\min_{w,b,\xi} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \qquad (1)$$

subjected to $\quad y_i\,(w^T \phi(x_i) + b\,) \geq 1 - \xi_i\,, \qquad \xi_i \geq 0$

Here training vectors x_i are mapped into higher dimensional space by the function ϕ. C >0 is the penalty parameter of the error term. Furthermore, $K(x_i,\, x_j) \equiv \phi(x_i)^T \phi(x_j)$ is called the kernel function. Three kernel functions are listed below.

- Linear: $K(x_{i,}\, x_j) = x_i^T x_j$.
- Polynomial: $K(x_i,\, x_j) = (\gamma x_i^T x_j + r)^d, \quad \gamma > 0$.
- RBF: $K(x_i,\, x_j) = exp(-\gamma \parallel x_i - x_j \parallel^2), \quad \gamma > 0$.

Where $\gamma, r\ and\ d$ are kernel parameters.

B. Artificial Neural Network (ANN)

Artificial neural networks are the computational models that consist of number of simple processing units called neurons distributed in layers namely input, hidden and output (Fig. 2) that communicate with one another over a large number of weighted connections. An artificial neural network is based on the operation of biological neural networks. The neurons in the ANN are the electronic counter part of the neurons of the human brain. Neuron of an artificial neural network consist of

- A set of input values (x_i) and associated weights (w_i)
- A function (φ) known as activation function that operates on the weighted sum (v_k evaluated by (2)), and maps the results to an output (y_k).

$$v_k = \sum_{j=1}^{p} w_j x_j \qquad (2)$$

The model in Fig.1 shows the interval activity of the neuron.

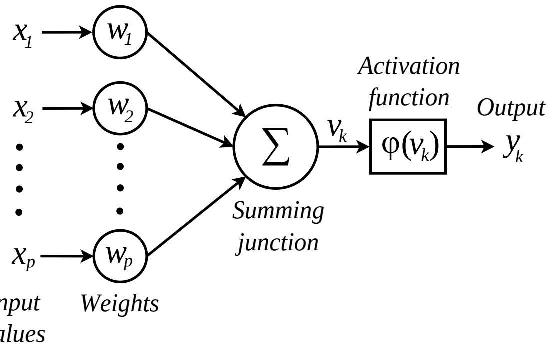

Fig. 1. A Neuron

Various activation functions (φ) available are: threshold, piecewise linear, sigmoid, Elliot and Gaussian etc. We have used Elliot and sigmoid as activation functions, which have been determined experimentally. There may be several hidden layers in the neural network, but we have used a single hidden layer. The number of hidden neurons is determined experimentally. A Neural network can be trained by using sample training data, and then the trained network can be used to predict the class of unknown test sample. Each neuron in output layer corresponds to each class.

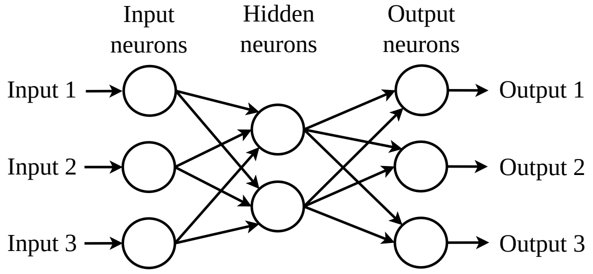

Fig. 2. A Neural Network

C. k-Nearest Neighbour (kNN)

The nearest-neighbor classifier is one of the simplest of all classifiers for predicting the class of the test sample. Training phase simply store every training sample, with its label. To make a prediction for a test sample, its distance to every training sample is computed. Then, keep the k closest training samples, where k≥1 is a fixed integer. Then a label is searched that is most common among these samples. This label is the prediction for this test sample.

This basic method is called the kNN algorithm. There are two major design choices to make: the value of k, and the distance function to use. We have chosen k = 1, 3, 5 and 7 and for the minimum distance, the metric employed is the Euclidean distance given by (3), which evaluates the distance d(x, y) between test and training sample.

$$d(x,y) = \| x - y \| = \sqrt{(x-y)*(x-y)} = \left(\sum_{i=1}^{m}(x_i - y_i)^2\right)^{\frac{1}{2}} \quad (3)$$

Where $x, y \in \mathbb{R}^m$

III. Feature Extraction Methods

Feature extraction is the process of extracting distinctive information from the digitized sample. The aim of feature extraction is to describe the pattern by means of a minimum number of features or attributes that are effective in discriminating it among pattern classes.

A. Statistical Features

Statistical features describe a pattern in terms of a set of characteristic measurements extracted from the pattern. The statistical features which have been used are: Profile Direction Codes, Zoning, Transition and Directional Distance Distribution.

1) Profile Direction Codes: A variation of chain encoding has been used on left, right, top and bottom profiles. First the sample image is scaled to 50*50. For finding the left profile direction codes, the image is scanned from left, from top to bottom and local directions of the profile at each pixel are noted. Starting from current pixel, the pixel distance of the next pixel in east, south or west directions is noted. The cumulative count of movement in three directions is represented by the percentage occurrences with respect to the total number of pixel movement and stored as a three component vector with the three components representing the distance covered in east, south and west directions, respectively. Similarly right, top and bottom profiles are calculated. Therefore a total of 12 profiles features have been obtained (3 for each profile).

2) Transition Features: In transition features, location and number of transitions from background to foreground pixels in the vertical and horizontal directions are calculated. To get this information, sample image is first scaled to 50*50 and then scanned from right-to-left, left-to-right, top-to-bottom and bottom-to-top. A transition which is close to the starting side is assigned a high value compared to a transition computed at the ending side. For example if the transitions were being computed from right-to-left, a transition found close to the right would be assigned a high value compared to a transition computed to the left. A maximum of five transitions have been recorded in each direction. If there were fewer transitions than the maximum value, then the remaining transitions would be assigned values of 0. It will produce four matrices, two matrices having dimensions W × 5(one for top-to-bottom and other for bottom-to-top) and other two matrices having dimensions H × 5(right-to-left, left-to-right), where W is the width and H is the height of the scaled sample image. After evaluation of transitions each matrix has been divided into five equal parts. We have taken the average of transitions vertically in each part. We got 100 (4 × 25) transition feature vector.

3) Zoning Features: For extracting these features, the sample image has been partitioned into the seven equal size windows both horizontally and vertically called zones. Density value (percentage of black pixels) for each of the zone has been calculated. All these density values have been used to form the input feature set. As we have partitioned the sample image into 49 zones, therefore a density value from each zone makes a feature vector set of size 49.

4) Directional Distance Distribution: For these features for each black/white pixel, nearest white/black pixel is located in eight different directions (0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, 315^0). These distances are then stored in a set of size 16, for each pixel. The first 8 elements of this set correspond to the distance of white neighbors of the black pixel in 8 directions. If the current pixel is white then these elements will be set to 0. Similarly the rest of 8 elements correspond to the black neighbors of the white pixel in 8 directions. If the current pixel is black then these elements will be set to 0. For this feature the image is scaled to the size of 36*36. After obtaining such sets for each pixel, the input image array has been divided into 3 equal parts both horizontally and vertically, hence producing 9 zones. From each zone 16 feature vectors have been obtained by adding the corresponding elements of all the sets, corresponding to the pixels in that particular zone. Therefore 16 features from each zone makes a total of 144 (16*9) feature.

B. Gabor Filter

A Gabor filter is a kind of local narrow band pass filter and selective to both orientation and spatial frequency. It is widely applied in the field of character recognition, face and texture recognition. A two dimensional Gabor filter is defined by the equation (4) given below:

$$f(x,y,\phi,\sigma_x,\sigma_y) = exp\left[-\frac{1}{2}\left\{\frac{R_1^2}{\sigma_x^2} + \frac{R_2^2}{\sigma_y^2}\right\}\right] \times exp\left\{i\frac{2\pi R_1}{\lambda}\right\} \quad \ldots(4)$$

Where $R_1 = xcos\phi + ysin\phi \; and \; R_2 = -xsin\phi + ycos\phi$

λ and ϕ are the wavelength and orientation of sinusoidal plane wave, respectively. Where σ_x and σ_y are the standard deviations of Gaussian envelop along x-axis and y-axis. In our case $\sigma_x = \sigma_y$. Before feature extraction the image is scaled to the size 32*32. The Gabor feature can be viewed as the response of Gabor filter, which can be obtained by convolving the filter with an image. A rotation of the x-y plane by an angle ϕ will result in a Gabor filter of orientation ϕ. The value of ϕ is given by $\phi = \pi(k-1)/m,\ k = 1 \ldots\ldots m,$ Where m denotes the number of orientations, which are 9 in our case. The filter response corresponding to all orientations are obtained from the whole image, each quadrant and each sub-quadrant of the image, which make a total of 189 features. We have also experimented by increasing the feature vector size to 252, which have been obtained by changing the number of orientations to 12.

C. Discrete Cosine Transform

Discrete Cosine Transform is the member of a family of sinusoidal unitary transforms. Discrete Cosine Transform efficiently encodes energy/the significant details of the image in a few coefficients. These transform coefficients serve as features for the image sample. For the images we have used two-dimensional DCT represented by equation (5). It calculates the two-dimensional cosine transform of an image. In this function M and N are the height and width of the image, but as the image is scaled to the size of 40*40, therefore for this equation M=N.

$$\mathrm{D(i,j)} = \mathrm{C(i)C(j)} \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} \mathrm{p(x,y)} \cos\left[\frac{(2x+1)}{2M} i\pi\right] \cos\left[\frac{(2y+1)}{2N} j\pi\right] \quad \ldots.(5)$$

Where

$$\mathrm{C(i)} = \begin{cases} \sqrt{\frac{1}{M}} & \text{if } i = 0 \\ \sqrt{\frac{2}{M}} & \text{if } i > 0 \end{cases} \quad \text{and} \quad \mathrm{C(j)} = \begin{cases} \sqrt{\frac{1}{N}} & \text{if } j = 0 \\ \sqrt{\frac{2}{N}} & \text{if } j > 0 \end{cases}$$

D(i, j) represents the DCT coefficient corresponding to the image pixel p(x, y). Therefore the coefficient corresponding to all the image pixels will constitute a feature vector set. Discrete cosine transform concentrates most of the image energy in very few coefficients. The first transform coefficient is called DC component which is at [0, 0] and rest are called AC components. As the image is scaled to 40*40, therefore a total of 1600 features (transform coefficients) can be obtained from it. But we have picked only 100 features in zigzag manner, as shown in Fig. 3. We have also evaluated the feature-classifier performance by increasing the feature size to 200, merely by selecting the first 200 features in zigzag manner.

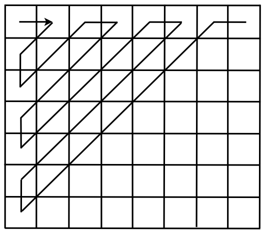

Fig. 3. Zigzag coefficient collection

D. Gradient Features

The gradient features are obtained in three steps: gradient computation, directional decomposition, and feature reduction.

For these features the input image is scaled to the size of 63*63. The gradient vector g(x, y) is then computed at each pixel location using the Sobel operator. Accordingly, the two components; gradient in x and y directions are computed as follow

$$\begin{aligned} g_x(x,y) = f(x+1,y-1) + 2f(x+1,y) \\ +f(x+1,y+1) - f(x-1,y-1) \\ -2f(x-1,y) - f(x-1,y+1), \end{aligned}$$

$$\begin{aligned} g_y(x,y) = f(x-1,y+1) + 2f(x,y+1) \\ +f(x+1,y+1) - f(x-1,y-1) \\ -2f(x,y-1) - f(x+1,y-1) \end{aligned}$$

The magnitude and direction of gradient vectors are evaluated from the components g_x and g_y. The gradient vectors are then decomposed into components in eight chain-code directions [8] as shown in Fig. 4(a). If a gradient vector lies between two discrete directions, it is decomposed into two components (Fig. 4(b)) along the two discrete directions; otherwise the magnitude of the vector is exclusively assigned to the corresponding direction. This decomposition results in 63*63*8 values. These values stored in an array are then divided into 81 blocks. The gradient magnitude is accumulated separately in each of 8 directions, for each block, which results in 648 feature values. These values are then down-sampled by using 5*5 Gaussian filter, which reduces the feature vector size to 200.

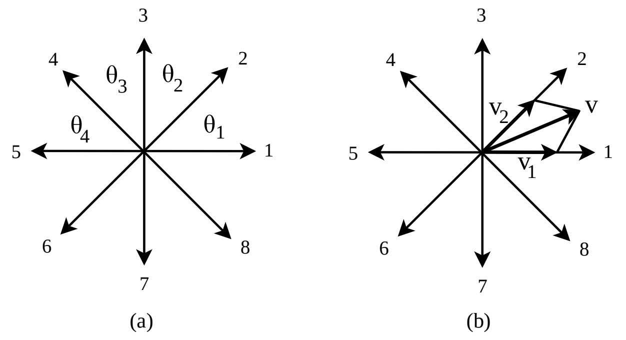

Fig. 4. (a) Eight chain code directions (b) Decomposition of gradient vector

TABLE II. FEATURES AND THEIR VECTOR LENGTH

S.No	Features	Size
1.	Profile direction codes	12
2.	Transition	100
3.	Zoning	49
4.	Directional distance distribution	144
5.	Gabor filter	189/252
6.	Discrete cosine transform	100/200
7.	Gradient features	200

IV. PERFORMANCE EVALUATION

Features of all training and testing samples have been extracted by using above said feature extraction methods. Each of the classifier explained earlier were then trained by using the training features.

A. Feature-SVM Combination

SVM trained with 7000 training samples has been subjected to classify the 1400 test samples. For classification Linear, Polynomial and Radial Basis Function kernels functions have been employed. We have experimented with these kernels by changing the parameters, like degree of the polynomial kernel, and γ for both polynomial and RBF kernels. Table III shows the percentage classification of the test samples. Here we have shown those values of γ for which the classification correctness is maximum.

TABLE III. PERFORMANCE OF FEATURE(S)-SVM COMBINATION

	SVM			
	Kernels			
Features	**Linear**	**Polynomial**		**RBF**
		***deg* = 2**	***deg* = 3**	
Statistical	**97.714**	97.714 γ=0.003	97.714 γ=0.003	97.143 γ=0.00001
Gabor	97.429	**98.000** γ=0.0005	97.143 γ=0.005	97.143 γ=0.0007
DCT	**98.571**	97.429 γ=0.005	98.000 γ=0.001	96.857 γ=0.003
Gradient	**99.429**	98.857 γ=0.005	98.857 γ=0.005	98.286 γ=0.00005
Stat. + Gabor	97.714	**98.000** γ=0.002	97.714 γ=0.002	96.857 γ=0.00005
Stat. + DCT	97.714	**98.000** γ=0.002	97.714 γ=0.002	97.143 γ=0.00001
Gabor +DCT	**98.286**	97.714 γ=0.003	97.714 γ=0.003	97.429 γ=0.0005
Stat. + Gradient	**99.143**	98.571 γ=0.0019	98.571 γ=0.00001	98.286 γ=0.00005
Gabor + Gradient	**99.429**	98.857 γ=0.0025	98.857 γ=0.0025	98.000 γ=0.00005
DCT + Gradient	**99.429**	98.857 γ=0.0001	97.571 γ=0.00001	98.286 γ=0.00003

B. Feature-ANN Combination

In ANN each output neurons represent the class to be detected. Therefore we have used 70 output neurons. The number of input neurons corresponds to the size of selected feature, and hence number of input neurons can be decided from the size of feature vector length.

The number of the hidden layer neurons was determined experimentally. We started with a number close to mean of input and output neurons and then checked the performance by increasing and decreasing the number of hidden layer neurons. Two functions Elliot and sigmoid have been used as activation function for hidden and output neurons respectively. These function have been determined experimentally form the training data.

TABLE IV. PERFORMANCE OF FEATURE(S)-ANN COMBINATION

Feature	**ANN**		
Statistical	91.143 hidden=100	94.571 hidden=180	**96.571** hidden=360
Gabor	92.000 hidden=90	93.429 hidden=125	**96.286** hidden=230
DCT	95.714 hidden=50	96.571 hidden=85	**97.429** hidden=150
Gradient	91.714 hidden=100	94.000 hidden=135	**96.857** hidden=240
Stat. + Gabor	94.571 hidden=200	94.857 hidden=280	**97.429** hidden=320
Stat. + DCT	92.714 hidden=180	95.714 hidden=230	**96.286** hidden=410
Gabor + DCT	93.143 hidden=80	93.429 hidden=170	**97.714** hidden=330
Stat. + Gradient	94.571 hidden=200	95.429 hidden=287	**97.714** hidden=470
Gabor + Gradient	94.000 hidden=180	93.714 hidden=230	**98.000** hidden=340
DCT + Gradient	91.714 hidden=100	94.286 hidden=185	**97.429** hidden=300

C. Feature-kNN Combination

k nearest neighbor is one of the simplest classification method. Here the Euclidian distance between test-sample feature vector and all of the training-sample feature vectors have been evaluated. And then depending upon the value of k the class of test-sample is predicted. Table V depicts the results of this combination for four different values of k for different feature extraction methods.

TABLE V. PERFORMANCE OF FEATURE(S)-KNN COMBINATION

	kNN			
Feature	***k* = 1**	***k* = 3**	***k* = 5**	***k* = 7**
Statistical	**96.000**	95.429	95.143	94.571
Gabor	**95.714**	95.143	94.571	94.000
DCT	**97.143**	95.143	95.429	96.857
Gradient	**96.571**	96.286	94.857	95.714
Stat. + Gabor	**96.000**	94.857	95.143	95.429
Stat. + DCT	**96.000**	95.429	95.429	95.857
Gabor +DCT	**97.143**	96.571	96.000	96.286
Stat. + Gradient	**96.571**	96.286	95.714	96.571
Gabor + Gradient	96.857	**97.143**	95.714	96.286
DCT + Gradient	**96.857**	96.857	95.714	95.714

D. Effect of Feature Size on Performance

The effects of increase in the feature vector length of Gabor and discrete cosine transform on performance have also been evaluated. In order to increase the feature vector length corresponding to Gabor features for both training and test samples, number of orientations has been increased from 9 to 12. This increases the feature size from 189 to 252. Similarly the numbers DCT feature vectors have been increased by selecting 200 feature vectors from the total of 1600 feature vectors in zigzag manner as shown in the Fig. 3.

TABLE VI. PERFORMANCE OF SVM-GABOR AND SVM-DCT WITH VARYING FEATURE VECTOR LENGTH

	SVM			
	Kernels			
Features	Linear	Polynomial		RBF
		deg=2	*deg*=3	
Gabor-189	97.429	**98.000** γ=0.0005	97.143 γ=0.005	97.143 γ=0.0007
Gabor-252	97.429	98.000 γ=0.0003	96.857 γ=0.003	97.571 γ=0.0001
DCT-100	**98.571**	97.429 γ=0.005	98.000 γ=0.001	96.857 γ=0.003
DCT-200	97.142	97.714 γ=0.001	97.428 γ=0.005	96.857 γ=0.00005

TABLE VII. PERFORMANCE OF ANN-GABOR AND ANN-DCT WITH VARYING FEATURE VECTOR LENGTH

	ANN		
Gabor-189	92.000 hidden=90	93.429 hidden=125	**96.286** hidden=230
Gabor-252	90.571 hidden=140	92.571 hidden=160	94.857 hidden=290
DCT-100	95.714 hidden=50	96.571 hidden=85	**97.429** hidden=150
DCT-200	95.714 hidden=100	94.429 hidden=135	95.429 hidden=160

TABLE VIII. PERFORMANCE OF KNN-GABOR AND KNN-DCT WITH VARYING FEATURE VECTOR LENGTH

	kNN			
	***k* = 1**	***k* = 3**	***k* = 5**	***k* = 7**
Gabor-189	**95.714**	95.143	94.571	94.000
Gabor-252	95.429	94.857	94.571	93.714
DCT-100	**97.143**	95.143	95.429	96.857
DCT-200	96.857	95.143	94.857	95.714

V. PERFORMANCE COMPARISON

For performance comparison, maximum percentage classification accuracy of all combinations (Tables III, IV and V) has been taken into account.

TABLE IX. COMPARISON OF FEATURE(S)-CLASSIFIER COMBINATIONS

	Classifiers		
Features	**SVM**	**ANN**	**kNN**
Statistical	97.714	96.571	96.000
Gabor	98.000	96.286	95.714
DCT	98.571	97.429	97.143
Gradient	**99.429**	96.857	96.571
Statistical + Gabor	98.000	97.429	96.000
Statistical + DCT	98.000	96.286	96.000
Gabor + DCT	98.286	97.714	97.143
Stat. + Gradient	99.143	97.714	96.571
Gabor + Gradient	99.429	98.000	97.143
DCT + Gradient	99.429	97.429	96.857

Table IX indicates that Gradient feature and its combination with other features, with support vector machines as classifier outperform the others. Discrete cosine transform also perform well with all the classifiers even though it has minimum feature vector length of size 100.

VI. CONCLUSION AND FUTURE SCOPE

From above discussion it has been found that Gradient feature has provided the maximum classification accuracy of 99.429% only with SVM as compared to other combinations. Above results also show that there is no observable increase in the performance with the increase in the feature vector length of Gabor and DCT features.

As the analysis has been done on the isolated recognizable units therefore there may be variation in the results (e.g. due to segmentation process) when these combinations will be used in actual optical character recognition.

The classification results show that different combinations complement each other; therefore as future scope, some methods can be devised to combine the classification outcome of these feature-classifier combinations to improve the classification accuracy of complete recognition system.

REFERENCES

[1] Sinha and H.N. Mahabala, “Machine recognition of Devanagari script,” IEEE Transactions on Systems, Man and Cybernetics, vol. SMC-9, pp. 435-441, 1979.

[2] R.M.K Sinha and V. Bansal, “On Devanagari Document Processing,” IEEE International Conference on Systems, Man and Cybernetics, vol. 2, pp. 1621-1626, 1995.

[3] V.Bansal and R.M.K. Sinha, “Integrating Knowledge Sources in Devanagari Text Recognition System,” IEEE Transactions on Systems, Man and Cybernetics-part A: Systems and Humans, vol. 30, No. 4, pp. 500-505, July 2000.

[4] V.Bansal and R.M.K.Sinha, “A complete OCR for printed Hindi text in Devanagari script”, Proceedings of the Sixth International Conference on Document Analysis and Recognition (ICDAR’01), pp. 800-804,2001.

[5] B.B. Chaudhuri and U. Pal, “An OCR System to Read Two Indian Language Scripts: Bangla and Devnagari (Hindi),” Proceedings of the 4th International Conference on Document Analysis and Recognition, vol. 2, pp. 1011-1015, Germany, 1997.

[6] S. Kompalli, S. Nayak and S. Setlur, “Challenges in OCR of Devanagari Documents,” Proceedings of the 8th International Conference on Document Analysis and Recognition (ICDAR’05), vol. 1, pp. 327-331, 2005.

[7] S. Kompalli, S. Setlur and V. Govindaraju, “Devanagari OCR using a recognition driven segmentation framework and stochastic language models,” International Journal on Document Analysis and Recognition. pp. 123–138, 2009.

[8] A. Kawamura, K. Yura, T. Hayama, Y. Hidai, T. Minamikawa, A. Tanaka and S. Masuda, “On-line recognition of freely handwritten Japanese characters using directional features densities,” Proceedings of 11th International Conference on Pattern Recognition, Hague, Netherlands, 1992, Vol. II, pp. 183-186.

[9] J. Singh and G.S. Lehal, “Optimizing Character Class Count for Devanagari Character Recognition,” International Conference on Information Systems for Indian Languages (ICISIL2011), CCIS vol. 139, pp. 144-149, 2011.

[10] C.-W. Hsu, C.-C. Chang and C.-J. Lin, “*A Practical Guide to Support Vector Classification*,” http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf. 2010.