Sādhanā (2022)47:112
https://doi.org/10.1007/s12046-022-01872-9

Devanagari handwritten characters recognition using DCT, geometric and hue moments feature extraction techniques

SNEHAL S GAIKWAD*, S L NALBALWAR and A B NANDGAONKAR

Department of Electronics and Telecommunication Engineering, Dr. B A T University, Lonere, Raigad, India
e-mail: snehalg14@gmail.com; nalbalwar_sanjayan@yahoo.com; abnandgaonkar@gmail.com

MS received 23 March 2021; revised 30 December 2021; accepted 18 March 2022

Abstract. Human can easily see and read the document word by word, character by character whether it is written clearly or not. He can also easily guess the next word after the current word. But machine or computer cannot easily predict the handwritten words until they are not trained enough. Optical character recognition is such a system which recognizes the characters both printed and handwritten. It works very well for English characters but the results are not that good for handwritten Devanagari characters, that is why the experiments are going on to train the machine with new features and classification techniques. In this paper, handwritten Devanagari characters recognition system has been presented using discrete cosine transform zigzag features, geometric features (open end points, intersection points, number of horizontal lines, length of horizontal lines, number of right diagonal lines, length of right diagonal lines, number of vertical lines, length of vertical lines, number of left diagonal lines, length of left diagonal lines, foreground area) and the hue moments features. For classification purpose, linear discriminant analysis (LDA), support vector machine with Quadratic kernel, Subspace Discriminant analysis, Subspace K-nearest neighbours and Weighted K-nearest neighbours techniques are considered. Authors have created a database of 3877 Devanagari characters from the scanned handwritten documents of the Marathi notebooks of 5th–9th class students which is very young age group to get variety of characters. Authors have achieved 93.6% recognition accuracy using a combination of all the features with Linear Discriminant Analysis (LDA) classifier.

Keywords. Handwritten documents; discrete cosine transform (DCT); geometric features; hue moments; classification.

1. Introduction

The character has two forms viz., printed character and handwritten character. There is very big difference between them with respect to shape, size, slant, etc. which makes handwritten character difficult to recognize by the machine. Human can easily interpret what is written on the document but it makes difficult for a machine to interpret the character especially when it is handwritten character. A lot of research is going on recognizing the handwritten characters as it is very big challenge for the machine to learn. Different person has unique writing style due to which the skew and slant are introduced in the character hence difficulty in the recognition task by machine. Also, depending upon the speed, mood and especially age of the writer, there is huge variation in writing a character which is also the fact to be considered for the recognition [1]. There are various applications of handwritten character recognition such as bank cheque reading, ancient document recognition, writer recognition, school notebooks preservation written by the students, etc., among which the handwritten recognition of very young age group (students from 5th to 9th class) is very much challenging as they are not that matured to write even a same letter in exact way. So, this challenge and the need to conserve human efforts in reading such messy documents motivated us to built an offline handwritten Devanagari characters recognition system which will recognize the characters having so much variance in shape, size, skew, slant, etc. To evaluate the performance of our system, the authors have considered 370 scanned pages of the notebooks written in Marathi language from the students to get the vast database consisting of अ to ज्ञ Devanagari characters. From these scanned documents, 3877 samples of characters are considered from which partition is made between training and testing characters set. The paper is framed as follows. Characteristics of Devanagari script and database description are discussed in sections 2 and 3 respectively. Literature survey is presented in sections 4. Section 5 describes the proposed system. Classification techniques are explained in section 6. Experimental results are discussed in section 7. Complexity

*For correspondence
Published online: 11 June 2022

Table 1. Basic Devanagari characters used in Marathi language.

Vowels	अ	आ	इ	ई	उ	ऊ	ऋ
	ए	ऐ	ओ	औ	अं	अः	
Consonants	क	ख	ग	घ	ङ	च	छ
	ज	झ	ञ	ट	ठ	ड	ढ
	ण	त	थ	द	ध	न	प
	फ	ब	भ	म	य	र	ल
	व	श	ष	स	ह	ळ	क्ष
	ज्ञ						

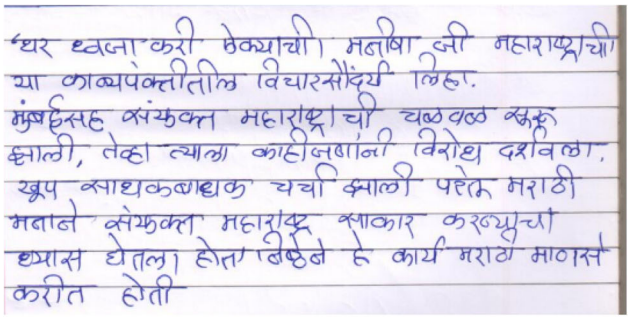
of the algorithm is provided in section 8 and concluding notes and future scopes are presented in section 9.

2. Devanagari script

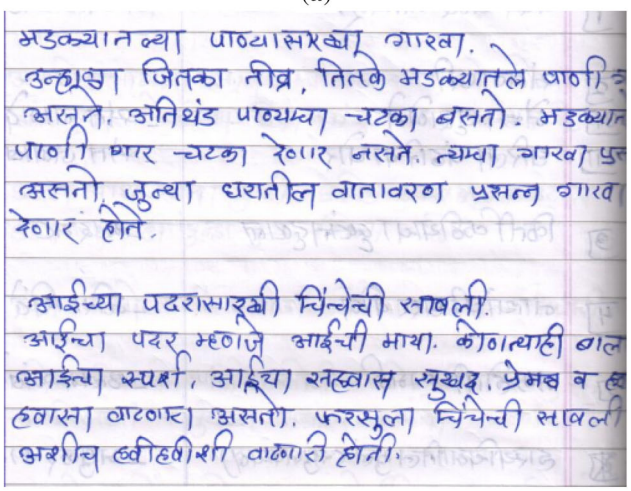
Devanagari script is used by more than 400 million people across the world. Most commonly this script is used in India and Nepal for almost 200 languages and the root of Devanagari script is Nagari script which is also the basis of other languages like Marathi, Sanskrit, Nepali, Bengali, Punjabi, Gujarati etc. Although the Marathi and Hindi languages share the same basis script, they do not share the same characters. For example, Hindi /da/ is written as ड़ whereas Marathi /da/ is written as ड. Some modifiers are also used with the vowels and consonants or they can be written individually to form a word. When two or more individual character such as, क्ष= क+ष, are combined to form new character then it is called as Compound character. When the modifiers are used with the individual characters such as, कैदी= क+ै+ द+ी, then they are called as Conjuncts [2]. Marathi language is having 13 vowels and 36 consonants as shown in table 1. There are many languages in India but not much work has been done for the recognition of Marathi characters [3].

3. Database description

For the proposed work, authors have created handwritten Devanagari character database of 3877 Devanagari characters from the 370 scanned documents of the handwritten notebooks of 5th–9th class students. Source documents were scanned at 300 dpi using an HP flatbed scanner and stored as colour images into .TIF format. The handwritten samples from the students are shown in figure 1. The main purpose behind creating the database from the very young age group is to get variety of shapes, sizes and slants even for the same character. This is due to their unconscious writing style especially when they are asked to write very much things in a single day. The document images are not enhanced using any external software. So, the techniques used for the enhancement of documents will be under preprocessing stage. From each of the documents the characters are extracted manually by using inbuilt MATLAB function 'imcrop' and processed further using proposed system for character recognition. The width and height for each of the handwritten character is different even in the same document which can be easily predicted from the samples shown in figure 1.



(**a**)



(**b**)

Figure 1. Samples of handwritten documents.

3.1 *Comparison of proposed database with existing handwritten Devanagari characters databases*

The proposed database of Indian Devanagari script in this work consists of handwritten samples of documents taken from the real life and to the best of author's knowledge, two standard databases are available for Devanagari characters. First, Bhattacharya and Chaudhuri [4] who developed the Hindi characters database having 30,000 samples of basic Hindi characters written by 1049 different writers which are available in gray scales. Second is, Dongre and Mankar [5] database which contains 5137 numerals and 20,305 isolated characters written by 750 different writers.

Proposed database in this paper is different from the above mentioned two databases in several aspects such as,

1. The original documents are available in colour images from which anyone can pick up the character and test it for recognition.
2. Document samples are not prepossessed that means the characters which are cropped from the colour documents consists of noise which was produced while scanning, also the size of cropped characters is not same.
3. Samples had been collected from a very young age group of students from 5th to 9th class which gives variety of writing a same character in a different way having different skew, slant and size although it is written by the same student.

4. Literature survey

The Devanagari content gives a rich composition framework to various Indian languages like Sanskrit, Hindi, Marathi and Nepali. Research on Devanagari character recognition started a couple of decades prior and a few OCR systems for the Devanagari character recognition have developed from that point [6].

The advancing investigation to improve the perusing capacities of machines offered rise to the field of handwritten document analysis and recognition (HDAR) which examines and recognizes the manually written records. Some commonly used applications of HDAR systems are automated postal addressing, zip code reading, data acquisition in bank checks and institutional records [7]. Each stage of character recognition system should be able to perform well towards the recognition process, otherwise performance rate degrades. Recognition of handwritten characters is still a very challenging task which suffers from handwriting variations, low-quality images, different character recognition techniques, classification problems and various other factors. The recent recognition methods are good at recognizing the good quality characters but fails in recognizing overlapped, similar shaped, slanted, skewed characters which reduces the recognition rate. Sharma *et al* [8] introduced different preprocessing (smoothening, binarization, thinning and spur removal), feature extraction (vertical bar, intersections, shirorekha touch frequency and number of endpoints and slopes of ending curves) and classification techniques for printed and handwritten characters. They achieved 93.33% accuracy on printed characters and 72.72% accuracy on handwritten characters.

Researchers are also attracted towards the ancient documents which are preserved in the museums. The characters in ancient documents are often overlapped, so thinning algorithm can badly twist them. Also, if the characters are touching to each other or they are half, then it degrades the recognition rate which is mostly rely upon the segmentation of the characters. Narang *et al* [9] focused on such touching and half characters, where they used 5484 samples of presegmented basic Devanagari characters dataset from ancient documents. They used zigzag DCT features and classified using support vector machine (SVM), Naïve Bayes and Decision tree classifiers in combination with adaboost (adaptive boosting) and bagging to improve the recognition results. For recognition purpose, they tried different lengths of DCT features in combination with classifiers. Their work fails for modifiers and conjuncts for recognition. Narang *et al* [10] developed a new approach for segmentation of lines, words and characters from ancient documents. They used zonewise projection profile concept, where they concentrated on vertical and horizontal projection of character, average line height and connected components in the character. The image was divided into fixed size vertical zones. For each vertical zone, horizontal projection profile (HPP) was calculated. Depending upon the threshold for each HPP, zonewise separating lines were considered for each row. They achieved line segmentation accuracy of 97%, character segmentation accuracy of 98.5% and overlapping and touching components accuracy of 100% and 96% respectively. But the accuracy of the algorithm degrades for the issues like if the average line height of character was not computed correctly and width of isolated character was large.

Characters has two main features viz., structural and statistical. Structural features includes the shape and overall structure of character where statistical features includes horizontal and vertical peak content, centroid, intersection and open endpoints [11]. Kumar *et al* [11] presented a study of recognition for handwritten English, Hindi and Punjabi characters with accuracy of 92.18%, 84.67% and 86.79% respectively. They extracted zoning features, diagonal features, horizontal peak extent based features, intersection and open endpoint based features. For classification, they used KNN (K nearest neighbor), SVM (support vector machine) and MLP (multilayer perceptron) classifiers. They also reported some challenges like similar shaped characters with same meaning, extraction of appropriate features and selection of feature extraction techniques, low quality images, cursive case of handwriting, inappropriate scanning and binarization which affects the text recognition rate. Puri *et al* [12] presented the work on classification of printed Hindi words into the respective category of documents. The model searches and stores the meaningful words and generates same word combinations with different modifiers to categorize them. For such words, the frequency of occurrence was calculated and classified in particular document. Impedovo *et al* [13] reviewed different types of uniform and non-uniform (slice-based, shape-based, hierarchical) zoning methods for handwritten character recognition. Bansal and Sinha [14] developed an approach for segmentation and decomposition of composite character. The composite character was converted into their fundamental symbols (individual basis characters) from which structural properties were extracted as features. Firstly, segmentation of words was done from which statistical information like height and width of each separated word was used to decide whether a character box was of composite characters or not. If it is, then those composite characters were segmented into

individual characters. In this approach, they achieved 85% accuracy for segmentation of characters. Online handwritten cursive and non-cursive handwritten recognition is addressed in [15] for Devanagari and Bengali script using Long-Short Term Memory (LSTM) and Bidirectional Long-Short Term Memory (BLSTM). They divided the word into upper, middle and lower zone from which the middle zone is used for the segmentation and the features are extracted. Basic stroke based class labelling approach was used as feature which gave 99.50% accuracy using RNN classifier.

After accomplishing the literature survey, authors found that very little work has been reported on the handwritten Devanagari characters where mostly the data was collected from the postal office and office records. Also, the data was created only for the individual characters where the persons are asked to write the character in the predefined box which gives almost similar size of all the characters. Hence, in this work, the authors have created a new Devanagari character database from a very young age group which is challenging to work on where the size of the characters is small as they are extracted from the notebooks and having lot of variations even for the same character. For these characters, the authors have presented a Devanagari character recognition system using combination of different features and tested with multiple classifiers.

5. Proposed system

For recognition of Marathi Handwritten documents, the proposed system consists of various phases like document scanning, preprocessing, segmentation, feature extraction and recognition. Block diagram of the proposed system is shown in figure 2.

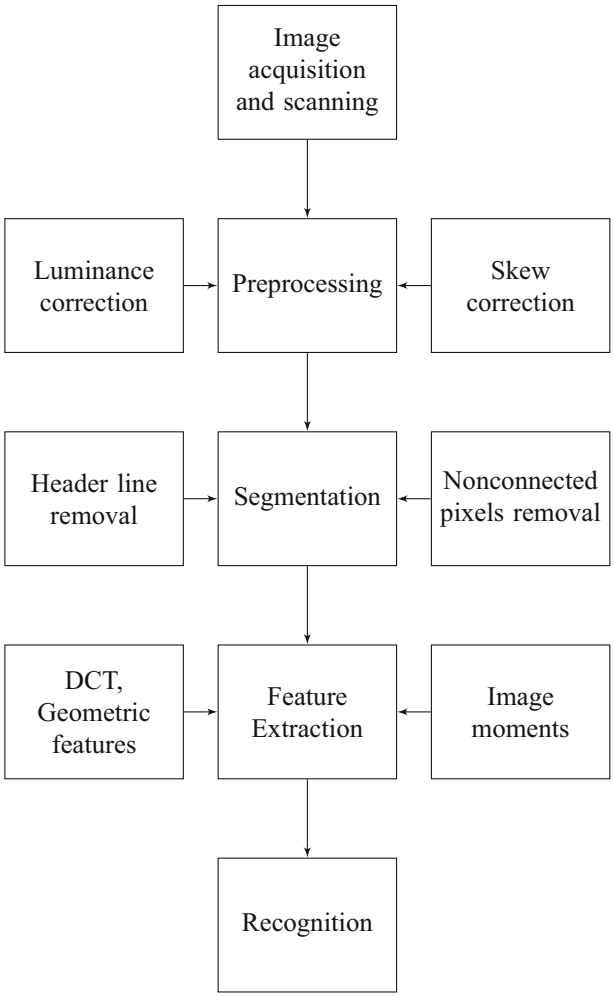


Figure 2. Block diagram of the proposed system.

5.1 *Image acquisition and scanning*

For this work, authors have collected the handwritten documents of different subjects written in Marathi language. Scanning means digitizing the handwritten document into an electronic form to avail that for further preprocessing. Quality and resolution are very important for recognition purpose, keeping this in mind a non-compressed lossless file format is required. For this purpose, TIFF (Tagged Image File Format) is used for storing the images.

5.2 *Preprocessing*

The following steps depicts how different operations are carried out in the preprocessing stage to enhance the quality of an image.

Step 1: The character is cropped from the scanned document using inbuilt 'imcrop' MATLAB function and result for the same is depicted in figure 3a.

Step 2: In this step, the luminance of an input image is corrected. Luminance is the intensity of light emitted from the surface in given direction. While scanning the document, the luminance is affected to so much extent and should be corrected to get more enhanced image. The luminance correction is applied to all Red (R), Green (G) and Blue (B) planes of the coloured image where each pixel value is multiplied by different factor (in our case, R plane is multiplied by 0.2, G plane is multiplied by 0.2, B plane is multiplied by 0.01) to get enhanced image an shown in figure 3b. As the values for R and G plane are increased, the image becomes so darker and degrades the quality of an image. So, the values are chosen which gives good results.

Step 3: In this step, luminance corrected image is converted into gray scale image. Again, gray intensity values are mapped into new grayscale values (in our case range is from 0.5 to 1) as shown in figure 3c.

Step 4: Here, the image is provided for the normalization and skew correction. Normalization is done using the standard deviation of the pixels of whole character, where

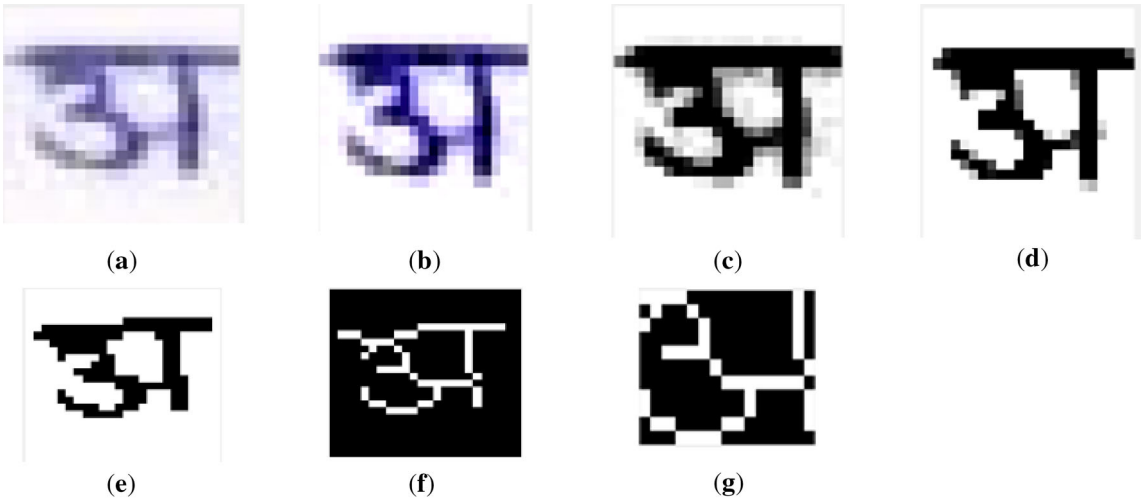


Figure 3. Preprocessing and Segmentation results **a** cropped colour image **b** luminance corrected/contrast enhanced image **c** gray intensity adjusted image **d** normalized image **e** binarized image **f** thinned image **g** cropped headerfree.

it is assumed that low standard deviation means how pixel intensities are spread around the mean. The result of normalization is shown in figure 3d, where it shows more clean view of the character shape. For skew correction of the handwritten character, entropy based information is used for finding optimal deskew direction.

Step 5: For binarization of the character, polynomial curve fitting algorithm using least square distance is implemented. It is also called as histogram based approach.

The general algorithm for polynomial thresholding is given below.

1. Plot the histogram of the normalized gray scale image to find the intensity values and their respective count.
2. Fit the polynomial on those intensity values with degree of 6 using Least Square Distance. (The degree is decided based on which gives good results of best fit after making number of trials.) This gives the 6 coefficients which are required for that degree polynomial to touch on the histogram.
3. Take the first order differentiation (difference) between new intermediate intensity values and sort them in ascending order (for ease of calculations).
4. On these new values, second order differentiation is taken (for better approximation) to be zero at a point (point of inflection), to find out minimum value with its index which is near to zero curve and that minima is used as a threshold to binarize the image.

The result of binarization is shown in figure 3e.

5.3 *Segmentation*

In this step, the header line removal and extra non connected pixel removal is carried out. The binary image is thinned first and then the area of interest is cropped out as shown in figure 3f, g respectively. Removal of the header line is not easy as the header line can be present or not. If it is present then, row of maximum pixels is picked up and considered as a header line to be removed.

Algorithmic steps for header line removal in this work are given below.

Step 1: The height of the character in terms of pixel is calculated (i.e., total rows in the image are calculated).

Step 2: From the first 50% (half horizontal) part of character, row consisting of greater than seven pixels is considered as the header line. (Index of the row having greater than 7 pixels is considered)

Step 3: Due to some pre-processing techniques, pixels in the header line are not limited to single row (they are spread above and below due to thinning operation), so the indices of rows above and below the row whose index is already considered as header line (from step 2) are also considered to be a part of header line.

Step 4: If from the first 50% (half horizontal) part of character, no row consists greater than seven pixels that means algorithm considers that header line is not present.

5.4 *Feature extraction*

Feature extraction is an integral part of pattern recognition which reduces dimensionality of an image. It resourcefully conveys the main characteristics of that image as a compact feature vector. So, it becomes very complex when the features or statistics of an object or any image is inadequate. Creating such attributes from an image is called feature extraction.

In this step, authors have considered different types of features like Discrete Cosine Transform (DCT) zigzag

features, geometric features (open end points, intersection points, number of horizontal lines, length of horizontal lines, number of right diagonal lines, length of right diagonal lines, number of vertical lines, length of vertical lines, number of left diagonal lines, length of left diagonal lines, foreground area) and the hue/image moments features.

5.4.1 *Discrete cosine transform (DCT)* DCT is a special case of class of transformation. For an image, the intensity values do not change abruptly between pixels hence this transformation shows that most of the significant information about the image is concentrated in just a few coefficients of DCT matrix and that is called as energy compaction attribute of DCT transformation.

The general 2D transformation can be written as,

$$T(u,v) = \sum_{x,y=0}^{N-1} \sum_{x,y=0}^{N-1} f(x,y) \quad g(x,y,u,v) \tag{1}$$

Where, f(x,y) is N x N image, g(x,y,u,v) = 2D forward transformation kernel of the basis functions.

The corresponding inverse transformation is given by,

$$f(x,y) = \sum_{u,v=0}^{N-1} \sum_{u,v=0}^{N-1} T(u,v)h(x,y,u,v) \tag{2}$$

Where, h(x,y,u,v) = 2D inverse transformation kernel of the basis functions.

For DCT, the forward and backward kernels are same.

2D forward and inverse DCT using kernals are given by following equations respectively.

$$T(u,v) = \alpha(u)\alpha(v) \sum_{x,y=0}^{N-1} \sum_{x,y=0}^{N-1} f(x,y) cos\left[\frac{(2x+1)\pi u}{2N}\right] cos\left[\frac{(2y+1)\pi v}{2N}\right] \tag{3}$$

$$f(x,y) = \sum_{u,v=0}^{N-1} \sum_{u,v=0}^{N-1} \alpha(u)\alpha(v)T(u,v) cos\left[\frac{(2x+1)u\pi}{2N}\right] cos\left[\frac{(2y+1)v\pi}{2N}\right] \tag{4}$$

The upper left corner of DCT basis matrix is DC coefficient which includes the lowest frequency in both the direction and rest are the AC coefficient. We can think of an image as construction of these basis images. The reason behind using the few AC coefficients along with DC coefficient is to get the more accurate image formation and unique feature selection. In proposed work, 15 DCT coefficients are taken in zigzag manner starting from upper left corner.

5.4.2 *Geometric features* Blumenstein *et al* [16] extracted various types of geometric features for handwritten English lower-case and upper-case characters using Back-Propagation (BP) and Radial Basis Function (RBF) networks classifiers. There is very huge difference in English and Devanagari character in terms of the shape and the way they are written. English characters are easy to write as they are not having the compound characters and conjuncts. Also, each English letter has different geometry but that is not the case for Devanagari characters as lot of characters are similar in shape and hence the geometry (e.g., फ, क). Important analysis of the pixels before extracting geometric features must be done and for that authors have considered following pixels position.

1. Open end points: If any pixel is having only one pixel in its neighbourhood, they are the open end points. For finding the line segments, these point's locations are stored. They are also called as starting points.
2. Intersection Points: If the pixel has more than one neighbour then it is identified as intersection. When the pixels are adjacent to the pixel under consideration then that pixel is not considered as intersection.
3. Surrounding pixels: When the pixels are connected with 8 connectivity with respect to central pixel, they are presented as surrounding pixels of that pixel under consideration and their positions are stored to find the direction of line made by those pixels in that zone.

By considering the above pixel properties, the direction of line is found by taking the difference between surrounding pixel's position. For each difference (the coordinates), we have decided the direction types from 1 to 8 as shown in figure 4. For example, for 3×3 region, if the pixel is present at top left corner, the coordinates will be (1,1) and the coordinates of central pixel P will be (2,2), then the difference between coordinates will be (1,1)–(2,2)= $(-1,-1)$, which give the direction point as 4 (for $(-1, -1)$ we have decided point value as 4). Again if the pixel is present at bottom right corner, the coordinates will be (3,3), then the difference between coordinates is (3,3)–(2,2)= (1,1), which will give the second direction point for line as 8 (for (1, 1) we have decided point value as 8). So, the direction of line will be (4,8) i.e., right diagonal. Likewise, for different difference values, we have decided the direction points from 1 to 8.

These line segments (horizontal, vertical, right diagonal, left diagonal) are extracted from different zones of an headerfree image and feature vector is formed. Due to the thinning operation on an image, there exist some spurious pixels which also gets marked in any particular line

4	5	6
3	P	7
2	1	8

Figure 4. Direction types of line segment.

segment. Hence, that line segment has to be normalized (discarding spurious direction values) by considering the number of values belonging to each direction type in a particular line type. For this, we have considered most repeated number in a particular line segment to replace spurious direction values. For zoning purpose, extra zeros were added row wise or column wise if needed, to zone the headerfree character mage into the windows of equivalent size. Then from each zone, the line segments were found with the help of starter points, intersection points and surrounding pixels. Algorithm then proceeds to calculate the number and lengths of line segments in each zone.

Total 11 geometric features are considered in this work viz., open end points, intersection points, number of horizontal lines, total length of horizontal lines, number of vertical lines, total length of vertical lines, number of right diagonal lines, total length of right diagonal lines, number of left diagonal lines, total length of left diagonal lines and foreground area (number of ones in the particular zone).

The number of lines (horizontal/vertical/right diagonal/left diagonal) in a particular zone are calculated as follows:

$$\text{Total number of lines} = 1 - \frac{\text{number of lines}}{\max(\text{image\_height}, \text{image\_width})} \tag{5}$$

Above formula is configured as the value for number of particular line type increases by one, the answer value will decrease by 0.05 approximately. Here, the character image size is resized to 12×18 to keep the constant value of width and height.

The normalized length of particular line type is also calculated where it is assumed that the maximum length of one single line type is a part of total number of ones in the image (skeleton size). As an example, if the line length is 7 pixels and the skeleton size is 28, then 7/28 = 0.25. (i.e., how much of the total skeleton occupied by that particular line type).

$$\text{Normalized\_Length} = \text{total number of pixels in a particular direction /skeleton size} \tag{6}$$

5.4.3 *Hue moments* Hue moments are certain particular weighted average of pixel intensities. They are used to represent the properties of an object like shape, size, position, orientation. The Devanagari handwritten characters have different shapes and orientations even for the same character class, so we get more discriminant features, which in turn reduces the accuracy of recognition. So, there is a need to extract such features which are translation, rotation and scale invariant. In this work, authors have considered 7 Hue moments also called as Image moments which are translation, rotation and scale invariant. Translation can be done using only central moments but if the image is closer or farther from the camera, the things get bigger or smaller respectively and we get huge scale differences. So, we want to create some descriptors that are invariant and hue moments are of that type. Actually, these moments are combination of absolute orthogonal invariants and similitude (similarity) invariants of central moments due to which pattern identification can be made independent of position, size, orientation [17]. Among the 7 hue moments, first 6 are absolute orthogonal invariants and 7th is skew orthogonal invariant which would be useful in differentiating mirror geometry in handwritten character.

As the character size is small, some pixels disappears due to thinning operation. So, authors have considered these image moments for dilated (using 2×2 all ones mask) character because as we dilate the image, less is the difference among the shapes of characters from same class.

For an image f(x,y) of size M by N, the central moment is calculated as follows,

$$\mu(i,j) = \sum_{x=1}^{M} \sum_{y=1}^{N} (x - \bar{x})^i (y - \bar{y})^j f(x,y) \tag{7}$$

where $\bar{x}$ and $\bar{y}$ are average of x and y respectively, also called as centre of mass and written as shown in equations (8) and (9),

$$\bar{x} = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} x f(x,y)}{\sum_{x=1}^{M} \sum_{y=1}^{N} f(x,y)} = \frac{m_{1,0}}{m_{0,0}} = \frac{\textit{first moment in } x}{\textit{overall area}} \tag{8}$$

$$\bar{y} = \frac{\sum_{x=1}^{M} \sum_{y=1}^{N} y f(x,y)}{\sum_{x=1}^{M} \sum_{y=1}^{N} f(x,y)} = \frac{m_{0,1}}{m_{0,0}} = \frac{\textit{first moment in } y}{\textit{overall area}} \tag{9}$$

Using these central moments, the invariants with respect to translation and scaling are formed by dividing with a properly scaled zero-th central moment as,

$$\eta(i,j) = \frac{\mu(i,j)}{\mu(0,0)^{1+((i+j)*0.5)}} \tag{10}$$

Using equation (10), the seven invariants with respect to translation, scale, and rotation are constructed as,

$$M1 = \eta 20 + \eta 02 \tag{11}$$

$$M2 = (\eta 20 - \eta 02)^2 + 4\eta 11^2 \tag{12}$$

$$M3 = (\eta 30 - 3\eta 12)^2 + (3\eta 21 - \eta 03)^2 \tag{13}$$

$$M4 = (\eta 30 - \eta 12)^2 + (\eta 21 + \eta 03)^2 \tag{14}$$

$$\begin{aligned} M5 = {} & (\eta 30 - 3\eta 12)(\eta 30 + \eta 12)[(\eta 30 - \eta 12)^2 \\ & - 3(\eta 21 + \eta 03)^2] + (3\eta 21 - \eta 03) \\ & (\eta 21 + \eta 03)[3(\eta 30 + \eta 12)^2 \\ & - (\eta 21 + \eta 03)^2] \end{aligned} \tag{15}$$

(a) (b) (c) (d)

Figure 5. Zoning of handwritten character image **a** 2 vertical zones **b** 2 horizontal zones **c** 3 vertical zones **d** 3 horizontal zones.

$$M6 = (\eta 20 - \eta 02)[(\eta 30 + \eta 12)^2 - (\eta 21 + \eta 03)^2] + 4\eta 11(\eta 30 + \eta 12)(\eta 21 + \eta 03) \quad (16)$$

$$M7 = (3\eta 21 + \eta 03)(\eta 30 + \eta 12)[(\eta 30 + \eta 12)^2 - 3(\eta 21 + \eta 03)^2] - (\eta 30 + \eta 12)(\eta 21 + \eta 03)[3(\eta 30 + \eta 12)^2 - (\eta 21 + \eta 03)^2] \quad (17)$$

Hence, authors have considered these three types of features in the present work from which DCT and geometric features are extracted from both dilated and skeleton types of images and image moments are extracted from dilated type of images. To extract these features, zoning is applied on the headerfree character which divides the character into the several parts of same dimensions. The dilated image is zoned into 2×1 and 1×2 parts while the skeleton image is divided into 3×1 and 1×3 parts and the dimension of whole image size is fixed to 12×18 as depicted in figure 5. For open end points and intersection points whole skeleton of the image is considered and for the rest geometric features zoning is applied. Hence, the feature vector of 129 (30 DCT + 92 geometric + 7 image moments) features are considered in the present work.

6. Classification

Classification techniques used by the authors in this work are explained in this section. Classification is used to classify the handwritten Devanagari characters into their respective category or class. Classification is done using Linear Discriminant Analysis (LDA) (Classifier1), Support Vector Machine (SVM) with Quadratic kernel (Classifier2), Subspace Discriminant Analysis (SDA) (Classifier3), Subspace K-nearest neighbours (SKNN) (Classifier4) and Weighted K-nearest neighbours (WKNN) (Classifier5). Following subsections introduces briefly the classifiers used in the present work.

6.1 *Linear discriminant analysis (LDA)*

Linear discriminant analysis (LDA) is based on the fact that it will lead to a linear separator in order to distinguish some categories. Deriving a separator is the challenging task. The very basic idea of LDA is to move the data from higher dimensions to lower dimensions i.e., it looks at the projections of the data lines from higher to lower dimensions. It basically defines two terminologies, which are inter class separability and between class separability and performs in a way of getting minimum inter class separability and maximum between class separability. So, it is challenging to select an axis that will try to optimize both of the criteria at the same time. It assumes all data as normally distributed. In our work, the purpose of using LDA is, it is a multi-class classifier which finds the direction to maximize the variance between classes and minimize the variance within classes. So, the mean is found by considering the point that is central to all of the data and then distances are measured between that central point and the points that are central in each category. After this, the distance between each category and central point is maximized while variance for each category is minimized.

Algorithm for LDA is explained below, Step 1: Compute within class variances (i. e. within class scatter matrix which shows how the data is scattered within individual class, S(w))

$$S(w) = \sum_{i=1}^{c} S_i \quad (18)$$

$$S_i = \sum_{x \in C_i}^{n} (x - \mu_i)(x - \mu_i)^T \quad (19)$$

S_i is Covariance/scattered matrix of class i (individual class) and μ_i is mean of class i (individual class).

Step 2: Compute between class variances (i.e. between class scatter matrix which shows how the classes are scattered, S(b))

$$S(b) = \sum_{i=1}^{c} N_i(\mu_i - \mu)(\mu_i - \mu)^T \quad (20)$$

where, μ is overall mean.

Step 3: Find the eigen value and eigen vector of $S(w)^{-1} S(b)$

6.2 *Support vector machine (SVM)*

SVM's were developed in the 60s and were improved significantly in late 1990s. Today they are the most

effective and powerful machine learning algorithm. From the number of categories, it separates the points on the plane into the different categories. The main consideration in SVM is how to obtain the decision boundary which will separate the points of the given categories so that when the new point will occur that has not been classified yet, the machine will know exactly into which category that will fall. Many lines or planes can be drawn based on the concept of maximum margin to divide these categories. These margin points are called the support vectors which will strongly categorize the points. In two dimensional space, the maximum margin is called as a line and in higher dimensional space it is called as a hyperplane as shown in the figure 6. For higher dimensions, support vector machines use the kernel functions to systematically find support vector classifiers. The reason behind using SVM in our work is their effectiveness in classifying the large number of features. They gives the good accuracy but as the data increases, they requires more training time as compare to other classifiers used in the present work.

In proposed work, 39 classes are considered where the features from each class are nonlinearly separable. So, we have experimented kernels like linear, quadratic, cubic and RBF to compute the relationships between the observations in higher dimensions and found the support vector classifier. The statistical analysis for the SVM classifier using different kernels for different combinations of features are shown in figure 7. As said above, due to nonlinearity of the data, linear SVM has less accuracy of recognition, so kernel with high power is used in calculating the dot product between two vectors in (high dimensional) feature space. Also, due to so much spread of features, RBF SVM does not perform well as compare to others because, when the features from particular class are separated by large distance then the kernel value becomes very small, near to 0 which means that the features are dissimilar. In our case, quadratic SVM performs better than other kernels hence, authors have considered only quadratic kernel SVM in the present paper.

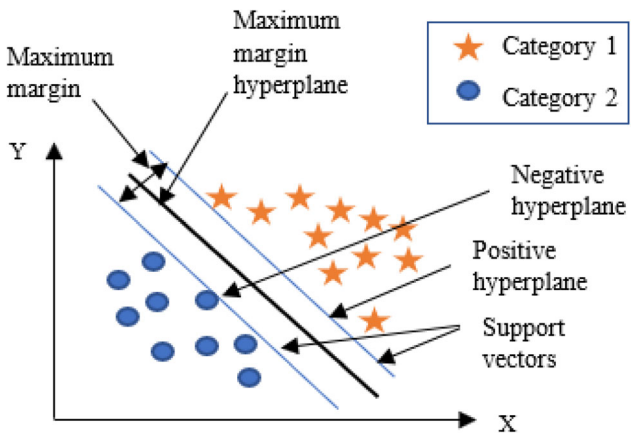


Figure 6. General SVM graphical stucture for two categories.

6.3 *Subspace discriminant analysis (SDA)*

Subspace discriminant analysis has two main aspects, first is extracting the most discriminant features from a subspace and second is to utilize as much discriminant information in the feature space. The subspaces are created to reduce the feature vector dimension and the classifier is built for those features in that subspace [18]. In the present paper, ensemble method randomly selects the features in the selected subspace and creates models for each learners (in our case, 30 learners) in the same subspace and finally combines them using majority vote. Here, the whole feature space is divided into two subspace dimensions.

6.4 *Subspace k-nearest neighbor (SKNN)*

This subspace method depends on a process which randomly selects a number of features from given feature vector of particular category which is equivalent to projecting all the features to the selected subspace and finding the k-nearest neighbors using projected distances for building the classifier. Each time a random subspace is chosen, another arrangement of k closest neighbors are found. The k-nearest neighbors in each chosen subspace are then amassed for a greater part vote on the class membership of the new unknown data point which is to be classified [19].

6.5 *Weighted k-nearest neighbor (WKNN)*

Simple KNN has the disadvantage of choosing the value of parameter k. For small k, the algorithm would be more delicate. For large k, the area may include too many features from other classes. So, authors have used WKNN classifier which takes the majority vote on the nearest neighbours using the distance metric and finds the closest neighbors those are more reliably indicates the same class of the image. To find the closest distance, L1 distance metric is used which is mostly preferred when we have to deal with high dimensional feature space [20]. In the proposed work, authors have used city block distance measure [L1 distance] for measuring the distances between the nearest neighbours and 10 nearest neighbours are considered. When the data is too much spread and not that much discriminant, WKNN may give less accuracy to classify the character into correct class.

7. Experimental results and discussion

In the proposed work, authors have considered 3877 characters (39 classes), which are extracted from the 370 scanned documents from the notebooks of 5th–9th class students. In recognition of vowels and consonants, 2

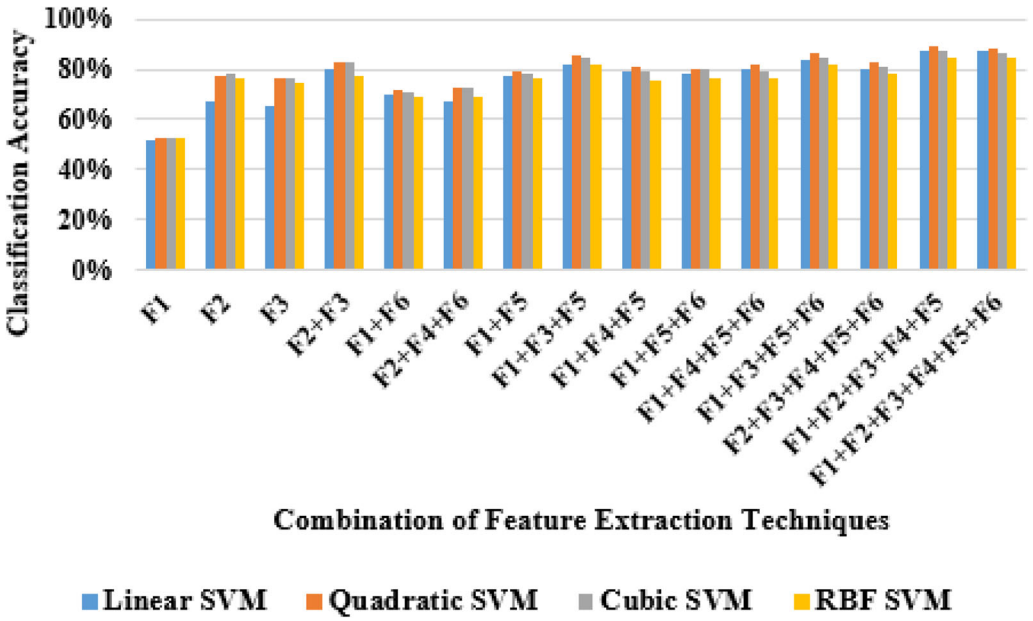


Figure 7. Statistical analysis for SVM classifiers using different kernels and combination of features.

geometric features from skeleton type of character (open end points and intersection points) (F1), 15 DCT zigzag coefficients from dilated type of character (F2), 15 DCT zigzag coefficients from skeleton type of character (F3), 36 geometric features from zoned dilated type of character (F4), 54 geometric features from zoned skeleton type of character (F5) and 7 image moments from dilated type of character (F6) are considered. For classification, 5 classifiers are considered which gave the highest accuracies for the present work, viz., Linear Discriminant Analysis (LDA) (Classifier1), Support Vector Machine (SVM) with Quadratic kernel (Classifier2), Subspace Discriminant Analysis (SDA) (Classifier3), Subspace K-nearest neighbours (SKNN) (Classifier4) and Weighted K-nearest neighbours (WKNN) (Classifier5) among which Linear Discriminant Analysis (LDA) (Classifier1) performs better than other classifiers. The steps for extracting the above mentioned features from two types of an image are given below.

7.1 *Features extraction from dilated handwritten character*

The images which are ready to extract the features are first dilated using 2×2 all one's mask.

Step 1: Apply the DCT to overall dilated image and extract the 15 DCT features from the upper left corner in zigzag manner. (Total 15 features are considered from this step.)
Step 2: Zone the image into 1×2 and 2×1 zones i.e., first divide the whole image into 1×2 (all rows are considered and columns are divided into two zones) and extract 9 geometric features from each zone, then divide the whole image into 2×1 (all columns are considered and rows are divided into two zones) and again extract 9 geometric features from each zone. (Total 36 features are considered from this step.)
Step 3: Seven image moments are extracted from the whole dilated image. (Total 7 features are considered from this step.)

So, from the dilated type of an image total 58 $(15 + 36 + 7)$ features are extracted.

7.2 *Features extraction from skeleton of handwritten character*

Step 1: From the whole skeleton character, 2 geometric features (open end points and intersection points) are extracted. (Total 2 features are considered from this step.)
Step 2: Apply the DCT to overall skeleton image and extract the 15 DCT features from the upper left corner in zigzag manner. (Total 15 features are considered from this step.)
Step 3: Zone the image into 1×3 and 3×1 zones i.e., first divide the whole image into 1×3 (all rows are considered and columns are divided into three zones) and extract 9 geometric features from each zone, then divide the whole image into 3×1 (all columns are considered and rows are divided into three zones) and again extract 9 geometric features from each zone. (Total 54 features are considered from this step.)

So, from the skeleton version of an image total 71$(2 + 15 + 54)$ features are extracted.

Total features considered for classification purpose from both the image versions are 129 $(58 + 71)$. The

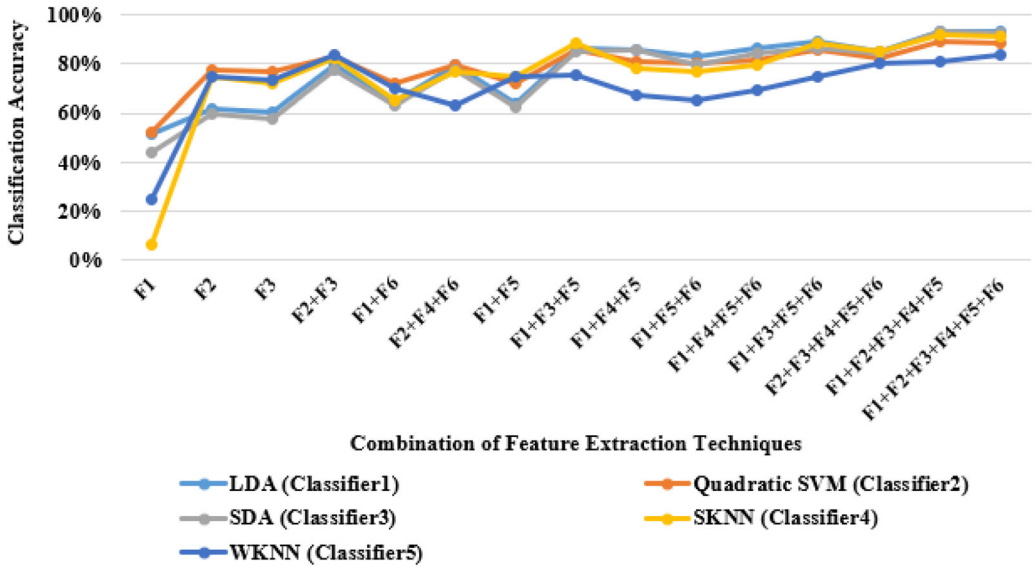


Figure 8. Statistical analysis for the results depicted in table 2.

classification results of proposed work using different combinations of these features for different classifiers using 5-fold cross-validation technique are depicted in table 2. Statistical analysis for these classification results is also depicted in figure 8. When only F1 feature set is used, the LDA and quadratic SVM performs better than other classifiers with 51.8% and 52.4 % accuracy respectively. The accracy is low for F1, as the character size is too small as they are extracted from the notebooks, it may fail to count the exact number of open end points, intersection points. For F2 feature set, the recognition accuracy of 77.3% is achieved for quadratic SVM and it increased for F2+F3 upto 82.8%. LDA gives the highest accuracy of 86.6% for feature set F1+F3+F5 and it is increased to 89% for F1+F3+F5+F6. The Devanagari characters are almost similar in shape which gives almost similar F1 (open end points and intersection points) for all the characters which leads minimum classification accuracy of 6.5% and 25.2% for SKNN and WKNN respectively as they uses the subspace of nearest neighbours. Using combination of all the features (F1+F2+F3+F4+F5+F6), proposed system achieved accuracy of 83.6% for WKNN, 88.7% for Quadratic SVM, 91.5% for SKNN, 92.8% for SDA and 93.6% for LDA. LDA performs better as it is supervised learning algorithm that computes the directions or discriminants using eigen vectors which maximizes the between class variability where the class labels are already known to it. So, in this work authors have considered the results obtained from LDA classifier and confusion matrix for the same is shown in table 3. Confusion matrix shows how the characters from particular classes are confused with the other classes.

Narang *et al* [21] reported a work of finding the features like Intersection and open endpoint features, Centroid features, Horizontal peak extent features, Vertical peak extent features and all of their possible combinations to improve recognition accuracy from ancient Devanagari documents. For classification of characters, they used Multilayer perceptron MLP, Neural Network, Convolutional Neural Network CNN, RBF-SVM and random forest techniques. Finally, the recognition accuracy of 88.95% was achieved

Table 2. Classification results using different combination of features and classifiers.

Combination of features	Classifier accuracy (%)				
	LDA	Quadratic SVM	SDA	SKNN	WKNN
F1	51.8	52.4	44.3	6.5	25.2
F2	62.1	77.3	59.6	74.6	74.5
F3	60.6	76.7	58	72.4	73.5
F2+F3	79.5	82.8	77.6	82.2	83.9
F1+F6	64.3	72	62.9	65.2	69.7
F2+F4+F6	79.9	79.4	77.4	76.8	63.1
F1+F5	63.7	72.4	62.7	75.1	75.1
F1+F3+F5	86.6	85.6	84.9	88.7	75.5
F1+F4+F5	85.7	81.1	85.8	78.4	67
F1+F5+F6	83.1	80.4	79.8	77.2	65
F1+F4+F5+F6	86.4	81.5	84.7	79.3	69.6
F1+F3+F5+F6	88.9	86	86.4	88.3	74.5
F2+F3+F4+F5+F6	85.1	82.4	84.1	84.8	80.2
F1+F2+F3+F4+F5	93	88.9	93.3	91.8	81
F1+F2+F3+F4+F5+F6	93.6	88.7	92.8	91.5	83.6

Table 3. Confusion matrix based on a combination of all features for both dilated (2 × 1, 1 × 2) and skeleton image (3 × 1, 1 × 3) of size 12 × 18 respectively using LDA classifier.

Test Character	Total characters for recognition	Correctly recognized characters	Accuracy of correctly recognized characters (%)	Confused with class
अ	128	114	89.06	ध(1),ग(6), म(1), ण (4), श(1), य (1)
आ	74	72	97.29	अ(1), ए(1)
ए	30	19	63.33	द(1), ड (1), घ(1), ण (1), प(4), त(1), उ(1), य (1)
ब	48	35	72.91	ळ (4), न(2), व(7)
भ	37	37	100	—
च	138	134	97.10	द(1), प(1), य (2)
छ	7	4	57.14	ए(1), ज(1), ळ(1)
द	98	83	84.69	ए(2),च(1), छ(1), ड(1), ई(1), प (1), र(3), त (1), ट(1), उ(2), य(1)
ड	55	49	89.09	द(1), इ(1), ळ(2), व(2)
ध	39	32	82.05	ब(1), क्ष(1), म(2), य(3)
ढ	12	10	83.33	ड(2)
ज्ञ	5	2	40	अ(1), ज(1)
फ	40	39	97.50	ज(1)
ग	75	67	89.33	अ(2), ब(1), च(4), ज(1)
घ	31	31	100	—
ह	65	62	95.38	ड(1), इ(2)
इ	25	22	88	द(3)
ई	12	10	83.33	च(1), इ(1)
ज	48	45	93.75	फ(2), ग(1)
झ	16	11	68.75	अ(3), आ (1), ध (1)
क	203	200	98.52	ळ(1), श(1), म(1)
ख	99	88	88.88	अ(2), ध (1), क(1), म(4), श(3)
क्ष	23	19	82.60	ख(1), श(3)
ल	168	160	95.23	च(1), क(2), म(3), त(2)
ळ	34	25	73.52	ब(6), न(2), व(1)
म	129	115	89.14	भ(1), क(1), ल(8), य (4)
न	172	161	93.60	ड (4), व(7)
ण	83	73	87.95	अ(2), ळ(2), न(1), प(1), व(3), य (1)
प	68	65	95.58	ए(1), च(1), य (1)
र	86	81	94.18	ए(1), ट(4)
स	84	84	100	—
श	31	21	67.74	अ(4), ख(1), ण(3), स(1), य(1)
त	245	244	99.59	ट(1)
ठ	36	34	94.44	ड (1), व(1)
त्र	9	4	44.44	अ(3), म(1), य(1)
ट	41	36	87.80	र(4),ठ (1),
उ	32	28	87.50	ए(2), ड (1), व(1)
व	168	164	97.61	द(1), न(1), ठ (2)
य	108	96	88.88	अ(1), च(2), म(4), ण(2), स(2), त्र(1)

using all the features and simple majority voting on the classifiers. Convolution neural network automatically finds the best features from the image dataset. Jangid *et al* [22] presented such work using layerwise deep convolutional neural network (DCNN) with the character recognition accuracy of 98%. Authors have also applied proposed algorithm on the benchmark databases like ISIDCHAR and V2DMDCHAR and achieved good recognition accuracy of 97.7% and 89.4% respectively. Table 4 represents the comparison of proposed work with existing works and benchmark databases.

8. Complexity of the algorithm

Complexity of proposed algorithm totally depends upon the finite step by step list of the instructions for solving the particular problem or getting required results. This is the

Table 4. Comparison of proposed work with existing works and benchmark databases (ISIDCHAR, V2DMDCHAR).

Authors	Database used	Features extracted	Classifier	Accuracy (%)
Narang *et al* [21]	Devanagari ancient text	Intersection and open endpoint, Centroid, Horizontal peak extent, Vertical peak extent	Simple majority voting on MLP, NN, CNN, decision tree, random forest	88.95
Jangid *et al* [22]	ISIDCHAR + V2DMDCHAR	Handwritten devanagari characters	Layerwise DCNN	98
Proposed work	Devanagari text (Our database)	DCT zigzag coefficients, Geometric features, Image moments	Linear Discriminant Analysis (LDA)	93.6
Proposed work	ISIDCHAR	DCT zigzag coefficients, Geometric features, Image moments	Linear Discriminant Analysis (LDA)	97.7
Proposed work	V2DMDCHAR	DCT zigzag coefficients, Geometric features, Image moments	Linear Discriminant Analysis (LDA)	89.4

function which gives the running time or memory requirement in terms of input size of the handwritten characters. As the database consists of different number of characters from different classes, for each class complexity of the algorithm depends upon the n number of images which are two dimensional. So, for each class as per as consecutive loops (for loop within for loop) in the algorithm are considered, quadratic time complexity is $O(k * m * n)$ where k is number of images in each class, m is number of rows and n is number columns of an image. Also, authors have calculated the execution time of the proposed algorithm which is 2.45 seconds using MATLAB2018b on the machine of HP Intel Core i7 processor.

9. Conclusion and future scope

In this work authors have reported a character recognition system for the handwritten Devanagari documents written from the young students of age less than 15 years. The documents were collected from the school notebooks and characters are segmented from them. Total 129 (58 from dilated image + 71 from skeleton image) features are considered for recognition using 5 classifiers viz., Linear Discriminant Analysis (LDA) (Classifier1), Support Vector Machine (SVM) with Quadratic kernel (Classifier2), Subspace Discriminant Analysis (SDA) (Classifier3), Subspace K-Nearest Neighbours(SKNN) (Classifier4) and Weighted K-Nearest Neighbours (WKNN) (Classifier5). Accuracy of 93.6% was achieved using Linear Discriminant Analysis (LDA) classifier on our database with the execution time of 2.45 s. On the benchmark databases like ISIDCHAR and V2DMDCHAR, we have achieved the accuracy of 97.7% in 3.67 s and 89.4% in 4.07 seconds respectively. Only individual vowels and consonants are considered for the classification in this work. In the future, authors will concentrate on classification of conjuncts and compound characters. New techniques for skew and slant correction will be developed and applied for better accuracy. Also, more innovative features will be considered using hybrid classification techniques.

Acknowledgements

The authors are thankful to the ISI, Kolkata for providing the handwritten Devanagari characters database.

References

[1] Bera S K, Chakrabarti A, Lahiri S, Smith E H B and Sarkar R 2019 Normalization of unconstrained handwritten words in terms of Slope and Slant Correction. *Pattern Recognition Letters* 128: 488–495

[2] Ding X 2012 Advances in Character Recognition. *BoD-Books on Demand*

[3] Bhalerao M, Bonde S and Vaidya M 2019 Frequently Used Devanagari Words in Marathi and Pali Language Documents. In: *Proceedings of the Advances in Computer Communication and Computational Sciences* 97–110

[4] Bhattacharya U and Chaudhuri B B 2005 Databases for research on recognition of handwritten characters of Indian scripts. In: *Proceedings of the Eight International Conference on Document Analysis and Recognition (ICDAR'05)* 789–793

[5] Dongre V J and Mankar V H 2012 Development of comprehensive Devnagari numeral and character database for offline handwritten character recognition. *Int. J. Applied Computational Intelligence and Soft Computing*

[6] Yadav M, Purwar R K and Mittal M 2018 Handwritten Hindi character recognition: a review *Int. J. IET Image Processing*12(11): 1919–1933

[7] Chaudhuri A, Mandaviya K, Badelia P and Ghosh S K 2017 Optical character recognition systems In: *Proceedings of the Optical Character Recognition Systems for Different Languages with Soft Computing* 9–41

[8] Sharma R and Mudgal T 2019 Primitive feature-based optical character recognition of the Devanagari script. In: *Proceedings of the Advanced Computing and Intelligent Engineering* 249–259

[9] Narang S R, Jindal M K and Kumar M 2019 Devanagari ancient character recognition using DCT features with adaptive boosting and bootstrap aggregating.*Int. J. Soft Computing* 23(24): 13603–13614

[10] Narang S R, Jindal M K and Kumar M 2019 Drop flow method: an iterative algorithm for complete segmentation of Devanagari ancient manuscripts. *Int. J. Multimedia Tools and Applications* 78(16): 23255–23280

[11] Kumar M and Jindal S R 2020 A study on recognition of pre-segmented handwritten multi-lingual characters. *Int. J. Archives of Computational Methods in Engineering*. 27(2): 577–589

[12] Puri S and Singh S P 2019 An efficient hindi text classification model using svm. In: *Proceedings of the Computing and Network Sustainability*. 227–237

[13] Impedovo D and Pirlo G 2014 Zoning methods for handwritten character recognition: A survey. *Int. J. Pattern Recognition* 47(3): 969–981

[14] Bansal V and Sinha R M K 2002 Segmentation of touching and fused Devanagari characters. *Int. J. Pattern recognition* 35(4): 875–893

[15] Ghosh R, Vamshi C and Kumar P 2019 RNN based online handwritten word recognition in Devanagari and Bengali scripts using horizontal zoning. *Int. J. Pattern recognition*. 92: 203–218

[16] Blumenstein M, Verma B and Basli H 2003 A novel feature extraction technique for the recognition of segmented handwritten characters. In: *Proceedings of the Seventh IEEE International Conference on Document Analysis and Recognition, Proceedings*. 137–141

[17] Hu M K 1962 Visual pattern recognition by moment invariants. *IRE transactions on information theory* 8(2): 179–187

[18] Ashour A S, Guo Y, Hawas A R and Xu G 2018 Ensemble of subspace discriminant classifiers for schistosomal liver fibrosis staging in mice microscopic images. *Int. J. Health information science and systems*. 6(1): 1–10

[19] Ho T K 1998 Nearest neighbors in random subspaces. In: *Proceedings of the Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*. 640–648

[20] Aggarwal C C, Hinneburg A and Keim D A 2001 On the surprising behavior of distance metrics in high dimensional space. In: *Proceedings of the International conference on database theory*.420–434

[21] Narang S, Jindal M K and Kumar M 2019 Devanagari ancient documents recognition using statistical feature extraction techniques. *Int. J. Sādhanā*. 44(6): 1–8

[22] Jangid M and Shrivastava S 2018 Handwritten Devanagari Character Recognition Using Layer-Wise Training of Deep Convolutional Neural Networks and Adaptive Gradient Methods. *Journal of Imaging* 4(2): 1–14