Multimedia Tools and Applications (2022) 81:10469–10495
https://doi.org/10.1007/s11042-022-12148-z

Offline Odia handwritten character recognition with a focus on compound characters

Raghunath Dey[1] · Rakesh Chandra Balabantaray[1] · Sanghamitra Mohanty[2]

Received: 4 June 2021 / Revised: 20 October 2021 / Accepted: 3 January 2022 /
Published online: 16 February 2022

Abstract

Recognition of Odia character images is one of the ongoing applications of offline OCR. An attempt has been made here to develop an efficient feature extraction procedure that can assist the recognition of Odia handwritten digits, basic characters, and compound characters. Three different kinds of strategies have been carried here for character recognition. First, the various characters are recognized using three feature extraction procedures individually, followed by a merged feature set with a set of standard machine learning algorithms. In the second approach, the recognition of the characters is performed by popular RNN and CNN by providing the same feature set instead of giving immediate images, unlike traditional networks. The same feature sets, as well as classifiers, are applied to recognize different categories of characters. The third task was to incorporate a set of Odia compound characters into our suggested character recognition framework. The dataset that has been created for this purpose consists of numerals, basic characters, and compound characters. The proposed method achieves a recognition accuracy of 86.56% on this dataset with 112 classes of characters.

Keywords Bounding box · Median blur filtering · Odia compound characters · Machine learning classifiers

1 Introduction and motivation

A machine can recognize handwritten text using optical character recognition (OCR). There are two ways to recognize handwritten texts: online and offline. In the case of online, Some

✉ Raghunath Dey
c118003@iiit-bh.ac.in

Rakesh Chandra Balabantaray
rakesh@iiit-bh.ac.in

Sanghamitra Mohanty
sangham1@rediffmail.com

[1] International Institute of Information Technology Bhubaneswar, India

[2] Sri Sri University, Cuttack, India

electronic devices can be used to trace the writing direction when the writing is going on [24], such as online signature authentication [18]. On the other hand, offline handwriting identification performs the identification of text from scanned textual images. It includes the recognition of numbers from bank cheques [1], addresses from letters, etc. With the help of this, it is possible to save both space and time. Here, there is no information available about the movement of the pen tip, the trajectory, or the direction of the text line. Offline OCR is, therefore, more critical than online OCR. Some of the essential processes involved in the offline process are shown in Fig. 1.

Several scripts are used in India for various languages. There is considerably less amount of research towards OCR present in the literature to support handwritten text identification. Furthermore, it can be noted that most of the studies attempt to recognize Devanagari and Bangla handwritten characters in comparison to other regional languages [41]. Odia is an Indo-Aryan language. More than 35 million people in Indian states such as Odisha, West Bengal, and Gujarat speak Odia. Assamese and Bangla languages are mostly related to Odia. The Odia writing is originated from the Kalinga script and is one of the many offshoots of ancient India's Brahmi script. Since ancient times, palm leaves have been used in Odisha to record the state's literature and history. Furthermore, since 1838, the printing press has been available in Odisha. There are many historical events and manuscripts in Odia by several renowned writers like 'Fakir Mohan Senapati', 'Radhanath Ray', 'Madhusudan Das', Gopabandhu Das, etc. In recent decades, attempts have been made to preserve these as digital files. Saving the contents as a text file form is a better choice than a scanned copy of the pages to avoid running out of storage space. The manual conversion of this job is impractical. It's essential to automate the conversion of these image files to text files, which necessitates the use of offline OCR for Odia characters. The lack of Odia databases to train the OCR engine is the primary impediment. The performance of Odia OCR must be improved to fulfill the requirements of real-time recognition. It motivates strongly to donate a good quality of Odia characters and numeral dataset to the research community working in this domain. Handwritten character recognition is among the several popular applications of computer vision. Though, this job is not simple. Researchers are encouraged to enhance recognition results using various pattern recognition algorithms, which helped the offline text retrieval and digitization process.

In the early years of 2005, researchers began their research study on offline Odia OCR. Odia has 10 numerals, 49 basic characters. More than 100 compound characters can be formed with the help of these basic characters. This basic character consists of 12 vowels and 37 consonants. These can be seen from Fig. 2, where all digits, basic characters, and some of the commonly used compound characters in the handwritten form are presented. Each of the character classes is represented by a green number at the top of the characters.

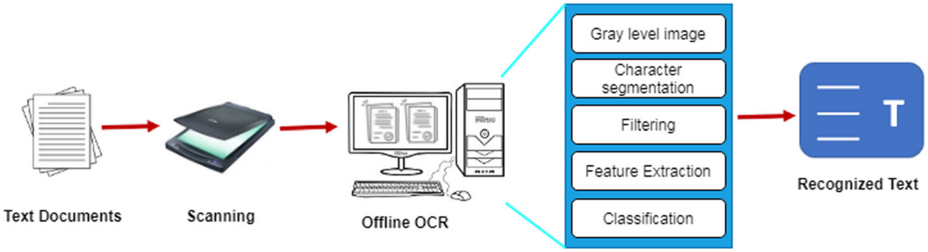


Fig. 1 Common steps involved in Offline optical character recognition

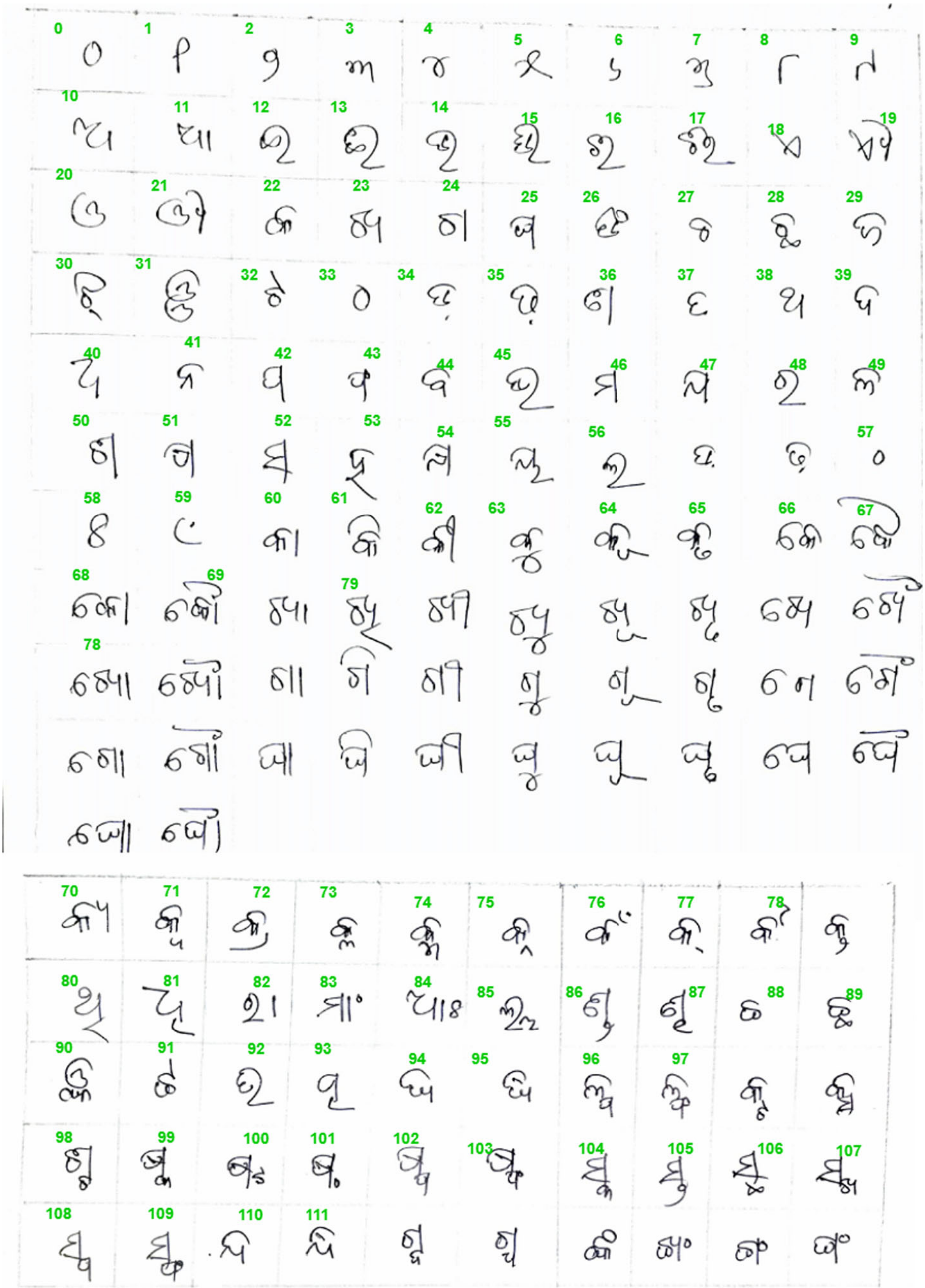


Fig. 2 Set of handwritten digits, basic characters and compound characters in Odia. Numbers mentioned in green colour at top position of the characters are the respective class labels present in the 'IIITBOdiaV2' database

These class labels are the class sequence present in the newly developed 'IIITBOdiaV2' database created in IIIT Bhubaneswar. The character groups numbered from 0 to 9 are the digit classes. The numbers 10 to 56 represent basic characters, and 57 to 59 represent special modifiers. There are two characters between classes 56 and 57 that are ignored from the list of basic characters in this database because they are very similar to classes 34 and 35, respectively. The modified characters of 'Ka' are represented by the character classes 60 to 69. Some of the most common compound characters are 70 to 111, except for 80, 81, 82, and 84. Class labels 80, 81, 82, and 84 are modified characters similar to class labels 60 to 69. Currently, the compound characters labeled 70 to 111 are present inside the database, though there are more than a hundred compound characters in the script. These are not included in this database due to insufficient collections of characters.

Though very little research has been conducted on the recognition of offline Odia characters, most of them are on Odia printed character recognition out of the literature. The recognition results in terms of accuracy of these printed characters are also very satisfying. The account of the research works becomes further smaller in the case of handwritten Odia character recognitions. Again, the majority of them are on handwritten numerals, but there are very few on handwritten characters. The recognition outcomes of the handwritten characters are truly very disappointing. Finally, it becomes zero in count in the case of handwritten compound character recognition. Many factors contribute to this, including allograph complexity, a scarcity of datasets, and a scarcity of commercial sectors. Smartphones in India are expanding exponentially in all the regions of India, which will lead to heavy man-computer interactions in the future. OCR developments for Indian scripts, including popular scripts such as Odia, will be in high demand. As a result of these factors, a high-quality Odia handwritten text with various modifiers and compound character recognizers is now required. It necessitates the collection of a standard handwritten Odia database consisting of different possible compound characters. The current study proposes a character database as well as a hybrid feature extraction technique. The neural networks models are designed to take extracted features as input instead of using direct images. It would speed up the character recognition task because it would take less time to process the extracted features than compared to immediate images.

1.1 Characteristics and complexities of Odia scripts

Unlike English, the Odia script does not have separate uppercase and lowercase characters. The Odia script has a vast number of simple and complex characters. Each of the Odia characters has various components with different characteristics. Most of them are circular in their upper sections and have perpendicular straight lines on the right side of the letters. The modified characters are of different shapes and sizes based on the 'matras' or 'kaara' used. Depending on the vowel, these characters can be small or big, and the 'matras' can occur on any of the four sides of a consonant character. Some of them are shown in Fig. 2, from class '60' to '69' and many more, which are not labeled here. When a consonant or a vowel is combined with another consonant, then a compound character is formed. Some of the formations of compound characters are shown in Fig. 3.

Several problems that an Odia character recognition device could face. Those are as follows:

1. Due to roundish, identical, and sometimes similar characters, distinguishing character class becomes difficult.

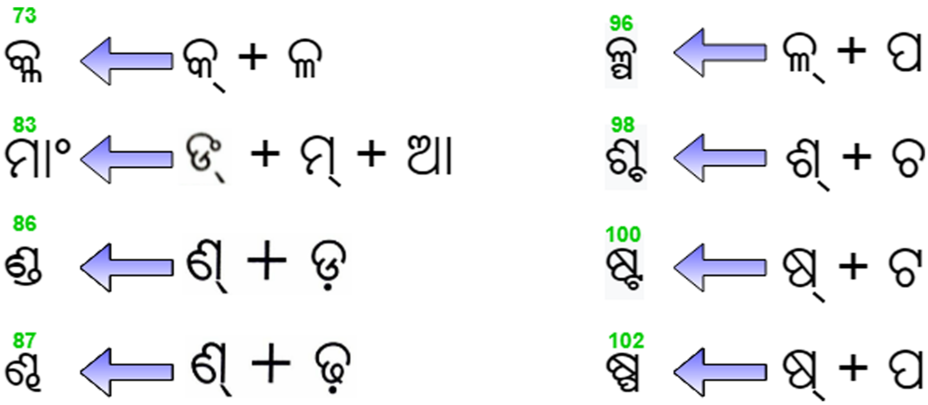


Fig. 3 Some examples of Odia compound characters using ‘juktas’ or ‘conjucts’. Numbers mentioned in green colour at top position of the characters are the respective class labels present in the ‘IIITBOdiaV2’ database

2. There is no clear and fixed rule for the interference of modifiers with their base character. It is a significant roadblock for word-to-character segmentation.
3. The structure of mirror images between two characters becomes one more challenge. The flipping of one letter, may look like another character. The characters ‘Ga’ and ‘Ma’ become ‘Tha’ and ‘Dha’, respectively, when they are vertically flipped. It is shown in Fig. 4.
4. The absence of benchmark databases is another weakness in recognizing Odia handwritten characters. There are currently only three publicly accessible databases. These are provided by, Indian Statistical Institute in Kolkata, NIT Rourkela, and IIT Bhubaneswar.

The implementation of a handwritten character recognition system necessitates a massive number of training samples. It’s also difficult to generate a dataset of this size because it takes a lot of time and effort. Only three extensive handwritten datasets have been developed in recent years. That includes a set of character images from handwritten Odia numerals and basic characters only. That is why an attempt has been made to build a database for Odia handwritten characters, including some widely used modified characters and compound characters.

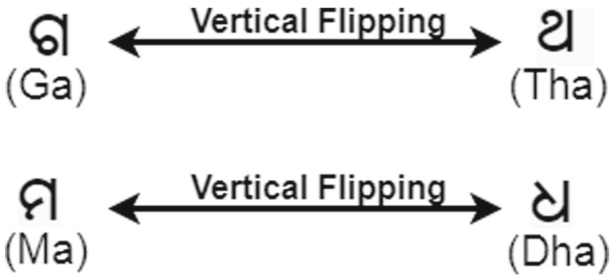


Fig. 4 Two examples of characters when flipped vertically produce another characters

2 Related works done

There are around 35 papers in terms of conferences and journals are present in literature for recognizing handwritten Odia characters in an offline manner. It is very little in comparison to other regional languages of India, like Gurumukhi, Bangla, Tamil, etc. All this research was done within the last two decades. In most of the papers, authors have used some synthetic handwritten character datasets by creating them on their own instead of any benchmark datasets. It happens due to the lack of availability of benchmark datasets in the public domain. Only three datasets are there which are made publicly available for research purposes. However, all these datasets do not provide all the categories of characters publicly. These are limited to digits and basic characters. Here in our literature study, we have emphasized those papers which are based on evaluating a set of benchmark datasets. Because using a benchmark dataset in the experiments is the best procedure to evaluate a recognition mechanism.

In the initial portion of the literature of Odia handwritten character recognition, it was found that some studies were performed on the *ISI Kolkata* numeral dataset. Tripathy et al. 2003 [42] suggested a method to recognize the numerals, and the authors have used threshold-based binarization as a primary preprocessing technique. Features are extracted based on reservoir area and location, the path of water flow, the number of loops, the center of gravity, ratio between reservoir and loop, profile-based features, on jump discontinuity. Finally, using a binary tree classification approach, the accuracy obtained was around 97.74%. The paper Roy et al., 2005 [32] suggested computing region of interest (ROI) based on bounding box and segmented the characters into blocks. The feature extraction was based on a chain code histogram having a dimension of 400. By applying neural networks and quadratic classifiers, an accuracy of 94.81% was reported in the study. Except for these preprocessing techniques, few others like Gaussian filter, Roberts filter are applied in [29] and accuracy of 98.54% was achieved using the same quadratic classifier.

In [4], the authors implemented binarization, thresholding, and normalization of the digit images in the preprocessing phase. Applying the Hidden Markov model (HMM) on extracted features like horizontal and vertical strokes, the accuracy of 90.50% was obtained. In [20], only Gaussian filter and resize were performed inside the preprocessing phase. The Zernike moment features were extracted and used on an ensemble of MLP based on Multiclass Adaboost, and finally, the recognition rate of 97.10% was found. The Otsu threshold and mean filter were implemented during the preprocessing step in the paper [9]. Following that, all of the samples were resized to 72 × 72. The kirsch gradient operator and curvature were used to extract features. The feature dimension was reduced using PCA. For classification, a modified quadratic discriminant function and its discriminative learning version were used. It produced an accuracy of 98.5%. In [34] authors merged their own created numeral dataset with the standard ISI Kolkata dataset. After this, LU factorization-based feature and naïve Bayes classifier were used for classification that helps to produce an accuracy of 92.75%. Some preprocessing methods, such as filter-based noise reduction and connected component analysis, were used in [11]. Following that, all of the digit samples were resized to a 24 × 24 format. On SVM, HMM, Bayes classifier, quadratic discriminant function, and kNN classifiers, different features such as Slantlet, Stockwell, and Gabor wavelet-based transformations were used. The highest level of accuracy was attained, with a score of 98.8%. Binarization, thinning, size normalization, and dilation techniques were used in the preprocessing phase on only 500 samples of the ISI Kolkata dataset by Sethy et al. 2016 [36]. The discrete cosine transform was used to extract features, which were then added to the ANN, resulting in 90% accuracy.

On the ISI Kolkata dataset, Majhi et al., 2018 [25] used preprocessing techniques such as a median filter and canny edge detection. After that, all of the samples were resized to 64 × 64. Using discrete Fourier, short-time Fourier, discrete cosine, discrete wavelet, S-transform, and curvelet of digits transformations, features were extracted. Using PCA, the feature dimension was reduced to 32. MLP, artificial neural network, radial basis function network, and probabilistic neural network were among the classifiers used. Finally, a 98.70% recognition rate was reported. In [6], only binarization and resize to 28 × 28 to whole ISI Kolkata datasets was performed. The total number of inputs became 784-pixel values for the suggested extreme learning machine. It gave an accuracy of 94.47%.

In [13], the only preprocessing added to the ISI Kolkata and IIT Bhubaneswar dataset is segmentation. The feature, based on sparse concept coded tetrolets was applied to RF, SVM, kNN, and modified quadratic discriminant function classifiers (MQDF). Finally, the accuracy of IIT Bhubaneswar digits and characters was 98.72% and 93.24%, respectively. Similarly, the ISI Kolkata dataset's digit classification accuracy was found to be 99.22%. In [40] also, the only preprocessing applied to the same datasets was size normalization. Then the images are supplied to a CNN model. It achieved up to 98.4% on the ISI Kolkata dataset and 97.71% on the IIT Bhubaneswar digit dataset.

Some preprocessing techniques, such as finding ROI using a bounding box and binarization using the otsu threshold, were performed on digit samples of the IIT Bhubaneswar dataset [10]. Following that, the samples were divided into nine zones. Non-redundant Stockwell transforms, Slantlet coefficients were extracted and used in GA, PSO, and Differential Evolution based optimization with kNN. Finally, on the ISI Kolkata and IIT Bhubaneswar digit datasets, 99.1% and 98.6% accuracy were obtained, respectively.

In [12], Denoising, Size Normalization, Binarization, and Morphology based operations were applied to the IIT Bhubaneswar dataset. Features like Stockwell transform, the weighted gradients were extracted and were studied on kNN and modified quadratic discriminant function (MQDF). It gave an accuracy of 99.1% on numerals and 95.14% on basic characters of the IIT Bhubaneswar dataset.

In [5], preprocessing techniques like Binarization, pruning, and dilation was applied to the IIT Bhubaneswar digit dataset. After that, the samples were normalized to 40 × 40. Features extracted through convolution layers whose weights are upgraded by JAYA optimization. Then these are applied to the Random Forest classifier, which produced an accuracy of 98.25%.

Only size normalization was used on the OHCSv1.0 (NIT Rourkela) and ISI Kolkata datasets in [7]. Convolution layers with a multi-objective Jaya-based optimized network were used to extract features. Then these are applied to SVM and Random forest. It produced an accuracy of 98.9% towards recognizing characters of the NIT Rourkela dataset using RF and 97.70% on the ISI Kolkata numeral dataset using the SVM classifier.

In [39], Size normalization, median filter, skeletonization like preprocessing were applied to ISI Kolkata and OHCSv1.0 (NIT Rourkela) dataset. Only 200 samples from each character class and 300 samples from each digit class were taken. Features were extracted using row symmetry and column symmetry chords. Then these were applied on the Decision Tree classifier and achieved a recognition rate of 96.2% on ISI Kolkata numerals and 95.6% on the OHCSv1.0 Character dataset. In [27], authors performed classification on 57 character classes of the OHCSv1.0 (NIT Rourkela) dataset. The functions like Binarizarion,

Table 1 Details of the publicly available Odia handwritten character datasets

Datasets	Contents	Labels	Total Samples	Dimension	No of writers
ISI Kolkata	Numerals	10	5970	variable	356
IIT Bhubaneswar	Numerals	10	5000	variable	500
IIITBOdiaV2	Numerals	10	2962	64 × 64	150
NITROHCS_V1.0	Characters	47	15040	64 × 64	150
IIT Bhubaneswar	Characters	70	35000	variable	500
IIITBOdiaV2	Characters	102	13236	64 × 64	150

Skew Detection & Correction, and Segmentation are performed in the preprocessing phase. Features based on the horizontal histogram and vertical histogram are collected and applied to the classifiers like SVM, kNN, and BPNN. Finally, the highest accuracy achieved was 83.75%.

The authors of [38] in 2018, used normalization and dilation to preprocess the NIT Rourkela dataset. From each of the 47 classes, only 150 specimens were taken. The discrete wavelet transform was used to extract features, which were then reduced using PCA. The feature set was provided to BPNN, and 94.8% accuracy was obtained. In 2019, the authors of [37] implemented noise reduction, skew correction, and normalization towards preprocessing on the NIT Rourkela dataset. All 350 samples are taken from every 47 classes. The extraction of features was based on symmetric axis chords, mathematical features such as Euclidean distance and Hamilton distance, and the feature dimension was then reduced using PCA. These were applied to a Gaussian kernel with a radial basis function neural network and achieved a recognition rate of 98.8%.

3 Database used for simulation

Experiments are carried out on the three preexisting benchmark datasets and one newly created dataset. Compound characters aren't present in any existing databases. As a result, we have been motivated to create a new dataset with compound characters. ISI Kolkata[1] Digit, NIT Rourkela's NITROHCSV1.0[2], and IIT Bhubaneswar[3] are the only currently available datasets. IIITBOdiaV2[4] is the name of our most recently created dataset. The following sections provide an overview of each of the datasets. The details of the publicly available datasets, including the recently created dataset, are elaborated in Table 1.

3.1 ISI Kolkata digit

The ISI Kolkata Odia database [3] contains 5,970 handwritten Odia numerals from 356 authors. The database was built using 105 emails, 166 job letters, and various forms. From the available data, a training set of 4970 samples and a test set of 1000 samples were

[1] https://www.isical.ac.in/~ujjwal/download/OriyaNumeral.html

[2] https://www.iitbbs.ac.in/profile.php/nbpuhan/

[3] https://nitrkl.ac.in/CS/Datasets.aspx

[4] https://www.iiit-bh.ac.in/academics/research/funded-projects/clia

generated. Each character has 100 copies in the test set, while imbalanced units are there in the training set. The samples come in a variety of sizes.

3.2 NITROHCSV1.0

One of the databases for handwritten Odia characters is NIT Rourkela's NITROHCSV1.0 [27]. Binarization and segmentation were used to preprocess the samples. Each specimen was archived after regular standardization. The samples of the dataset are developed by 150 people regardless of gender, age, and education level. They wrote 57 characters on a series of formatted sheets, including ten numbers and 47 basic characters on each. Developers have used many machinery equipments to eliminate paper clattering and bursting.

3.3 IIT Bhubaneswar

Samples from the IIT Bhubaneswar [12] database were obtained in several locations and weather conditions. People were required to write in an unregulated environment in order to realize typo variations. Papers were scanned at both 600 dpi and 300 dpi. Each handwritten class of Odia numerals and characters has 500 entries in the database. There are a total of 80 classes, ten for numerals and 70 for characters. There are two types of written samples are collected. The characters are written first on clean 70 GSM pages, and then on predesigned grid arrangements of pages. The characters in the grid format are all the same height and divided by a fixed width.

3.4 IIITBOdiaV2

IIITBOdiaV2 is an Odia character database consisting of numbers and alphabets at the same time. The grid design on the papers was made and supplied to the 150 volunteers. All the volunteers' ages range from 5 to 70 years old. All sample copies were scanned at 300 dpi at IIIT Bhubaneswar using an HP scanner. After multiple preprocessing activities such as filtering and threshing, the characters are segmented and isolated from the parent texts. They are then fine-tuned to ensure that the samples for each class written by volunteers are spread evenly during the development and processing of the dataset. In the IIITBOdiaV1, there were ten numbers digit groups and fifty basic-character groups presented. In this study, a few popular compound characters, as well as modified characters, are added, and it became IIITBOdiaV2. There are currently 112 numbers of class labels present in the database. It includes ten numbers digit classes, 50 basic characters, and the rest 52 class groups include compound characters and modified characters. Through contacting the author by email, datasets can be accessible to anyone, provided these should be used only for scholarly purposes.

4 Database preprocessing

In every character recognition scheme, preprocessing is important. It becomes much more essential when we build a handwritten character recognition mechanism. A variety of impediments can be seen, such as noise, paper quality loss after use, poor writing styles of individuals, and so on. These are very common when recording a database of handwritten characters. To remove these flaws, several stages of preprocessing are needed. In the first step, we use preprocessing techniques like median blur filtering, bounding box, and dilation to build the 'IIITBOdiaV2' database. Other preprocessing procedures like thresholds, binarization, thinning and resize are carried out in the second stage before classification and identification.

4.1 Median blur filter

The median blur filter [21] is popular among order statistical filters for its high efficiency and effective rate in eliminating specific noise forms such as Gaussian and random salt and pepper noise. It replaces the middle pixel of the corresponding $P \times P$ window with the average value of each of the cells. The median differs considerably in the noisy regions that are present in the images. Here the median blur filter from Python OpenCV has been used with a 'P' value of '3' to minimize the noise pixels on the character images before segmentation. The requisite formula is shown in (1). Where, K_{pq} is the set of coordinates in a rectangular sub-image window which has a center at (m, n). The $D(m, n)$ is the restored destination image, and $S(p, q)$ is corrupted or source image and calculated area under the dimension of K_{pq}.

$$D(m, n) = medianBlur\left\{S(p, q)\right\}, \quad where(p, q) \in K_{pq} \tag{1}$$

4.2 Bounding box with dilation

The sheets issued to the volunteers are set up with uniform grids so that the character font sizes do not differ considerably. The next crucial step was to find a method that could determine the locations of the characters on the grid paper in the proper alphabetical order. The region of interest (ROI) was discovered using the bounding box technique [23]. However, the bounding box faces some flaws. The bounding box occasionally caught unwanted degraded grid lines. Another few issues were the uncompromising styles of characters and the construction of characters. Some of the characters are made up of many pieces. The bounding box selects the broken characters as well as the fragmented pieces. That is why a minimum contour length of '60' was used as a threshold to fix these problems. These errors can be corrected by the reduced threshold value, such as 60-pixel length, which includes eroded left grid lines. As a result, a few main components of certain characters with widths less than '60' have been omitted. The matra 'u' of the modified character 'Ku' has a contour of less than 60. The bounding box assumed that these fragments were distinct characters because their lengths were less than 60. Practically, these are not distinguishing characters; they are most likely to be letter fragments. The problem was solved by merging partial character components using a morphological dilation technique with kernel size 10. The bounding box considers these as a single entity and helps to find definite characters.

4.3 Thresholding and binarization

Character images must be correctly preprocessed before the recognition job to enhance its accuracy. For a gray-scale segmented character image, the intensity values of the pixels range from 0 to 255. In the text image, the values '255' and '0' signifies to 'bright' and 'dark', respectively. The intermediate values have a light-black or greyish appearance. A high-dimensional storage technique is required to represent and execute these. Text pictures, on the other hand, do not necessitate such representations. Text images typically have two values. The first is the text line itself, and the second is the rest of the background. As a result, when dealing with character images, only binary representations are required. The 'Otsu global thresholding' [28] method is used to generate this binarized version of the samples. The breakpoint has been set at the 'T' threshold. If the intensity value is more than 'T', the updated value will be 255 or white; otherwise, it will be '0' or black. All pixel values were divided by their highest value to standardize the images. After passing through the thresholding, the images are binarized [16], with '0' representing a black or background image and '1' representing a white for the text line.

4.4 Thinning

The thinning operation [2] was carried out after the binarization phase. The line width of the images varies from place to place, as the data is handwritten. Hence, a thin version of the same character image becomes necessary. Throughout the skeleton or thinned version of the character images, the thickness of the text line is one pixel everywhere. The thin version of a character image is shown in Fig. 5. All the scanned images will be thinned and supplied to the feature extraction procedure for further processing. However, the background of samples has been resized before obtaining their thinner versions to remove the existing unnecessary parts.

4.5 Resize

Initially, all of the samples from various datasets are variable in size and large. As a result, it will be computationally expensive. To standardize the collections retrieved from the different repositories, they need to be resized to a fixed dimension. So that, the requirement of time and space can be reduced to some extent. The image size cannot be so small that the clarity of the image is compromised. As a result, the 64 × 64 and 64 × 128 pixel scales were chosen as the best size for all samples from each dataset in various scenarios.

5 Feature extractions

Regardless of the writers, the extraction of features of handwritten text images should not be affected hugely. A proper recognition framework is necessary that can be used by anyone. In this case, a contour-based representation of the character is a preferably better choice. Because a character written by any writer may be of various sizes but may have an immense resemblance with respect to its structure. As a result of this, each recognition method's performance may improve. This section of the research gives a quick overview of three different feature extraction methods. These are: (I) the angular motion of a character, (II)

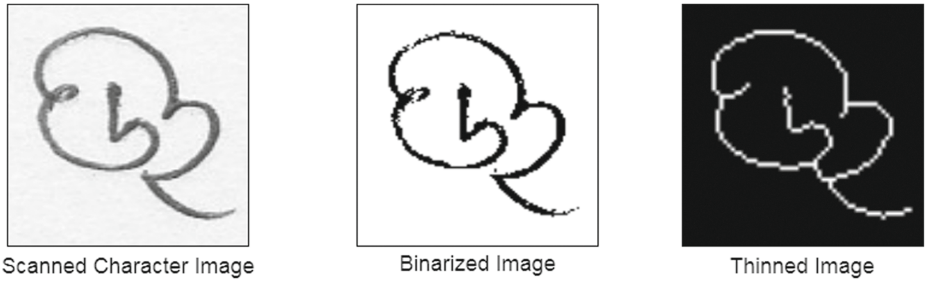


Fig. 5 Thinned version of a character followed by its binary version

the distance between the character text lines and their centers, and (III) the five types of slopes formed by the text line of a character.

5.1 AMS Feature

'AMS' is known as the Angular Motion of Shape base feature. This is an effective technique for finding features based on the shape or contour of a character. It believes that all the different classes of characters in any script produce different structures. That is why the AMS feature [15] for any class of characters must be unique. In such kind of feature extraction technique, all the images are fragmented into two zones. These are the upper zone and lower zone. In most of the characters, the upper portion of the character looks completely different than its lower portion. This helps in producing quite an identifiable feature from a character. After extracting the required features from every character image, the algorithm stores them in the AMS feature matrix. The necessary pseudo-code of the program is shown in Algorithm 1. Initially, a 3 × 3 window that moves vertically across two zones of each character image. This type of window is also called a sliding window [14]. For the upper zone, the row-wise and column-wise range values are shown in (2). Similarly, for the lower zone, the row-wise and column-wise range values can be seen in (3).

$$i_{upper} = 0 \; to \; \lceil row/2 \rceil - 1 \; and \; j_{upper} = 0 \; to \; col - 1 \tag{2}$$

$$i_{lower} = \lceil row/2 \rceil \; to \; row - 1 \; and \; j_{lower} = \; 0 \; to \; col - 1 \tag{3}$$

As the character images are converted from gray-scale to binary version, that is why the images have two values, which are 1 and 0. In our case, '1' signifies text, whereas the '0' signifies background. When the sliding window moves throughout the thin version of the character image and finds a '1' in its center position, it looks after the other eight adjacent cells. Otherwise, the window ignores the contents. The eight adjacent cells are named with alphabets from 'a' to 'f'. Depending on the presence of '1' intensity value in the character image fitting on the sliding window, it generates the code, as shown in Fig. 6. The number of possible combinations of alphabets formed from the eight cells of the sliding window is $2^8 = 256$. This value is completely based on the concept of set theory. If there are 'n' numbers of characters in a set, then the total number of combinations of characters that can be formed is 2^n. These codes of alphabet combinations will be repeated as the window moves throughout the shape of a character. One thing that can be noted is that the window moves only from left to right in a vertical manner. That is why the number of codes will be 256/2=128, instead of 256. This value of 128 is only for one zone out of two zones. So for two zones, this value will be $128 \times 2 = 256$. All these 256 alphabet combination codes were stored in a matrix called 'Headermatrix'. All of the codes from this matrix must be copied as column headers to the AMS feature matrix. As a result, whenever an alphabet combination

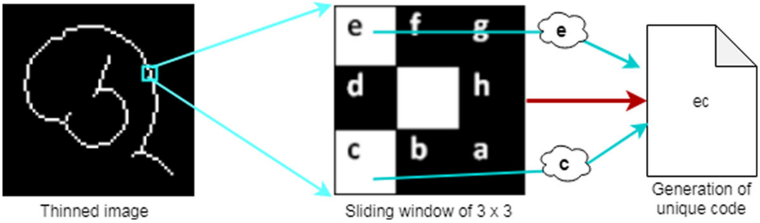


Fig. 6 Code generation using a sliding window of size 3 × 3

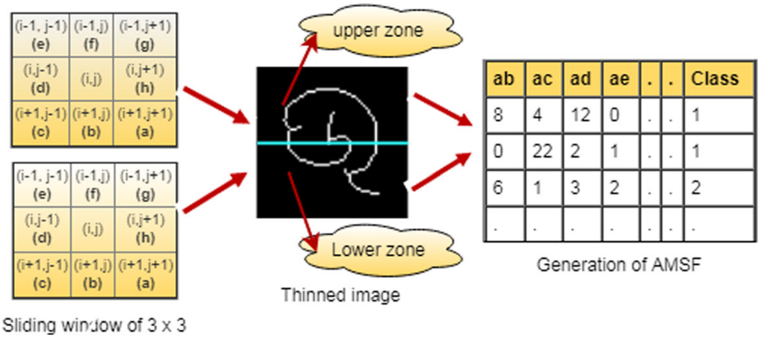
is matched to one of the column headers of the feature matrix, the frequency count of the relevant code is increased. The final updated value for that column concerning the sequence of rows occupied by the sample in the feature matrix is stored, as shown in Fig. 7. Hence, the numbers shown in the table of Fig. 7 represent the frequency or the number of occurrences of the alphabet combination present at the corresponding column header. The whole process is shown in Algorithm 1. The algorithm confirms that getting the frequency for all codes from two zones for all samples is an iterative process. There will be 257 columns in the final version of the AMS feature matrix, including the last column for the class label. In the final version of the AMS feature matrix, there will be 257 columns which include the class label in the last column. The total number of rows will be the number of samples present in the particular database.

Algorithm 1 AMS based Feature extraction.

```
Input: $Img_1 \dots Img_N \in DB$
Initialize: The Header-matrix, a total of 256 number column headers with class.
$i \leftarrow 1$
repeat
    Divide each $Img_i$ into two zones, $z_1$ and $z_2$    ▷ $z_1 \leftarrow$ upper zone and $z_2 \leftarrow$ lower zone
    $j \leftarrow 1$
    repeat
        Check value 'v' at each cell of $z_j$
        if $v == 1$ then
            Check it's adjacent cell value $v_{adj}$
            if $v_{adj} == 1$ then
                Generate Code as in Fig. 6
                Increment code[frequency]
            end if
        else
            Continue
        end if
        Update Header-matrix[code]
        $j{+}{+}$                                   ▷ For each zone
    until $j \leq 2$
    $i{+}{+}$                                       ▷ For each sample
until $i \leq N$
```

5.2 CTD Feature

This is also called the ‘Center to Text Distance’ feature [15]. Most of the characters of any script have different shapes and distances from their centers. The characters that belong to a particular class may have a similar amount of distance from the center of the text. With regards to this fundamental idea, this CTD feature extraction is designed. There ‘8’ different distances are computed from the center based on ‘8’ different angles. The details of these ‘8’ distances $\{R_x, R_u, R_b, L_x, L_u, L_b, T_y, and\ By\}$ can be understood from the Fig. 8. After applying various preprocessing methods as discussed in Section 4, the final binarized thin version of the samples was collected. These are supplied to the CTD feature matrix computation program. The required pseudo code is shown in the Algorithm 2. Initially,



(i-1, j-1) (e)	(i-1,j) (f)	(i-1,j+1) (g)
(i,j-1) (d)	(i,j)	(i,j+1) (h)
(i+1,j-1) (c)	(i+1,j) (b)	(i+1,j+1) (a)

(i-1, j-1) (e)	(i-1,j) (f)	(i-1,j+1) (g)
(i,j-1) (d)	(i,j)	(i,j+1) (h)
(i+1,j-1) (c)	(i+1,j) (b)	(i+1,j+1) (a)

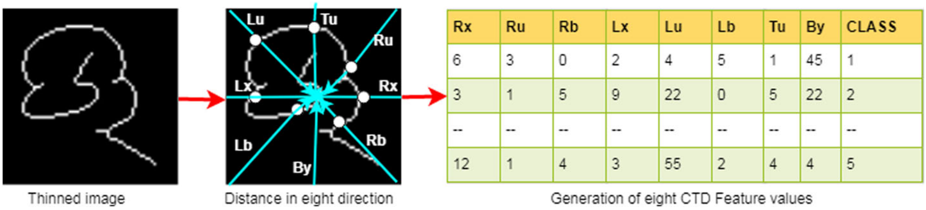
ab	ac	ad	ae	.	.	Class
8	4	12	0	.	.	1
0	22	2	1	.	.	1
6	1	3	2	.	.	2
.	.	.	.	.	.	.

Fig. 7 Extracting AMS feature from two portion of character image using sliding window

the center of each character image is calculated. From this center, the controller will move through these ‘8’ directions. Wherever there is a white text pixel fetched in any of the ‘8’ directions, the corresponding distance from the cross-section point to its center is computed. These distances are calculated using the Euclidean distance method. These distance values are all floating-point values and all are stored in the CTD feature matrix.

Algorithm 2 CTD based Feature extraction.

```
Input: $Img_1 \dots Img_N \in DB$    ▷ $Img_1 \dots Img_N$ are the thinned version of input images
for each $Img_i \in \mathcal{DB}$ do
    COMPUTELENGTH($R_x, R_u, R_b, L_x, L_u, L_b, T_y, By$)
end for
function COMPUTELENGTH(X)
    for each cell $cell_i \in \mathcal{I}mg_i$ from Center to X position do
        if value($cell_i$)==1 then
            length L= Euclidean_Length($cell_i$, Center)
        else
            length L= 0
        end if
    end for
    return $L$
end function
Output: Length from eight different positions on text line to the center of character
```



Rx	Ru	Rb	Lx	Lu	Lb	Tu	By	CLASS
6	3	0	2	4	5	1	45	1
3	1	5	9	22	0	5	22	2
--	--	--	--	--	--	--	--	--
12	1	4	3	55	2	4	4	5

Lu
Tu
Ru
Lx
Rx
Lb
By
Rb
Thinned image
Distance in eight direction
Generation of eight CTD Feature values

Fig. 8 Eight directional distance values from center of characters

5.3 FoG feature

This Five orientation gradient (FoG) based feature is based on the gradient or slope occupied by a thin line of a text image. It also depends on the contour of a character, like the AMS and CTD features. An image sample is a two-dimensional array, where each cell of all rows and columns of the matrix consists of pixel intensity values. For this type of feature extraction, a 2 × 2 window is used that will move horizontally from left to right. Initially, the character image samples are binarized and resized to 64 × 128 followed by the median blur filtering. There are four zones column-wise but eight zones row-wise. Hence, each image sample is divided into 32 (8 × 4 = 32) equal zones, where each zone has the size of 16 × 16. These 32 numbers of zones representation on a character is shown in Fig. 9 with an ordering of 1 to 32. When the window moves, then at least two cells are expected to have a white or '1' pixel value as their content. If at least two cells of the window have white intensity, then the algorithm is designed to generate features for the characters. In this scenario, a single pixel's existence was ignored. Because, after an initial preprocessing with the 'median blur' function, the appearance of a single-pixel vanishes inside the samples. The addition of a one-pixel case to the feature set increases its dimension unnecessarily. Hence this is eliminated.

It's important to note that the maximum number of gradients or slopes that can be created through the 2 × 2 window is five, as seen in Fig. 10. Two types of intensity values, such as black and white, are present in the sliding window. A pixel of written text is represented by white or '1', while the backdrop is represented by black or '0'. In general, there are three scenarios faced by the sliding window. (1) if only two cells have white pixels, (2) if all four cells have white pixels, and (3) if only three cells have white pixels.

The four forms of potential slope or gradient variants are collected in Case 1. From the four cells a, b, c, d, if 'b' and 'd' are both '1', a line with a forward 45° gradient is formed, as shown in Fig. 10a. The backward gradient type with a 45° angle is formed if each of the 'a' and 'c' cells has an integer '1', as seen in Fig. 10b. A horizontal gradient of 0° is generated

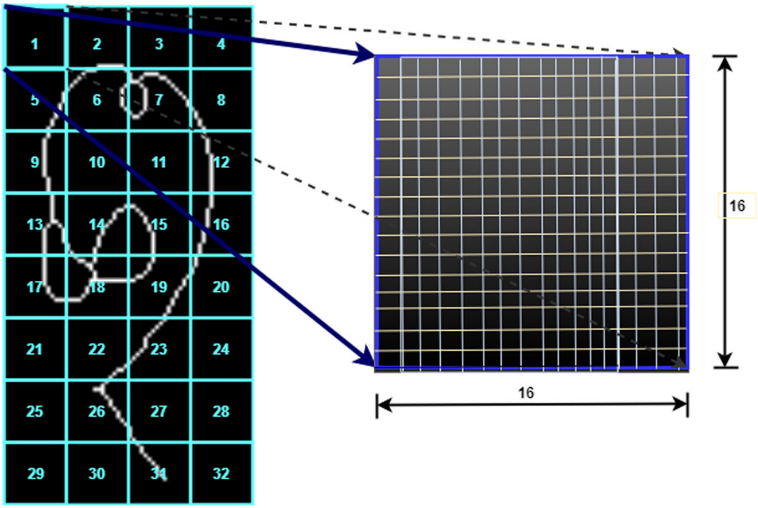


Fig. 9 Dividing each character sample into 32 equal zones, where the size of each zone is 16 × 16

if the positions a, d, or b, c are ‘1’, as seen in Fig. 10c. Similarly, cells a and b or d and c are filled with ‘1’, resulting in a vertical line with a 90° gradient, as shown in Fig. 10d.

The feature is extracted when all the four cells of the window are filled with a white or ‘1’. This kind of representation belongs to case 2. All the cells of the window are completely occupied by the thin text line of the character. It is also referred to as a “no gradient” state. Figure 10e shows its illustration. Likewise, when three cells are filled with white color, it belongs to case 3. Logically, this is the situation that arose by the merging of two different representations from case 1. After that, for each respective zone starting from 1 to 32 sequentially, the frequencies are calculated by moving the window horizontally. Afterward, all of the computed gradient frequencies are stored as a vector sequentially based on the zone numbers. Finally, five simple gradient types, including one with no gradient, have been considered in such a FoG type of feature representation. There are 32 regions in all. It is why the feature matrix is of 160 dimensions, i.e. $32 \times 5 = 160$. The steps for obtaining the Five Orientation Gradient (FoG) function are defined in Algorithm 3.

Algorithm 3 FoG based Feature extraction.

```
Input: $Img_1 \ldots Img_N \in DB$    ▷ $Img_1 \ldots Img_N$ are the thinned version of input images
Initialize: The FoG-Header matrix, a total of 160 number column headers with class.
for each $Img_i \in DB$ do
    resize $Img_i$ into $64 \times 128$
    Split each $Img_i$ into 32 zones, of $16 \times 16$ each
    $j \leftarrow 1$
    repeat
        for $p \leftarrow 1$ to 16 do                    ▷ Move a 2 × 2 window horizontally
            for $q \leftarrow 1$ to 16 do
                $f_1$=compute code of forward 45°
                $f_2$=compute code of backward 45°
                $f_3$=compute code of $Horizontal$ 0°
                $f_4$=compute code of $Vertical$ 90°
                $f_5$=compute code of $No\ gradient\ region$    ▷ $f_1, f_2, f_3, f_4, f_5$ are calculated using Fig. 10a, 10b, 10c, 10d, 10e, respectively
                Update Frequency[$f_1, f_2, f_3, f_4, f_5$]
            end for
        end for
        $V_j \leftarrow$ Frequency[$f_1, f_2, f_3, f_4, f_5$]
        $j++$
    until $j \leq 32$
    Update FoG Matrix with the vectors $V_{1\ to\ 32}$
end for
```

6 Proposed character recognition model

An offline handwritten character recognition method for Odia characters, including the compound characters, is suggested here. An offline handwritten character recognition method for Odia characters, including the compound characters, is suggested here. In literature, few papers are there, based on handwritten compound characters [8, 33] of Bangla language of India, where authors have decomposed the compound characters into their basic

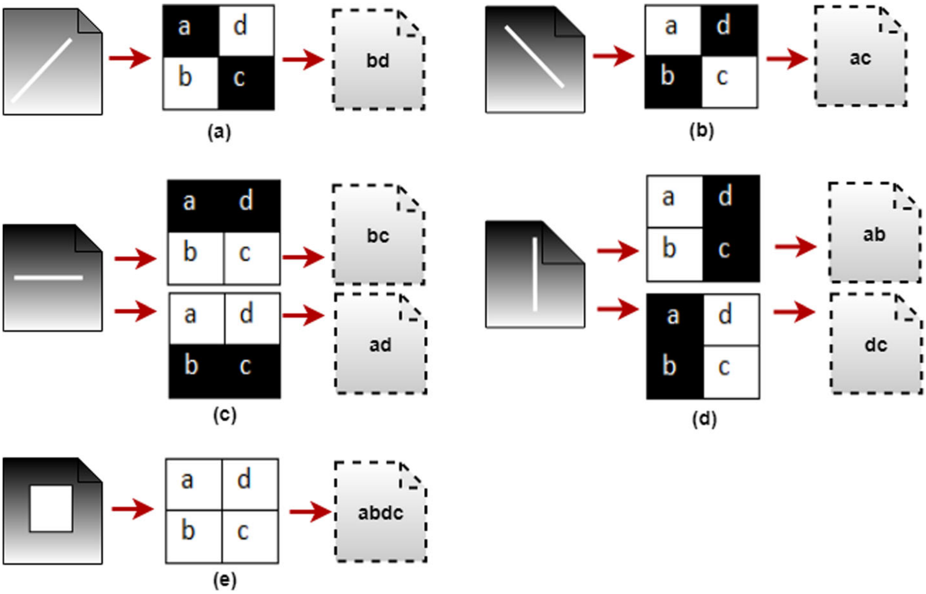


Fig. 10 Representing the angular positions of the connecting pixels in a 2 × 2 cell and extracting the character code from this

forms of characters and 'matras' [30]. Then they have used their recognition methodologies to classify these. This logic may not be fruitful here because if a compound character of Odia would decompose, then it would reform more than one character, as shown in Fig. 11. Due to which the meaning of the particular text will be different. As a result, rather than breaking compound characters to identify them in their simplest form, an attempt has been made to recognize the complete compound characters at once. All the steps involved during the identification process are shown in the block diagram, Fig. 12.

There are three fundamental tasks involved in this recognition process and can be identified from the block diagram. Those are: preprocessing, feature extraction, and classification methods. Sections 4 and 5, previously addressed the preprocessing and feature extraction phases. After this, the most necessary task is to classify the test samples. It can be achieved by giving the test samples appropriate class labels based on their extracted features.

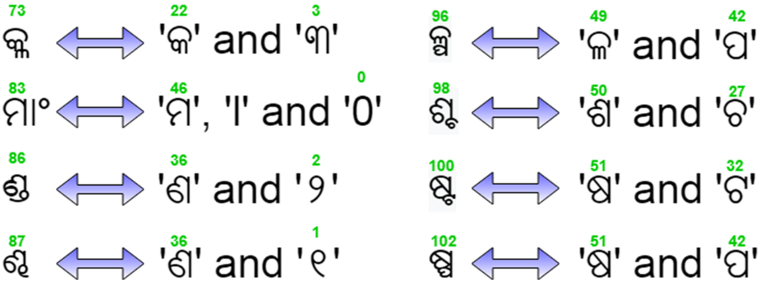


Fig. 11 Decomposition of one compound character forms multiple other characters. Numbers shown in green colour on top of the characters are the respective class labels present in the 'IIITBOdiaV2' database

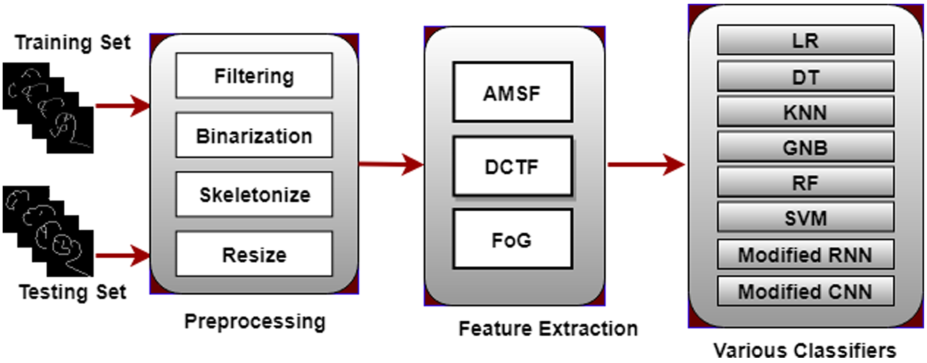


Fig. 12 Block diagram for the proposed handwritten character recognition framework

Different experiments were carried out using six traditional machine learning algorithms and two neural network models. The machine learning classifiers include Logistic Regression (LR) [35], Gaussian Naive Bayes (GNB) [26], Decision Tree (DT) [26], K-Nearest Neighbors (kNN) [31], Random Forest (RF) [22], Support Vector Machine (SVM) [31], Modified RNN, and the CNN [17]. The three feature sets, AMS, CTD, and FoG, are used to compute and evaluate the accuracies obtained by these classifiers. After creating all these three kinds of feature matrices, then the classifiers are applied, and the classification accuracy is computed. Except for these three types of feature matrices, one more feature matrix has also been considered for conducting the experiments. That is the resulting merged feature matrix of all three feature sets.

- AMS = dimension is 256 and 1 more for the class labels.
- CTD = dimension is 8 and 1 more for the class labels.
- FoG= dimension is 160 and 1 more for the class labels.
- AMS + CTD + FoG = dimension is 256+ 8+ 160 = 424 and 1 more for the class labels.

The Python sci-kit learning environment has been used to access the inbuilt classifiers such as LR, DT, kNN, GNB, RF, and SVM. Most of the function parameters are set to default depending on the classifier chosen. Wherever any tuning has been done, it is mentioned here. Using solver='lbfgs', multi class='auto', and max_iter=10000 for Logistic Regression. The solver is solely responsible for making the algorithm more effective. Because the dataset contains a little large number of samples, the number of iterations is increased to achieve successful convergence. The criterion='entropy' and estimators='300' have been set explicitly for Random Forest. The n_estimators determine the number of trees to be built. To produce a better split point, an 'information gain based entropy' criterion was used. To maintain a standard and neutral study, these standards are kept consistent for all the conducted experiments. All the rest parameters for all the classifiers are set to the default value. Instead of a conventional RNN and CNN, a tweaked version of RNN and CNN is used in this research using Keras in python. Instead of providing a set of images, the feature matrix is used as an input in these neural network architectures.

6.1 Modified recurrent neural network (RNN)

A recurrent neural network (RNN) is a form of neural network that operates similar to a traditional neural network. The only difference here is the memory estimation takes a short time. Multiplying the input weight and activation functions, the network model generates results like a regular neural network. The RNN returns to this performance several times itself. The LSTM model was designed using the Keras Deep Learning Library. The time-step was named because the model requires time to generate results. A repeated value of 300 has been taken for a specific time step. After the data is loaded into memory, the LSTM model can be designed, fitted, and checked. One hidden LSTM layer is used in this suggested model. Since the AMS, CTD, and FoG features have been combined so that the resulting feature matrix will have 8+256+160=424 dimensions. Depending on which, the input layers in the experiments have a maximum of 424 neurons. A dropout layer is used to remove the networks' over-fitting during the training. The value of the dropout is taken as 0.05 in this case. The model is made up of three hidden layers. At the end of each fully connected dense layer, the activation function 'ReLU' is used to maximize model efficiency. The network is compiled with the 'Sparse categorical loss entropy' and the 'Adam Variant of Stochastic Gradient Descent' loss function as the potential optimizers. Here, the batch scale and the value of epoch, both are taken as 50. Based on the dataset's output class labels, the network output layer will have anything from 10 to 112 neurons. In the output layer, an activation function such as Softmax is used to increase the model's prediction to the probability class.

6.2 Modified convolutional neural network (CNN)

A convolutional neural network is used in several applications, such as visual recognition, object tracking, facial recognition, etc. CNN is a form of deep learning that is inspired by the neuron signaling model of the animal visual cortex. The weight learning of CNN and bias are very much close to the procedure of feedforward neural networks. We designed a convolutional neural network with Keras environment under Python to achieve cutting-edge efficiency. In this network architecture, the Keras sequential model was used. The add() processes are used to stack layers. The rectified linear unit (ReLU) is one of the best activation functions that have been used in the network, with three densely connected convolution layers. The suggested feature vector is used as an input, whose dimensions are 8, 160, 256, or 424, depending on the selected feature set. The first input layer has nodes based on the input feature dimension and 0.03 dropouts to reduce the overfitting effect. Excluding this input layer, 3 more hidden layers are incorporated. The 'adam' optimizer is used as a 'stochastic gradient descent (SGD)' optimizer. The validation accuracy obtained in the estimation process is dependent on 'sparse categorical cross entropy' loss. Both the epoch and the batch size are given a value of 50 for these experiments. The set of output neurons and the softmax multi-class activation function are presented in the final layer. Based on the data set to be classified, the number of labels will vary from 10 to 112.

7 Experimental analysis

Here, various experiments have been conducted based on the datasets, feature sets as well as classifiers. Four Odia handwritten character datasets were taken for the experiments: ISI Kolkata, IIT Bhubaneswar, NITROHCS_V1.0, and IIITBOdiaV2. Though the IIT Bhubaneswar dataset consists of digits, basic characters, and compound characters, here we could collect only the digits and basic characters. The NITROHCS_V1.0 dataset consists of basic characters only. The IIITBOdiaV1 dataset initially had digits and basic characters only. Later, we combined some commonly used compound characters with the existing ones to create IIITBOdiaV2. Several experiments are conducted on these datasets. Then all the results are represented in tabular form so that a small difference between the outcomes can be identified clearly. The results displayed in this section except for ISI Kolkata is the average of k-fold cross-validation, where the k value is 5. The ISI Kolkata dataset has a separate training and testing set, but the others do not.

In the first trial, all the three feature sets extracted from the different datasets are given to the classifiers, then their obtained accuracy is compared. In the second trial, all the three feature sets were merged to have a larger dimension of the dataset and the same has been given as input to all the classifiers that have been mentioned earlier. All the obtained accuracy from different classifiers based on the feature set is compared. Tables 2, 3 and 4 represent the outcomes of different classifiers on different features for ISI Kolkata, IIT Bhubaneswar, and IIITBOdiaV2 digit dataset, respectively. More or less, all the classifiers are giving quite better results using the merged set of all three features. Still, a clear comparison can be seen that all the classifiers outperform on the FoG feature than AMS, though the FoG is a feature set with only 160 dimensions. This dimension is quite less than the AMS type feature, which is 256 in dimension. Furthermore, when all of the features are combined into one and then fed into the classifiers, the accuracy of all of the classifiers grows. It means that all the feature sets are relevant towards the correct recognition of characters. In the merged dataset, all the feature subsets have a clear contribution towards accurate character identification.

Tables 5, 6 and 7 present the outcomes of different classifiers on different basic character datasets from IIT Bhubaneswar, NITROHCSv1 and IIITBOdiaV2, respectively. Here also the same fact happened to like the digit recognition. Always the neural network-based models perform better than the other state-of-the-art classifiers. Out of the three datasets, the recognition accuracy of IIITBOdiaV2 basic characters is quite low compared to the other basic character datasets. It implies the basic characters present inside the IIITBOdiaV2 dataset are a little complex and must be difficult to recognize by any recognition methodologies.

Tables 8 and 9 represent the results obtained by the classifiers on IIT Bhubaneswar and IIITBOdiaV2 datasets, respectively. Here these datasets consist of digits and basic characters only. Usually, if the number of classes increases, then the accuracy decreases. But here, it does not happen because these datasets consist of characters as well as digits. Indeed,

Table 2 Recognition accuracy of different classifiers on different feature set of ISI Kolkata digit dataset

	Feature set			
Classifier	DCTF	AMSF	FoG	DCTF+AMSF+FoG
GNB	65.32	91.37	93.47	94.12
DT	67.27	92.36	94.11	94.56
kNN	68.38	93.14	94.23	94.62
LR	69.39	93.8	94.45	95.1
SVM	70.54	94.28	95.81	96.37
RF	71.58	95.48	96.16	96.71
RNN	72.14	96.14	97.32	97.87
CNN	72.7	96.62	97.75	98.22

Table 3 Recognition accuracy of different classifiers on different feature set of IIT Bhubaneswar digit dataset

	Feature set			
Classifiers	CTD	AMS	FoG	CTD+AMS+FoG
GNB	63.7	91.28	92.49	93.84
DT	64.29	92.71	94.11	94.76
kNN	65.98	93.12	93.78	94.73
LR	66.29	93.4	94.32	95.88
SVM	67.76	94.22	95.14	96.1
RF	68.32	94.67	95.92	96.56
RNN	70.87	95.12	96.66	97.34
CNN	70.56	95.43	96.87	97.21

Table 4 Recognition accuracy of different classifiers on different feature set of IIITBOdiaV2 digit dataset

	Feature set			
Classifiers	CTD	AMS	FoG	CTD+AMS+FoG
GNB	62.65	88.3	90.41	91.77
DT	63.42	90.76	92.66	93.52
kNN	64.76	92.41	93.7	94.45
LR	65.84	93.49	94.87	95.69
SVM	66.38	94.83	95.42	96.43
RF	67.85	95.32	96.44	97.52
RNN	68.27	96.3	97.2	98.33
CNN	68.4	96.71	97.84	98.54

Table 5 Recognition accuracy of different classifiers on different feature set of IIT Bhubaneswar basic character dataset

	Feature set			
Classifiers	CTD	AMS	FoG	CTD+AMS+FoG
GNB	56.48	76.45	79.63	80.65
DT	58.19	79.23	81.24	82.56
kNN	60.17	79.72	83.94	84.63
LR	59.43	80.63	83.23	84.47
SVM	60.93	81.55	84.75	85.31
RF	62.4	83.24	86.52	87.38
RNN	66.86	85.91	87.12	88.15
CNN	67.56	86.11	87.47	88.23

Table 6 Recognition accuracy of different classifiers on different feature set of NITROHCSv1 basic character dataset

	Feature set			
Classifiers	CTD	AMS	FoG	CTD+AMS+FoG
GNB	61.34	81.44	85.21	86.34
DT	64.83	82.23	85.23	88.66
kNN	63.85	83.87	87.65	89.86
LR	65.93	84.86	88.97	90.35
SVM	66.34	85.62	89.34	91.46
RF	67.48	86.19	90.23	92.56
RNN	69.39	88.12	91.64	93.12
CNN	70.11	89.23	91.56	93.35

Table 7 Recognition accuracy of different classifiers on different feature set of IIITBOdiaV2 basic character dataset

	Feature set			
Classifiers	CTD	AMS	FoG	CTD+AMS+FoG
GNB	50.53	75.54	76.83	77.32
DT	52.35	77.33	78.45	79.44
kNN	51.86	76.35	78.3	79.23
LR	52.87	76.56	79.76	80.97
SVM	53.73	78.23	80.25	81.34
RF	54.63	80.34	81.34	82.63
RNN	55.23	80.75	82.32	83.27
CNN	56.25	81.62	82.65	83.56

Table 8 Recognition accuracy of different classifiers on different feature set of IIT Bhubaneswar (digit + basic character) dataset

	Feature set			
Classifiers	CTD	AMS	FoG	CTD+AMS+FoG
GNB	63.49	82.34	84.7	84.87
DT	66.79	84.89	86.1	88.23
kNN	66.23	85.23	87.82	88.98
LR	66.72	85.73	88.12	89.34
SVM	67.28	86.43	89.53	90.76
RF	68.29	87.52	90.23	91.86
RNN	69.23	90.23	91.32	92.12
CNN	70.34	89.23	91.59	92.45

Table 9 Recognition accuracy of different classifiers on different feature set of IIITBOdiaV2 (digit + basic character) dataset

	Feature set			
Classifiers	CTD	AMS	FoG	CTD+AMS+FoG
GNB	59.76	79.82	82.45	84.2
DT	62.98	81.98	84.34	86.11
kNN	64.62	83.05	85.87	86.26
LR	65.32	83.03	85.34	86.73
SVM	66.52	84.93	85.83	87.23
RF	67.09	85.82	87.29	88.98
RNN	68.12	86.24	88.21	89.76
CNN	68.34	86.84	89.57	90.23

the recognition result of numerals is always better than character recognition. So when the digits are mixed with alphabets, it helps to enhance the recognition accuracy. In both IIT Bhubaneswar and IIITBOdiaV2, the number of classes is 60, and IIITBOdiaV2 has fewer samples. Nonetheless, the recognition accuracy is low. So it can be inferred that the IIITBOdiaV2 dataset has more complex specimens than the IIT Bhubaneswar dataset.

Tables 10 and 11 display the recognition rate in terms of accuracy for characters that include compound characters. Table 10 displays the results where the features are extracted from 102 classes because it consists of basic characters and compound characters. Whereas Table 11 presents the results for all the classifiers applied on features that are extracted from 112 numbers of classes because it consists of digits, basic characters as well as compound characters. It should also be noted that the number of classes in Table 10 is less than that in Table 11, but its recognition accuracy is lower. Because, in Table 11, the digit samples are merged, which helps to enhance its recognition.

7.1 Comparison

Images can be used to train and evaluate a neural networks models. A comparison has been performed on the neural networks based on the input provided, first by providing immediate

Table 10 Recognition accuracy of different classifiers on different feature set. Samples are taken from 102 classes (basic character + compound character) of IIITBOdiaV2 dataset

	Feature set			
Classifiers	CTD	AMS	FoG	CTD+AMS+FoG
GNB	48.42	73.28	74.98	75.72
DT	50.69	75.52	76.23	77.83
kNN	51.09	75.29	76.62	79.1
LR	50.34	76.46	77.32	78.23
SVM	52.69	77.42	78.52	79.7
RF	52.84	78.62	79.76	80.24
RNN	54.72	80.23	81.12	82.74
CNN	55.24	80.73	81.56	82.45

Table 11 Recognition accuracy of different classifiers on different feature set. Samples are taken from 112 classes (digit + basic character + compound character) of IIITBOdiaV2 dataset

	Feature set			
Classifiers	CTD	AMS	FoG	CTD+AMS+FoG
GNB	53.67	74.54	80.05	80.74
DT	55.58	76.43	81.3	81.58
kNN	55.38	77.35	81.49	81.84
LR	55.95	78.39	82.74	82.68
SVM	57.26	79.38	83.29	83.52
RF	58.36	80.37	83.73	84.49
RNN	61.19	82.18	84.25	86.26
CNN	60.25	82.62	84.84	86.56

images and secondly by providing the extracted features. The efficiency of the character recognition approach using both methods can be compared using two different parameters: accuracy and execution time per epoch. The accuracy and execution time per epoch of the CNN and RNN depending on the input are shown in Table 12. Although the traditional CNN and RNN had a marginally higher accuracy when images were used as input, it did so at a substantial cost in terms of execution time. Here, the two main factors which influence the increase in the execution time of traditional CNN are sample count and input sample resolution (in pixels). The number of layers of CNN has been kept constant in both scenarios. The suggested feature extraction method, on the other hand, provides the classifiers with a constant feature dimension regardless of the image size. Thus, the conventional CNN and RNN will always take more time to execute than the proposed approach. This study not only emphasizes accuracy but, at the same time, the execution time is also being considered. As a result, further experiments with basic characters and compound characters based on the direct images as input to the classification system have been ignored here. Table 13 represents a comparison of our proposed approach with the others present in the literature based on the accuracy obtained. It can be seen that our suggested approach, as well as feature extraction technique, is unable to reach the peak as others have performed, but the results are very much closer. This study is the first attempt towards the handwritten Odia compound character recognition. As a result, it is confirmed that the suggested feature extraction techniques are fruitfully working on a different category of Odia characters while also successfully supporting numerous machine learning algorithms.

Table 12 Comparison of recognition accuracy of RNN and CNN classifier on different digit dataset based on the input provided

	Traditional RNN		Modified RNN		Traditional CNN		Modified CNN	
Dataset	Accuracy	Time	Accuracy	Time	Accuracy	Time	Accuracy	Time
ISI Kolkata	98.06	41 Sec.	97.87	2.5 Sec.	98.73	76 Sec.	98.22	4 Sec.
IIT Bhubaneswar	97.45	36 Sec.	97.34	2 Sec.	97.67	70 Sec.	97.21	3.5 Sec.
IIITBOdiaV2	98.66	24 Sec.	98.33	1 Sec.	98.72	45 Sec.	98.54	3 Sec.

Table 13 Comparison of recognition accuracy achieved by different researchers present in literature with the proposed recognition model on Odia handwritten datasets

Literature	dataset	samples	Method	Accuracy
Das et al., 2019 [6]	ISI Kolkata	Digit	ELM	94.47
Sethy et al., 2020 [40]	ISI Kolkata	Digit	CNN	98.4
Das et al., 2020 [7]	ISI Kolkata	Digit	SVM	97.7
Dash et al., 2020 [13]	ISI Kolkata	Digit	MQDF	99.22
Proposed	ISI Kolkata	Digit	CNN	98.22
Dash et al., 2017 [12]	IIT Bhubaneswar	Digit	kNN,MQDF	99.1
Sethy et al., 2020 [40]	IIT Bhubaneswar	Digit	CNN	97.71
Proposed	IIT Bhubaneswar	Digit	RNN	97.34
Dash et al., 2017 [12]	IIT Bhubaneswar	Basic chars	kNN,MQDF	95.14
Dash et al., 2020 [13]	IIT Bhubaneswar	Basic chars	MQDF	93.24
Proposed	IIT Bhubaneswar	Basic chars	CNN	88.23
Sethy et al., 2019 [37]	NITROHCSV1.0	Basic chars	rbf NN	98.8
Das et al., 2020 [7]	NITROHCSV1.0	Basic chars	RF	98.9
Proposed	NITROHCSV1.0	Basic chars	CNN	93.35

8 Conclusion and future work

Odia is one of the most popular languages in India, used by more than 50 million people in the whole world. There is a considerable need for an Odia offline handwritten character recognition mechanism to identify numerals, basic characters, and compound characters. Though the number of users is increasing every day due to the digital involvement of computers and smartphones, still very little research has been done on handwritten character recognition in the literature. Here, in this study, the issue regarding the offline Odia handwritten character recognition has been addressed. One of the major issues was the unavailability of the Odia character dataset publicly that includes compound characters. With this in mind, a dataset IIITBOdiaV2 has been created at IIIT Bhubaneswar. The dataset currently has 16198 numbers of characters in total from 112 character classes. To date, the development process of this dataset is going on, and these numbers are increasing day by day. Here, 52 categories of compound characters were incorporated, which are very popular and widely used by most users. It is a fact that deep learning models are always a leader in the case of pattern recognition. Generally, it takes the images as input directly. The computation takes a long time when the image size and the number of samples are large. We have used deep learning networks, with the input as a set of features (AMS, CTD, and FoG) with a much lower dimension than the size of original character images. These extracted features of the images are not only small in size, but they are also more functional because they help in the character identification across many classes. At the same time, these feature sets also support various machine learning classifiers by giving a consistent recognition result in terms of accuracy under k-fold cross-validation. We are trying to append some other possible compound characters of the Odia script to this existing dataset so that the quality of the character recognizer can be enhanced. It should also be noted that the proposed approach's recognition accuracy is not as high as those found in the literature. In the future, some new feature extraction approaches could be used to yield better recognition results with fewer feature dimensions. Nowadays, most documents are available in a distributed environment, and in such cases, new techniques like edge computing [19] can also be used in the future for the offline handwritten character recognition task.

Declarations

Conflict of Interests The authors declare that there are no conflicts of interest.

References

1. Agrawal P, Chaudhary D, Madaan V, Zabrovskiy A, Prodan R, Kimovski D, Timmerer C (2021) Automated bank cheque verification using image processing and deep learning methods. Multimed Tools Appl 80(4):5319–5350
2. Baruch O (1988) Line thinning by line following. Pattern Recogn Lett 8(4):271–276
3. Bhattacharya U, Chaudhuri B (2005) Databases for research on recognition of handwritten characters of indian scripts. In: Eighth international conference on document analysis and recognition (ICDAR'05). IEEE, pp 789–793
4. Bhowmik TK, Parui SK, Bhattacharya U, Shaw B (2006) An hmm based recognition scheme for handwritten oriya numerals. In: 9th international conference on information technology (ICIT'06). IEEE, pp 105–110
5. Das D, Dash R, Majhi B (2018) Optimization based feature generation for handwritten odia-numeral recognition. In: 2018 fourteenth international conference on information processing (ICINPRO). IEEE, pp 1–5
6. Das D, Nayak DR, Dash R, Majhi B (2019) An empirical evaluation of extreme learning machine: application to handwritten character recognition. Multimed Tools Appl 78(14):19495–19523
7. Das D, Nayak DR, Dash R, Majhi B (2020) Mjcn: Multi-objective jaya convolutional network for handwritten optical character recognition. Multimed Tools Appl 79(43):33023–33042
8. Das N, Acharya K, Sarkar R, Basu S, Kundu M, Nasipuri M (2014) A benchmark image database of isolated bangla handwritten compound characters. Int J Doc Anal Recog (IJDAR) 17(4):413–431
9. Dash KS, Puhan N, Panda G (2014) A hybrid feature and discriminant classifier for high accuracy handwritten odia numeral recognition. In: 2014 IEEE region 10 symposium. IEEE, pp 531–535
10. Dash KS, Puhan NB, Panda G (2015) Handwritten numeral recognition using non-redundant stockwell transform and bio-inspired optimal zoning. IET Image Process 9(10):874–882
11. Dash KS, Puhan NB, Panda G (2015) On extraction of features for handwritten odia numeral recognition in transformed domain. In: 2015 eighth international conference on advances in pattern recognition (ICAPR). IEEE, pp 1–6
12. Dash KS, Puhan NB, Panda G (2017) Odia character recognition: a directional review. Artif Intell Rev 48(4):473–497
13. Dash KS, Puhan NB, Panda G (2020) Sparse concept coded tetrolet transform for unconstrained odia character recognition. arXiv:2004.01551
14. Dey R, Balabantaray RC (2019) A novel sliding window approach for offline handwritten character recognition. In: 2019 international conference on information technology (ICIT). IEEE, pp 178–183
15. Dey R, Balabantaray RC, Mohanty S (2021) Sliding window based off-line handwritten text recognition using edit distance. Multimed Tools Appl. https://doi.org/10.1007/s11042-021-10988-9
16. Dey R, Balabantaray RC, Piri J (2020) A robust handwritten digit recognition system based on sliding window with edit distance. In: 2020 IEEE international conference on electronics, computing and communication technologies (CONECCT). IEEE, pp 1–6
17. Guo Y, Liu Y, Bakker EM, Guo Y, Lew MS (2018) Cnn-rnn: a large-scale hierarchical image classification framework. Multimed Tools Appl 77(8):10251–10271
18. Huh JH (2020) Surgery agreement signature authentication system for mobile health care. Electronics 9(6):890
19. Huh JH, Seo YS (2019) Understanding edge computing: Engineering evolution with artificial intelligence. IEEE Access 7:164229–164245
20. Jindal T, Bhattacharya U (2013) Recognition of offline handwritten numerals using an ensemble of mlps combined by adaboost. In: Proceedings of the 4th international workshop on multilingual OCR, pp 1–5

21. Justusson B (1981) Median filtering: Statistical properties. In: Two-dimensional digital signal prcessing II. Springer, pp 161–196
22. Kaur RP, Kumar M, Jindal MK (2019) Newspaper text recognition of gurumukhi script using random forest classifier. Multimed Tools Appl:1–14
23. Lempitsky V, Kohli P, Rother C, Sharp T (2009) Image segmentation with a bounding box prior. In: 2009 IEEE 12th international conference on computer vision. IEEE, pp 277–284
24. Maalej R, Kherallah M (2020) Improving the dblstm for on-line arabic handwriting recognition. Multimed Tools Appl 79(25):17969–17990
25. Majhi B, Pujari P (2018) On development and performance evaluation of novel odia handwritten digit recognition methods. Arab J Sci Eng 43(8):3887–3901
26. Maswadi K, Ghani NA, Hamid S, Rasheed MB (2021) Human activity classification using decision tree and naïve bayes classifiers. Multimed Tools Appl:1–18
27. Mohapatra RK, Mishra TK, Panda S, Majhi B (2015) Ohcs: A database for handwritten atomic odia character recognition. In: 2015 fifth national conference on computer vision, pattern recognition, image processing and graphics (NCVPRIPG). IEEE, pp 1–4
28. Otsu N (1979) A threshold selection method from gray-level histograms. IEEE Trans Syst Man Cybern 9(1):62–66
29. Pal U, Sharma N, Wakabayashi T, Kimura F (2007) Handwritten numeral recognition of six popular indian scripts. In: Ninth international conference on document analysis and recognition (ICDAR 2007), vol 2. IEEE, pp 749–753
30. Pramanik R, Bag S (2018) Shape decomposition-based handwritten compound character recognition for bangla ocr. J Vis Commun Image Represent 50:123–134
31. Rehman A, Naz S, Razzak MI (2019) Writer identification using machine learning approaches: a comprehensive review. Multimed Tools Appl 78(8):10889–10931
32. Roy K, Pal T, Pal U, Kimura F (2005) Oriya handwritten numeral recognition system. In: Eighth international conference on document analysis and recognition (ICDAR'05). IEEE, pp 770–774
33. Roy S, Das N, Kundu M, Nasipuri M (2017) Handwritten isolated bangla compound character recognition: a new benchmark using a novel deep learning approach. Pattern Recogn Lett 90:15–21
34. Sarangi PK, Ahmed P, Ravulakollu KK (2014) Naïve bayes classifier with lu factorization for recognition of handwritten odia numerals. Indian Jo Sci Technol 7(1):35–38
35. Sebti A, Hassanpour H (2017) Body orientation estimation with the ensemble of logistic regression classifiers. Multimed Tools Appl 76(22):23589–23605
36. Sethy A, Patra PK (2016) Off-line odia handwritten numeral recognition using neural network: a comparative analysis. In: 2016 international conference on computing, communication and automation (ICCCA). IEEE, pp 1099–1103
37. Sethy A, Patra PK (2019) Off-line odia handwritten character recognition: an axis constellation model based research. Int J Innov Technol Exploring Eng (IJITEE) 8(9S2):788–793
38. Sethy A, Patra PK, Nayak DR (2018) Off-line handwritten odia character recognition using dwt and pca. In: Progress in advanced computing and intelligent engineering. Springer, pp 187–195
39. Sethy A, Patra PK, Nayak S, Jena PM (2017) Symmetric axis based off-line odia handwritten character and numeral recognition. In: 2017 3rd international conference on computational intelligence and networks (CINE). IEEE, pp 83–87
40. Sethy A, Patra PK, Nayak SR (2020) Offline handwritten numeral recognition using convolution neural network. Mach Vis Inspection Syst Image Process Concepts Methodologies Appl 1:197–212
41. Singh PK, Sarkar R, Das N, Basu S, Kundu M, Nasipuri M (2018) Benchmark databases of handwritten bangla-roman and devanagari-roman mixed-script document images. Multimed Tools Appl 77(7):8441–8473
42. Tripathy N, Panda M, Pal U (2003) System for oriya handwritten numeral recognition. In: Document recognition and retrieval XI, vol 5296, pp 174–181. International Society for Optics and Photonics

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.