ISSN 2007-9737

An Optimized Workflow for Odia Handwritten Character Recognition

Pragnya Ranjan Dash*, Rakesh Chandra Balabantaray

International Institute of Information Technology,
Dept. of CSE,
India

{c122006@, rakesh}@iiit-bh.ac.in

Abstract. Classifying handwritten Odia scripts is challenging because of the script's complex character shapes and the lack of large annotated datasets. Odia is a low-resource language with only limited digital materials, making the development of effective recognition systems important for improving access and ensuring fair digital representation. The present study addresses the classification of handwritten Odia data, including basic characters, digits, and a set of frequently used compound characters. The proposed method combines several preprocessing steps with a lightweight Convolutional Neural Network (CNN), and data augmentation is applied to enrich the training samples and reduce overfitting. To evaluate the approach, four benchmark datasets were used: NITROHCS V1.0 (basic characters), ISI Kolkata (digits), IIT Bhubaneswar (digits and characters), and IIITBOdiaV2 (digits and characters). The model was trained on one dataset and tested on the others to assess adaptability. Additional evaluation was performed on real handwritten data consisting of both characters and digits. The experimental results demonstrate the effectiveness of the CNN model, showing an accuracy that either surpasses or closely matches earlier proposed systems using the same dataset.

Keywords. Thresholding, gaussian filter, edge detection, segmentation, preprocessing, CNN, recognition.

1 Introduction

A machine can recognize handwritten text by using optical character recognition (OCR). There are two ways to do this: online and offline, as shown in Figure 1. Online involves using electronic devices to track writing direction as it occurs [21], such as in online signature authentication [27]. On the other hand, offline handwriting identification involves recognizing text from scanned images [5]. This not only saves space but also time. In offline OCR, the system doesn't get details about how the pen moves, its path, or the direction of the text line. So offline OCR is considered more crucial than online OCR [16].

Many languages are spoken in India in a variety of scripts. However, there is a surprising scarcity of research in the literature supporting the identification of handwritten text using optical character recognition (OCR). Notably, most studies focus on recognizing Devanagari, Tamil, Telugu, and Bangla handwritten characters, sidelining other regional languages [41]. Over 35 million people speak Odia, an Indo-Aryan language, in states including Gujarat, West Bengal, Andhra Pradesh, and Odisha. Odia has its origins in the Kalinga script, a variation of the Brahmi script used in ancient India. Odisha's literature and history have been preserved over the years on palm leaves.

Authors like Madhusudan Das, Upendra Bhanja, Gopabandhu Das, Radhanath Ray, Gangadhar Meher, and Fakir Mohan Senapati have made significant contributions to Odia literature. Over the past few decades, efforts have been made to safeguard these literary treasures by transforming them into digital data, following advancements in digital preservation. To avoid storage issues, opting for text files over scanned copies makes practical sense. However, since manually converting these vast volumes of content is not feasible, the solution lies in employing offline OCR for Odia characters [36]. This technology enables

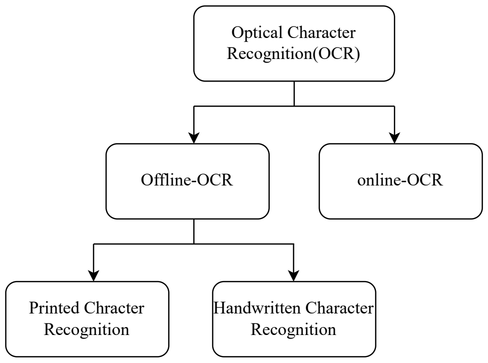

Optical Character Recognition(OCR)
Offline-OCR
online-OCR
Printed Chracter Recognition
Handwritten Character Recognition

Fig. 1. Types of Optical Character Recognition

the automatic conversion of image data into text files, ensuring a more efficient and sustainable approach to preserving these cultural gems.

Odia script consists of 10 numerals and 49 basic characters, from which over 100 compound characters can be constructed. The fundamental character set comprises 12 vowels and 37 consonants. In the Odia script, writing flows from left to right, and the conventional concept of upper and lowercase is absent. The majority of Odia characters exhibit a roundish nature, and the similarities between many characters create challenges for classification tasks [16].Unlike Hindi and Bangla, Odia lacks horizontal lines, making segmentation tasks less difficult. Adding to the complexity, when a vowel follows a consonant, the character undergoes a modified shape.

This modified form can appear on the left, right, both sides, bottom, or top of the character, giving rise to what is known as ”matras.” These matras contribute to the distinctive appearance of characters based on the accompanying vowel. Furthermore, the script introduces an additional layer of complexity with compound characters, known as ”juktakshara.” These compound characters emerge when a consonant or vowel follows another consonant. The merging of these characters results in a compound shape, adding more complication to the script's visual dynamics of the character. Although only limited work has been carried out on offline recognition of Odia characters, most existing studies concentrate on printed text. Reported results for printed character recognition are generally satisfactory. In contrast, research on handwritten Odia characters is relatively scarce [12]. The majority of available work addresses handwritten numerals, while relatively few studies investigate handwritten characters [37]. Recognition performance is particularly poor in the case of handwritten compound characters. Several factors contribute to this difficulty, such as the complexity of allographs, the scarcity of suitable datasets, and minimal engagement from commercial initiatives.

Odia, as a low-resource language, encounters major obstacles in the development of Optical Character Recognition (OCR) system. The limited availability of large annotated datasets, combined with minimal investment in dedicated OCR systems, restricts the creation of accurate recognition models. Progress is further slowed by the absence of strong commercial and academic initiatives focusing specifically on Odia. Such challenges are not exclusive to Odia; many other low-resource languages face similar issues arising from insufficient data and limited research activity, which in turn affects advances in natural language processing (NLP) and machine learning. At the same time, the increasing use of smartphones in India and the rising demand for digital content in regional languages highlight the urgency of building OCR systems for Odia. Automatic recognition of handwritten characters has the potential to reduce the digital gap and improve language accessibility, thereby supporting greater linguistic equity.

To address these challenges, a specialized OCR model using a Convolutional Neural Network (CNN) was developed for recognizing handwritten Odia characters. The model was trained and evaluated on several benchmark datasets, including those from NIT Rourkela, IIT Bhubaneswar, ISI Kolkata, and IIIT Bhubaneswar, which contain both numeral and character data. The IIT Bhubaneswar and IIIT Bhubaneswar datasets also include frequently used compound characters, along with a subset of simple characters carrying matras. This broader coverage enables the system to recognize not only basic characters but also common compound forms. For robustness assessment, training was performed on one benchmark dataset, while testing was carried out on others to examine adaptability. Further evaluation was conducted using real handwritten samples collected from multiple individuals, demonstrating the model's potential for real-world application.

The proposed approach can be outlined as follows:

– A set of optimized preprocessing techniques was applied to reduce noise in the dataset samples and to standardize them into a common format across different datasets.
– A custom lightweight Convolutional Neural Network (CNN) module was developed specifically for character recognition.
– The use of CNN eliminates the need for manual feature extraction, thereby improving efficiency and enabling a more automated recognition process.
– To demonstrate the model's robustness, it was tested on a different dataset than the one used for training and validation.
– The proposed approach is data-efficient and adaptable, making it particularly well-suited for applications in low-resource languages, such as Odia.

2 Related Work

In the field of offline Odia character recognition, approximately 45 papers have been published in conferences and journals over the last two decades. This amount of research is relatively small compared to other Indian languages, such as Gurumukhi, Hindi, Telugu, Bangla, and Tamil. Interestingly, many authors in these papers have had to create their synthetic datasets because there's a lack of publicly available benchmark datasets. Unfortunately, only four datasets have been shared with the research community, and they only cover digits, basic characters and some compound characters. Our literature review paid special attention to papers that evaluate their methods using benchmark datasets. This is important because using such standardized datasets is considered the best way to test how well a recognition system works.

In the early stages of exploring Odia handwritten character recognition in the literature, a focus was on the ISI Kolkata digit dataset. Tripathy et al. in 2003 [42] proposed a method for numeral recognition, employing threshold-based binarization as a key preprocessing step. The authors derived features from various aspects, including reservoir area, location, water flow path, loop count, center of gravity, the ratio between reservoir and loop, profile-based features, and jump discontinuity. Conclusively, employing a binary tree classification approach yielded an accuracy of approximately 97.74%. Roy et al. [34] proposed a method that first identified the region of interest (ROI) using bounding box parameters and then segmented characters into blocks. Features were extracted through a chain code histogram with a dimensionality of 400. For classification, neural networks and quadratic classifiers were employed, and the reported accuracy reached 94.81%. In another study, Bhowmik et al. [3] used binarization, thresholding, and normalization during preprocessing of digit images. The extracted features, based on horizontal and vertical strokes, were modeled using a Hidden Markov Model (HMM), which achieved an accuracy of 90.50%.

Jindal T. [23] applied only two preprocessing steps: Gaussian filtering and image resizing. After preprocessing, Zernike moment features were extracted and classified using an ensemble of Multilayer Perceptrons (MLPs) combined with Multiclass AdaBoost.Perceptrons (MLPs) based on Multiclass AdaBoost. Ultimately, this approach yielded a recognition rate of 97.10%. In the study described in [35], the authors Sarangi PK. et al. combined a numeral dataset they created with the established ISI Kolkata dataset. Subsequently, features were extracted through LU factorization, and a Naïve Bayes classifier was employed for classification, yielding an accuracy of 92.75%. In study [28], Majhi et al. applied preprocessing methodologies, including the utilization of a median filter and Canny edge detection. Subsequently, all samples were resized to a dimension of 64×64. Features were extracted through diverse transformations—namely, discrete Fourier, short-time Fourier, discrete cosine, discrete wavelet, S-transform, and curvelet of digits. Employing PCA, the dimensionality of features was condensed to 32. Various classifiers, such as MLP, artificial neural networks, radial basis function networks, and probabilistic neural networks, were enlisted. Ultimately, the study reported an impressive recognition rate of 98.70%.

Dash KS et al.[13], proposed an approach where the sole preprocessing step applied to both the ISI Kolkata and IIT Bhubaneswar datasets was segmentation. The features, founded on sparse concept-coded tetrolets, were subjected to various classifiers, including Random Forest (RF), Support Vector Machine (SVM), k-Nearest Neighbors (kNN), and a modified quadratic discriminant function (MQDF). Consequently, remarkable accuracies were achieved, with IIT Bhubaneswar digits and characters attaining 98.72% and 93.24%, respectively. Similarly, the ISI Kolkata dataset demonstrated high accuracy, particularly in digit classification, with an accuracy of 99.22%. Likewise, in this study by Sethy et al.[40], exclusive preprocessing was employed for the identical datasets, involving size normalization. Following this, the images were fed into a Convolutional Neural Network (CNN) model. The outcomes demonstrated notable accuracy rates, reaching up to 98.4% on the ISI Kolkata dataset and 97.71% on the IIT Bhubaneswar digit dataset.

In the investigation documented in [10] the Sparse Concept Coding-based Image Representation (SCCST) efficiently extracts low-dimensional features using an octave sampling-based non-redundant S-transform, with sparsity reducing feature dimensions and preserving geometric structure. The method uses k-NN and SVM classifiers, achieving 95.48% (SVM) and 96.35% (k-NN) accuracy for IIT BBS handwritten characters, 97.80% (SVM) and 99.2% (k-NN) for numerals. Das et al. [7] employed preprocessing techniques such as binarization, pruning, and dilation on the IIT Bhubaneswar digit dataset, normalizing samples to 40×40 pixels. Features were extracted through convolutional layers, optimized using the JAYA algorithm, and classified with a Random Forest model, yielding an accuracy of 98.25%. Similarly, in [11], regions of interest were localized using bounding boxes and binarized with Otsu's thresholding. The digit samples were partitioned into nine zones, from which Stockwell transforms and Slantlet coefficients were extracted. Optimization was performed with GA, PSO, and Differential Evolution, and classification with kNN achieved 99.1% accuracy on the ISI Kolkata dataset and 98.6% on the IIT Bhubaneswar dataset.

Solely relying on size normalization in [9], the OHCSv1.0 (NIT Rourkela) and ISI Kolkata datasets underwent preprocessing. Subsequently, convolution layers with a multi-objective Jaya-based optimized network were employed to extract features. These features were then utilized in conjunction with Support Vector Machine (SVM) and Random Forest classifiers. Remarkably, this approach achieved an accuracy of 98.9% in character recognition for the NIT Rourkela dataset using Random Forest and 97.70% on the ISI Kolkata numeral dataset using the SVM classifier. Detailed in [39], the preprocessing steps included size normalization, median filtering, and skeletonization applied to both the ISI Kolkata and OHCSv1.0 (NIT Rourkela) datasets. A subset of the data was used, with only 200 samples from each character class and 300 samples from each digit class. Features were extracted based on row symmetry and column symmetry chords. These features were subsequently applied to a decision Tree classifier, resulting in a commendable recognition rate of 96.2% on ISI Kolkata numerals and 95.6% on the OHCSv1.0 Character dataset.

In the 2018 study documented in [38], the authors Sethy A. et al employed normalization and dilation as preprocessing techniques for the NIT Rourkela dataset. A selective approach was taken, with only 150 specimens taken from each of the 47 classes. The feature extraction process involved the discrete wavelet transform, and the dimensionality of these features was subsequently reduced using Principal Component Analysis (PCA). The resulting feature set was then input into a Backpropagation Neural Network (BPNN), achieving an accuracy of 94.8%. In the year 2019, as described in [37], the researchers implemented noise reduction, skew correction, and normalization as preprocessing steps for the NIT Rourkela dataset. A comprehensive set of 350 samples was selected from each of the 47 classes.

Feature extraction was carried out based on symmetric axis chords, incorporating mathematical features such as Euclidean distance and Hamilton distance. The dimensionality of these features was subsequently reduced using Principal Component Analysis (PCA). This refined feature set was then applied to a Gaussian kernel with a radial basis function neural network, resulting in an impressive recognition rate of 98.8%.

In 2022, Raghunath Dey et al.[16] conducted experiments utilizing six traditional machine learning algorithms and two neural network models. The machine learning classifiers encompass Logistic Regression (LR), Gaussian Naive Bayes (GNB), Decision Tree (DT), K-Nearest Neighbors (kNN), Random Forest (RF), Support Vector Machine (SVM), Modified RNN, and CNN. Three distinct feature sets, namely AMS, CTD, and FoG, were employed to compute and assess the accuracies achieved by these classifiers. Notably, the results indicated an accuracy of 98.22% on the ISI Kolkata dataset, 97.34% and 88.23% on the IIT Bhubaneswar numeral and character datasets, and an accuracy of 93.35% on the NIT Rourkela dataset. The study also highlighted a newly created dataset IIITBOdiaV2 at IIIT Bhubaneswar.

The analysis of related work indicates that most existing methods rely on manual feature extraction, followed by classification on specific datasets, which leads to good performance on certain benchmark datasets. However, their performance has not been thoroughly tested on other benchmark datasets in the Odia language, meaning these approaches may struggle to generalize. Another significant gap is the lack of focus on compound handwritten character recognition, which is a challenging and tedious task. Therefore, there is a need for a recognition model that eliminates the manual feature extraction process, provides consistent and accurate results across a broader range of benchmark datasets, and can effectively handle compound character recognition.

3 Dataset Description

Experiments were conducted using four established benchmark datasets that are currently available for the Odia language. The datasets are ISI Kolkata Digit, NIT Rourkela's NITROHCSV1.0, IIT Bhubaneswar, and IIIT Bhubaneswar IIITBOdiaV2. Real data from students of IIIT Bhubaneswar was collected for testing purposes. Detailed information about each dataset is provided in Table 1.

3.1 NITROHCSV1.0

The dataset referred to as ”NITROHCSv1.0” originates from the National Institute of Technology Rourkela (NIT Rourkela) and serves as a valuable resource for handwritten Odia character recognition research [29]. Each of the forty-seven folders within the dataset represents a distinct character from the Odia alphabet, encompassing the entire Odia script. Comprising a total of forty-seven folders. Within each of these folders, a collection of 320 handwritten samples representing a specific character class of the Odia language is located. These images possess dimensions of 81x81 pixels. Notably, this dataset exhibits a remarkable diversity in its composition, originating from 160 different individuals across various age groups, with each contributor submitting samples on two separate occasions.

3.2 IIT Bhubaneswar

The IITBBS Odia Numeral and Character database has been developed to support research on Odia script recognition [12]. The numeral subset consists of 10 classes with about 500–600 samples each, while the character subset includes 105 classes covering both basic and compound Odia characters, with 120–155 samples per class. A key challenge is the presence of border lines in many character images, which adds complexity to recognition and necessitates preprocessing for border removal.

3.3 ISI Kolkata Digit

The ISI Kolkata dataset [2] was created for Odia digit recognition. It contains 10 folders, with each folder corresponding to a single digit. On average, every folder includes about 500 to 600 handwritten samples of the respective numeral.

3.4 IIITBOdiaV2

The IIIT Bhubaneswar IIITBOdiaV2 dataset [16] comprises numerals, basic characters, and frequently used compound characters, developed by distributing grid-based A4 sheets to 150 volunteers aged 5–70, which were scanned at 300 dpi and preprocessed through filtering and thresholding. The dataset contains 112 classes, each stored in a separate folder; for the present work, only numerals and basic characters (the IIIT OdiaV1 subset [16]) were considered. To improve robustness, preprocessing steps such as blurring, thresholding, binarization, and skeletonization were applied, along with data augmentation involving rotations [22] and brightness/contrast adjustments [8] to capture variations in writing style and imaging conditions. In addition, a real-life extension, the IIITBOdia dataset, was collected from 30 students, where handwritten Odia digits and basic characters were extracted from A4 sheets and organized into class-specific folders. These samples underwent Gaussian blurring, binarization, morphological filtering, and contour-based edge detection to reduce noise and enhance stroke clarity. Representative examples of Odia numerals, simple characters, and compound characters are shown in Figure 2.

4 Preprocessing

Preprocessing plays a vital role in handwritten character recognition, as it directly influences both accuracy and efficiency. In building such a system, it is important to carefully handle the challenges that arise from variations in handwriting and image quality. Many challenges can be observed in handwritten character samples, including noise, deterioration of paper quality after use, poor handwriting, and more. Preprocessing is required in multiple steps to eliminate these impediments. The preprocessing stage in this method includes several operations such as image resizing, edge detection, dilation, thresholding, binarization, thinning, and Gaussian blurring. The main goal of these steps is to convert the raw text image into a cleaner and standardized format, making it easier for the recognition model to process the data accurately and efficiently. The steps of character recognition are shown in Fig.(3).

4.1 Gaussian Blur Filter

Gaussian filters are used in noise reduction due to their distinctive feature of assigning varying weights to pixels based on their proximity to the center of the filter. By following the Gaussian function, these filters prioritize nearby pixels, gradually diminishing the impact of those farther away. Gaussian blurring produces a smooth effect while retaining important image features, which makes it a widely used technique in image processing. It works by convolving the image with a two-dimensional Gaussian function, where each pixel value is replaced by a weighted average of its neighbors. Standard deviation controls the extent of smoothing for noise reduction. The two-dimensional Gaussian function, which is used in image processing, is the product of two one-dimensional Gaussians along the x and y axes as given in equation (1):

$$(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{1}$$

x, y**:** Represent the spatial coordinates in the two-dimensional Gaussian function.

σ**:** Standard deviation of the Gaussian distribution.

e**:** Base of the natural logarithm.

The first step involved converting the image into grayscale format. Subsequently, the image undergoes a Gaussian blurring process using OpenCV's Gaussian blur function, employing a 3x3 Gaussian kernel.

The size of the kernel plays a crucial role in determining the extent of the blurring effect on the image. The standard deviation of the Gaussian kernel is denoted by the '0' parameter, and its calculation is automatically determined by the specified kernel size. The formula is shown in equation (2):

$$D(m, n) = \text{GaussianBlur}\{S(p, q), \sigma\}, (p, q) \in K_{pq}, \tag{2}$$

Table 1. Details of the publicly available benchmark Odia handwritten datasets

Datasets	Contents	Labels	Total samples	Dimension	No. of writers
ISI Kolkata	Numericals	10	5970	variable	356
IIT Bhubaneswar	Numericals	10	5000	variable	500
IIITBOdiaV2	Numericals	10	2962	64×64	150
NITROHCS_V1.0	Characters	47	15040	81×81	150
IIT Bhubaneswar	Characters	105	47000	variable	500
IIITBOdiaV2	Characters	102	13236	64×64	150

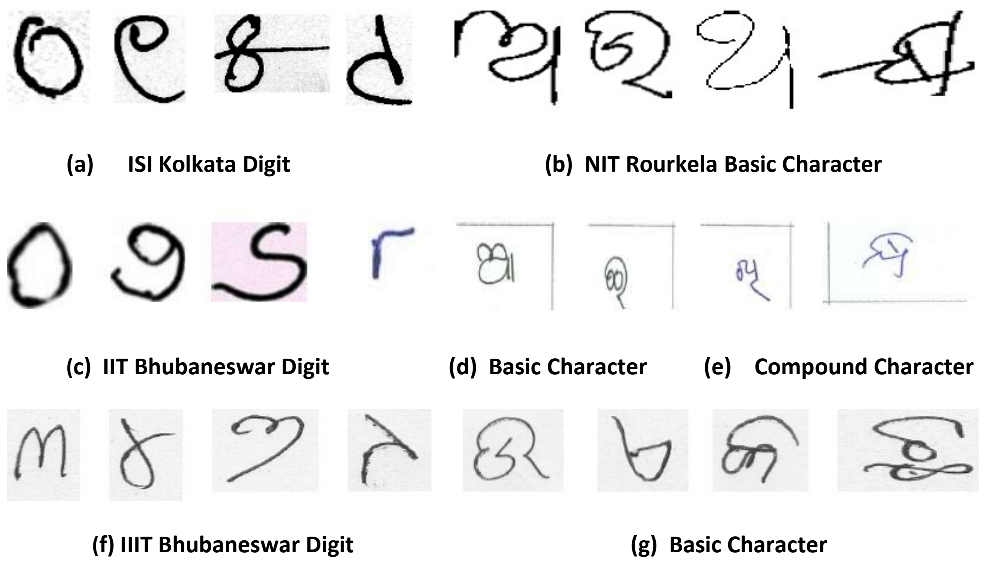

(a) ISI Kolkata Digit
(b) NIT Rourkela Basic Character
(c) IIT Bhubaneswar Digit
(d) Basic Character
(e) Compound Character
(f) IIIT Bhubaneswar Digit
(g) Basic Character

Fig. 2. Selected samples from benchmark datasets

K_{pq} : The set of coordinates in a rectangular sub-image window with (m, n) as the center.

$D(m, n)$: The restored destination image.

$S(p, q)$: The source image, representing the calculated area under the dimension of K_{pq}.

σ : The standard deviation used in the Gaussian blur.

4.2 Thresholding, Binarization and Dilation

To improve accuracy in character recognition tasks, preprocessing of grayscale character images, where pixel intensities range from 0 to 255, is essential. In these images, '255' signifies brightness, while '0' indicates darkness, with intermediate values representing various shades of gray. Employing a high-dimensional storage technique is necessary for such images. On the contrary, text images, characterized by only two values—text lines and background—require a simpler binary representation. The Otsu thresholding [30] is applied in this approach to binarize character images [20].

In the Otsu thresholding method, the process begins by calculating the histogram of pixel intensities in a grayscale image. This histogram is then normalized to create a probability distribution function representing the likelihood of encountering a pixel with a specific intensity. The cumulative distribution function is derived by summing up the probabilities from the probability distribution function.

The mean intensity and global mean are calculated as weighted averages, where the weights are given by the probability distribution function. Subsequently, the between-class variance is computed, measuring the separation between foreground and background pixel intensities using the cumulative distribution and means. The optimal threshold is determined as the intensity value maximizing the between-class variance.

Finally, the image is binarized based on this optimal threshold, with pixel values assigned 1 for intensities greater than or equal to the threshold and 0 for intensities below the threshold [45]. The mathematical details are shown in equations 3 and 4:

$$T_{\text{optimal}} = \arg\max_{T} \left(\sum_{i=0}^{L-1} \left[\left(\sum_{k=0}^{i} \frac{\text{Histogram}(k)}{\text{Total Pixels}} \right) \cdot \left(\sum_{j=0}^{L-1} j \cdot \frac{\text{Histogram}(j)}{\text{Total Pixels}} \right) - \sum_{j=0}^{i} j \cdot \frac{\text{Histogram}(j)}{\text{Total Pixels}} \right]^2 \right), \quad (3)$$

$$\texttt{value1, otsu} = \texttt{threshold(blur, 0, 255, THRESH\_BINARY | THRESH\_OTSU)}. \quad (4)$$

After applying Otsu's thresholding, variations in intensity lead to certain portions of characters being erroneously assigned the background value, resulting in disconnected components. To address this issue, a dilation operation is applied to bridge the gaps and join the broken parts of characters. Dilation in image processing expands the boundaries of foreground objects.

It involves sliding a kernel over the image and setting the pixel value to 1 if at least one pixel in the kernel is 1. Here (3,3) size kernel has been taken for the dilation operation. Let A be the binary image, B be the kernel, and then the dilation operation $\oplus$ is shown in equation (5):

$$(A \oplus B) = \{s \mid (\hat{B})_s \cap A \neq \emptyset\}. \quad (5)$$

4.3 Edge Detection, Bounding Box

The proposed method applied the Canny edge detection method to extract edges from each text image. The technique involves computing the image gradients in both horizontal and vertical directions using Sobel operators(G_x,G_y), which highlight intensity changes [44]. Non-maximum suppression is employed to refine edges by retaining local maxima in the gradient direction, emphasizing the most prominent edges.

Double thresholding is then applied to categorize edges into strong, weak, or non-edges based on their gradient magnitude. Pixels surpassing a high threshold are marked as strong edges, those below a low threshold as non-edges, and those in between as weak edges. The final step, edge tracking by hysteresis, focuses on connecting weak edges to strong edges. A weak edge pixel connected to a strong edge pixel is considered part of the edge, while unconnected weak edges are discarded. Here, 40 and 150 are used as the lower and upper thresholds for the gradient value.

After edge detection, the contours are extracted from a preprocessed image using OpenCV. Then, the 'RETR_EXTERNAL' function of OpenCV is used to ensure that only the external contours are retrieved, focusing on the contours that outline distinct shapes. Additionally, 'CHAIN_APPROX_SIMPLE' function is used to simplify the representation of the contours by compressing consecutive segments into their endpoints, reducing memory overhead. The resulting variable stores the detected contours, providing a basis for subsequent image analysis or visualization tasks.

In the IIT Bhubaneswar dataset, most of the text images contain additional lines at the top and right, or at the bottom and left sides. These extraneous contours are also extracted during the preprocessing phase. To isolate the main text region from surrounding artifacts, the `max(contourArea)` function in OpenCV is applied.

This function selects the contour with the largest area among all detected contours, which generally corresponds to the primary text region since text contours are typically larger than unwanted border lines. From this contour, the bounding

ISSN 2007-9737

Algorithm 1 Image Preprocessing Algorithm

```
1: Set the directory path where the images are stored
2: Get a list of classes (subdirectories) in the specified directory
3: for each class do
4:     Get the list of images in the class directory
5:     for each image in the class do
6:         Convert to grayscale: Gray = 0.299R + 0.587G + 0.114B
7:         Apply Gaussian blur with a $3 \times 3$ kernel
8:         Perform Otsu's thresholding to obtain a binary image
9:         Invert the binary image
10:        Apply dilation with a $3 \times 3$ structuring element
11:        Detect edges using the Canny edge detector
12:        Apply dilation to enhance edge connectivity
13:        Find contours in the dilated edges
14:        for each contour do
15:            Extract bounding box coordinates
16:            Draw bounding box on the binary image
17:            Extract ROI (region of interest)
18:            Thin ROI edges using a thinning algorithm
19:            Further process thinned edges if necessary
20:        end for
21:    end for
22: end for
```

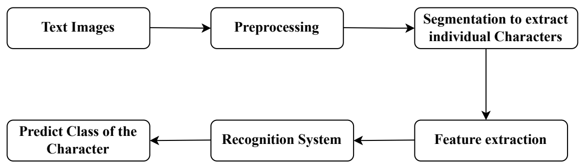

Text Images
Preprocessing
Segmentation to extract individual Characters
Feature extraction
Recognition System
Predict Class of the Character

Fig. 3. Steps of character recognition

box coordinates (x, y, w, h) are obtained using the `boundingRect(max(contourArea))` function. A white bounding box is then drawn around the detected region, and the region of interest (ROI) is extracted based on these coordinates. This ensures that subsequent analysis is performed only on the relevant text area.

4.4 Thinning

The thinning operation is an important step in the image processing pipeline, particularly due to the variation in line widths that naturally occurs in handwritten data [1]. For consistent analysis, it is necessary to obtain a uniform representation of the characters. To achieve this, the `ximgproc.thinning` function from OpenCV's `ximgproc` module is applied. This function skeletonizes binary images by reducing strokes to one-pixel-wide representations while preserving their connectivity and overall structure. The algorithm works by iteratively removing boundary pixels, resulting in a simplified skeleton that retains the essential shape of the original character.

4.5 Resizing

In the initial stage of the study, the samples collected from different datasets varied in size. To ensure consistency and facilitate uniform processing across all repositories, the images were resized to a standard dimension as a key preprocessing step.This not only contributed to a reduction in both time and space requirements but also facilitated the training of our proposed model, which was designed to operate with fixed-size images. All image samples have been resized to 64×64 pixels. The selection of a 64×64 pixel size was determined to be the optimal compromise, effectively balancing computational efficiency with image clarity. This choice ensures that the images are of a manageable size for computational processes while still maintaining sufficient clarity for meaningful analysis. Algorithm 1 demonstrates all the necessary preprocessing steps.

5 Proposed Method

The proposed method uses a lightweight Convolutional Neural Network (CNN) for text image classification [19]. The model is designed to handle 64×64 binary images. It follows a sequence of convolutional layers, each paired with a max-pooling operation to reduce spatial dimensions. The output of the final pooling layer is flattened and passed through fully connected (dense) layers before reaching the output layer. The number of neurons in the output layer corresponds to the number of classes, and a softmax activation function is applied to produce class probabilities.

The first convolutional layer contains 32 filters of size 3×3, followed by ReLU activation and a 2×2 max-pooling layer [31, 15]. This stage extracts basic features while reducing spatial complexity. The second convolutional layer increases the filter count to 64 (size 3×3), again followed by ReLU activation and max-pooling. This layer captures more detailed structures beyond the basic features. A third convolutional layer with 128 filters of size

3×3 is then applied, along with ReLU and a final max-pooling step. This deeper stage allows the network to learn more complex patterns in the data.

The output from the last pooling layer is flattened into a one-dimensional vector and passed to three fully connected layers. The first dense layer has 64 neurons, followed by a second with 128 neurons, both of which refine the representations learned by the convolutional blocks. The final dense layer contains a number of neurons equal to the dataset classes and applies softmax activation to generate probability distributions [24]. To reduce overfitting, dropout with a rate of 0.2 is applied before the last layer [6]. This regularization step improves the robustness of the model, ensuring more reliable classification of handwritten Odia characters.

The mathematical details of the activation function used are shown in equations (6) and (7):

$$\mathrm{ReLU}(z) = \max(0, z), \tag{6}$$

$$\mathrm{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}. \tag{7}$$

In this work, categorical cross-entropy is adopted as the loss function because it is well-suited for multi-class classification problems [26, 14]. It measures the difference between the predicted probability distribution and the true class labels, and works naturally with the softmax activation function in the output layer [18]. By penalizing incorrect predictions, it guides the network to assign higher probabilities to the correct classes and improves overall accuracy. For optimization, the Adam algorithm is employed [4]. Adam combines the advantages of Momentum and RMSprop by maintaining adaptive learning rates for each parameter [25]. Specifically, it tracks two moving averages: the first moment (the mean of gradients) and the second moment (the uncentered variance of gradients). These values are updated during training and used to dynamically adjust parameter updates, resulting in faster convergence and more stable learning [43, 32]. The algorithm also incorporates bias correction to enhance the accuracy of estimates, particularly in the early stages of training. The update rule for a parameter θ in Adam involves adjusting it based on the learning rate, the first moment estimate, and the square root of the biased second moment estimate as given in equation (8):

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t. \tag{8}$$

Prior to model training, one-hot encoding was applied to the class labels within the dataset [33]. This technique converts each class label into a binary vector, where only one element is set to 1 to indicate the presence of a particular class, and all other elements are 0, as shown in Equation (9). This transformation provides a clear and consistent label representation, making it easier for the model to interpret the data. Furthermore, one-hot encoding enhances the efficiency of model training by aligning the label format with the softmax output, thereby facilitating accurate learning in multi-class classification tasks:

$$\text{One-Hot}(i, n) = \delta_{ij} = \begin{cases} 1, & \text{if } i = j, \\ 0, & \text{if } i \neq j. \end{cases} \tag{9}$$

$\text{One-Hot}(i, n)$: One-hot encoding vector for category i in a set of n categories.

δ_{ij}: Kronecker delta, equal to 1 if $i = j$ and 0 if $i \neq j$.

The proposed methodology involves a multi-faceted approach, incorporating preprocessing and data augmentation, followed by the training of the model using the diversified augmented dataset. The applied augmentations include rotation, as well as adjustments to contrast and brightness, as detailed earlier. This comprehensive strategy aims to expose the model to a diverse set of examples, enhancing its ability to recognize different characters present in various benchmark datasets mentioned in the ”Dataset Description” section and enabling it to recognize characters in diverse natural environments and various writing styles. The architectural details of the proposed model are shown in Fig. 4, which provides a comprehensive overview of the different layers used in our proposed model. The description of different parameters used in the model is shown in Table 2. The detailed specifics of character

Table 2. Parameters of the Proposed Model

Parameter	Name/Value	Description
Activation Function	ReLU	Simple, non-linear, and efficient thresholding. Accelerates training, mitigates vanishing gradients, and improves feature extraction.
Categorical Cross-entropy	Measures divergence between predicted and true probability distributions in multi-class classification, guiding the model toward accurate predictions.	Loss Function
Optimization Algorithm	Adam	Combines Momentum and RMSprop for adaptive learning rates, ensuring faster and stable convergence across diverse neural networks.
Classifier	Softmax	Converts logits into probability distribution, enabling efficient multi-class classification, decision-making, and interpretability.
Learning Rate	0.001	Selected heuristically.
Dropout Rate	0.2	Selected heuristically.
Total Epochs	20	Selected heuristically.

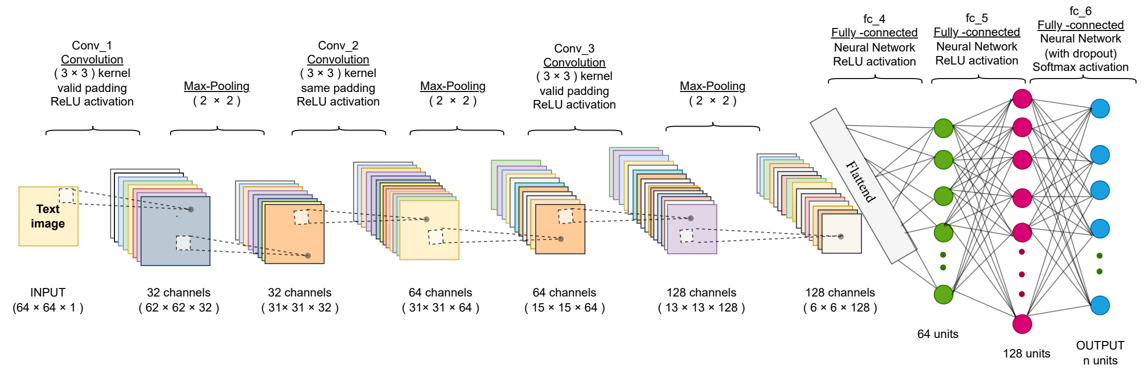


Fig. 4. Proposed Model with Convolution, Max-pool, and Fully connected layer

classification are illustrated in algorithm 2. This algorithm offers a concise overview of the model training on the benchmark dataset and outlines the evaluation of its performance.

6 Result Analysis

In this experiment, a series of experimental investigations was conducted to recognize Odia handwritten characters. Our analysis incorporated four distinct datasets taken from: ISI Kolkata, IIT Bhubaneswar, NITROHCSV1.0, and IIITBOdiaV2, encompassing digits, basic characters, and compound characters typical of the Odia script. To ensure the integrity of our training and validation data, each dataset is split into separate training and validation subsets, adhering to an 80-20 split ratio using a stratified data splitting algorithm to maintain class balance across both training and validation sets, thereby maintaining proportional distribution of samples from each class [17]. Performance metrics such as precision, recall, and F1-score were evaluated for each of dataset as per the equation 10,11,12,13.

This helped to understand how well the model works for different aspects. To ensure the proposed model's strength on different, unseen data, it has been tested with various benchmark datasets and real-world data. The analysis showed high accuracy, confirming the model's reliability

Table 3. Result analysis of proposed model on benchmark datasets

		Training(%)	Validation(%)			
Dataset	Samples	Accuracy	Accuracy	Precision	Recall	F1-Score
ISI Kolkata	Digit	99.44	99.28	99.29	99.27	99.28
IIT Bhubaneswar	Digit	99.27	99.18	99.22	99.17	99.19
IIIT Bhubaneswar	Digit	98.82	98.57	98.54	98.60	98.56
NITROHCSv1.0	Basic Chars	99.18	99.12	99.14	99.12	99.12
IIT Bhubaneswar	Characters	94.10	92.87	94.11	94.05	94.07
IIIT Bhubaneswar	Basic Chars	96.89	95.84	95.77	95.87	95.82

Table 4. Testing accuracy of the model on benchmark and real-life numeral data

	Testing Accuracy(%) on Numeral Dataset			
Training Dataset	ISI Kolkata	IIT Bhub.	IIIT Bhub.	IIITBOdia real
ISI Kolkata Numeral	99.28	92.46	91.4	91.43
IIT Bhubaneswar Numeral	92.14	99.18	91.62	91.57
IIIT Bhubaneswar Numeral	91.02	90.92	98.57	91.05

Table 5. Testing accuracy of the model on benchmark and real-life character data

	Testing Accuracy(%) on Character Dataset			
Training Dataset	NIT Rourkela	IIT Bhub.	IIIT Bhub.	IIITBOdia real
NIT Rourkela Character	99.12	89.40	91.05	91.42
IIT Bhubaneswar Character	87.75	92.87	88.42	88.54
IIIT Bhubaneswar Character	90.80	89.70	95.84	91.65

and adaptability in different situations. The training accuracy, validation accuracy, precision, recall, and F1-score of the proposed model on different benchmark datasets are shown in Table 3.

A series of graphical representations depicting the progression of recognition accuracy and loss values across multiple epochs has been presented in Figures 5 and 6, showcasing the evolution of accuracy and loss metrics over time, providing insight into the performance trends of our model.

To assess the model's robustness post-training and validation, performance tests have been conducted using random samples from diverse benchmark datasets and real-life data, as previously mentioned. While datasets such as ISI Kolkata, IIIT BBSR, and IIT BBSR shared digit datasets with identical class counts, allowing for direct testing without adjustments to the output layer, discrepancies arose in character datasets such as NITROOHCS V1.0 and IITBOdia V1.

These Datasets primarily consisted of basic characters with matching class counts, unlike the IIT BBSR dataset, which included basic characters and frequently used compound characters,

Algorithm 2 Character Recognition with Convolutional Neural Network

```
1: Set the directory path where the images are stored.
2: Get a sorted list of classes in the directory.
3: Count the number of classes ($n$) and assign labels.
4: Initialize empty lists images[] and labels[].
5: for each class do
6:     Create the path to the class directory.
7:     for each image in the class directory do
8:         Read the preprocessed image.
9:         Convert the image to binary.
10:        Resize the image to $64 \times 64$ pixels.
11:        Append image and class index to images[] and labels[].
12:    end for
13: end for
14: Convert lists to NumPy arrays.
15: Perform one-hot encoding on the labels.
16: for each class do
17:     Compute class weights using maximum class frequency.
18: end for
19: Split dataset into training (80%) and validation (20%) using stratified split.
20: for $i = 1$ to $3$ do
21:     Add Conv2D layer with filters = [32, 64, 128][$i$], kernel size $(3, 3)$.
22:     Add ReLU activation.
23:     Add MaxPool2D layer with pool size $(2, 2)$.
24: end for
25: Flatten the output of convolutional layers.
26: for $i = 1$ to $2$ do
27:     Add Dense layer with units = [64,128][$i$] and ReLU activation.
28: end for
29: Add Dense output layer with $n$ units and softmax activation.
30: Compile model with categorical cross-entropy loss and Adam optimizer.
31: Train model using fit() with training/validation data, epochs, batch size, class weights, and checkpoint.
32: Make predictions on test dataset.
```

resulting in differing class counts. During testing with the IIT BBSR dataset, basic characters were exclusively selected to ensure compatibility. Furthermore, when evaluating a model trained on the IIT BBSR dataset, the last fully connected layer was adjusted during testing by configuring the number of neurons to match the class count present in the testing dataset. The testing accuracy of the proposed model, trained on one of the benchmark datasets and tested on other benchmark and real-life data, is shown in Table 4 and 5:

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{10}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{11}$$

$$\text{F1 Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{}\text{Precision} + \text{Recall}, \tag{12}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}. \tag{13}$$

TP : Prediction of True Positives cases
TN : Prediction of True Negative cases,
FP : Prediction of False Positive cases
FN : Prediction of False Negative cases

6.1 Result Comparison

The quality of the suggested approach is validated through a rigorous comparison based on recognition rates with existing literature. Table 6 summarizes key findings from several studies on the datasets under consideration. Notably, our proposed model consistently outperforms prior work in terms of recognition rate across multiple benchmark datasets. The proposed model achieves peak accuracy on numerical datasets such as ISI Kolkata, IIT BBSR, and IIIT BBSR. Additionally, it achieves peak accuracy on character datasets from NIT Rourkela and IIIT BBSR, demonstrating its efficacy across diverse contexts.

ISSN 2007-9737

Table 6. Comparative analysis of recognition accuracy between existing literature and the proposed model on Odia handwritten datasets

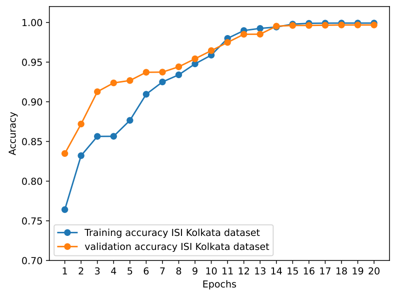
Literature	Dataset	Samples	Method	Accuracy (%)
Das et al., 2019 [8]	ISI Kolkata	Digit	ELM	94.47
Sethy et al., 2020 [40]	ISI Kolkata	Digit	CNN	98.40
Das et al., 2020 [9]	ISI Kolkata	Digit	SVM	97.70
Dash et al., 2020 [13]	ISI Kolkata	Digit	MQDF	99.22
R. Dey et al., 2021 [16]	ISI Kolkata	Digit	CNN	98.22
Proposed	ISI Kolkata	Digit	CNN	**99.44**
Dash et al., 2017 [10]	IIT Bhubaneswar	Digit	KNN, SCCST	99.20
Sethy et al., 2020 [40]	IIT Bhubaneswar	Digit	CNN	97.71
R. Dey et al., 2021 [16]	IIT Bhubaneswar	Digit	RNN	97.34
Proposed	IIT Bhubaneswar	Digit	CNN	**99.27**
R. Dey et al., 2021 [16]	IIIT Bhubaneswar	Digit	CNN	98.72
Proposed	IIIT Bhubaneswar	Digit	CNN	**98.82**
Das et al., 2020 [9]	NIT Rourkela	Basic chars	RF	98.90
Sethy et al., 2019 [37]	NIT Rourkela	Basic chars	RBF NN	98.80
R. Dey et al., 2021 [16]	NIT Rourkela	Basic chars	CNN	93.35
Proposed	NIT Rourkela	Basic chars	CNN	**99.18**
Dash et al., 2017 [10]	IIT Bhubaneswar	Characters	KNN, SCCST	96.35
Dash et al., 2020 [13]	IIT Bhubaneswar	Characters	MQDF	93.24
R. Dey et al., 2021 [16]	IIT Bhubaneswar	Characters	CNN	88.23
Proposed	IIT Bhubaneswar	Characters	CNN	94.10
R. Dey et al., 2021 [16]	IIIT Bhubaneswar	Basic chars	CNN	83.56
Proposed	IIIT Bhubaneswar	Basic chars	CNN	**96.89**

7 Conclusion and Future Scope

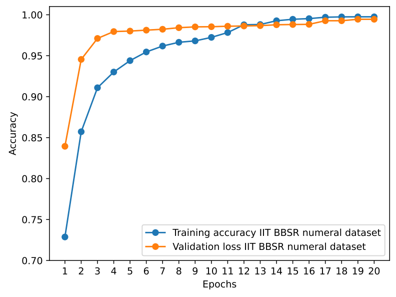
The Odia language, spoken by over 50 million people worldwide, is one of India's most widely spoken languages. Despite its widespread use, Odia remains a low-resource language in the fields of natural language processing and pattern recognition, with limited availability of large, annotated datasets and robust tools. There is a significant demand for an offline handwritten character recognition system for the Odia language, encompassing numerals, basic characters, and compound characters.

Although computers and smartphones are increasingly integrated into daily life, research efforts focused on handwritten character recognition in Odia are still sparse. This study aims to address this gap by focusing on offline recognition of Odia handwritten characters, particularly within the constraints of low-resource language settings. Deep learning models are highly effective at recognizing patterns, yet their accuracy can drop when working with large images or samples that contain a lot of noise. To address these issues, a series of preprocessing steps were carried out to clean the data and bring the handwritten inputs into a uniform format. In addition, data augmentation was used to introduce realistic variations, including rotations and changes in brightness and contrast, so that the model could better handle differences in writing style, orientation, and lighting conditions.

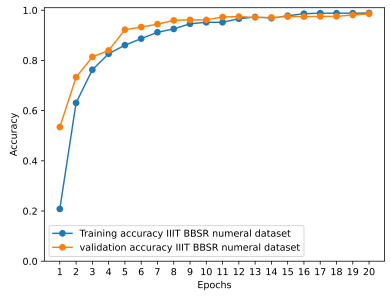
In this work, a lightweight CNN model was



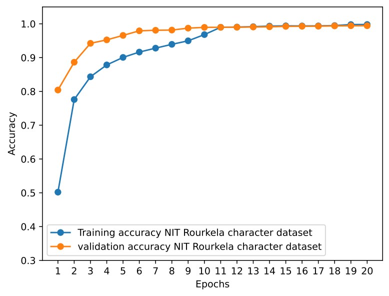
(a) ISI kolkata



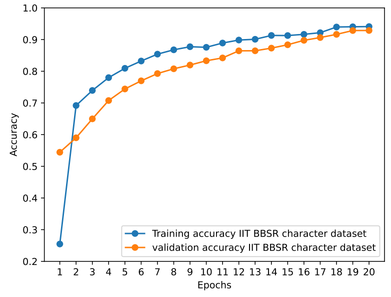
(b) IIT BBSR num.



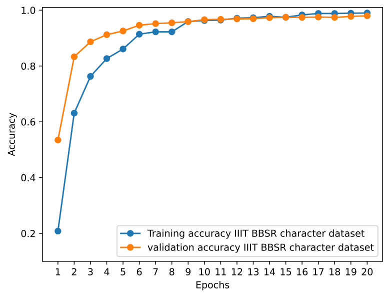
(c) IIIT BBSR num.



(d) NIT Rourkela



(e) IIT BBSR char.



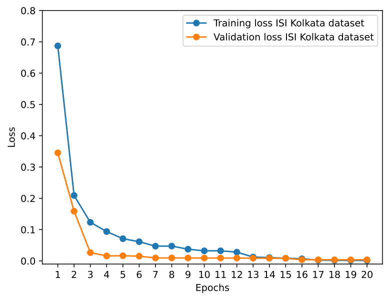
(f) IIIT BBSR char.

Fig. 5. Accuracy plot of model on benchmark datasets

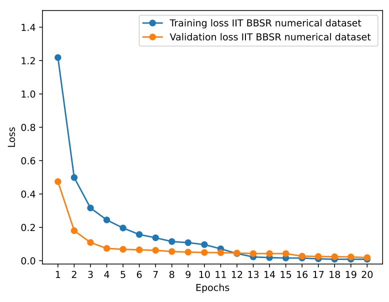
designed, trained, and validated on multiple benchmark datasets. The datasets underwent carefully chosen preprocessing steps to reduce noise and preserve essential character details, enabling the model to achieve strong pattern recognition.

Experimental results show that the proposed model consistently outperforms existing approaches, achieving state-of-the-art accuracy on all numeral datasets as well as on the NIT Rourkela and IIIT Bhubaneswar character datasets.

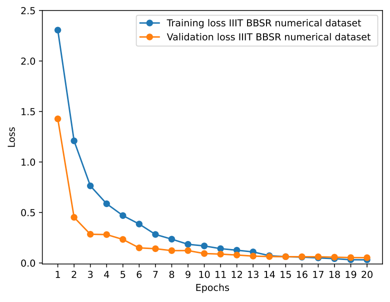
Additional evaluations on benchmark and real-life samples further demonstrate the model's robustness and reliability. Looking ahead, extending the framework to handle complete words or full-page handwritten documents could open new possibilities for document retrieval and large-scale handwriting analysis.



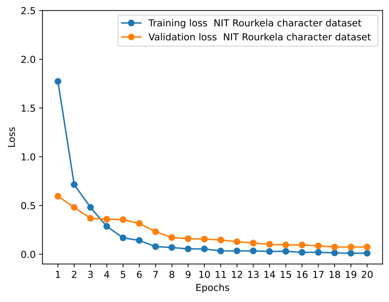
(a) ISI kolkata



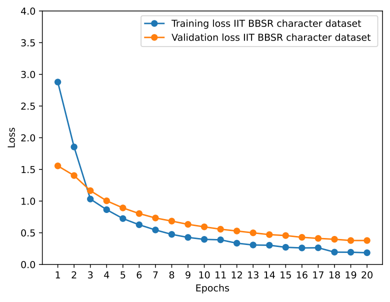
(b) IIT BBSR num.



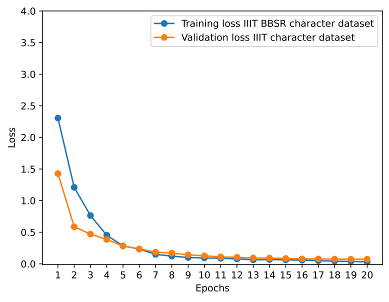
(c) IIIT BBSR num.



(d) NIT Rourkela



(e) IIT BBSR chars.



(f) IIIT BBSR chars.

Fig. 6. Loss plot of model on benchmark datasets

ISSN 2007-9737

Overall, the findings highlight the effectiveness of the proposed model and its potential contribution to the broader field of character recognition and document processing.

References

1. **Baruch, O. (1988).** Line thinning by line following. Pattern Recognition Letters, Vol. 8, No. 4, pp. 271–276.
2. **Bhattacharya, U., Chaudhuri, B. (2005).** Databases for research on recognition of handwritten characters of Indian scripts. Eighth International Conference on Document Analysis and Recognition (ICDAR'05), IEEE, pp. 789–793.
3. **Bhowmik, T. K., Parui, S. K., Bhattacharya, U., Shaw, B. (2006).** An HMM based recognition scheme for handwritten Oriya numerals. 9th International Conference on Information Technology (ICIT'06), IEEE, pp. 105–110.
4. **Bock, S., Weiß, M. (2019).** A proof of local convergence for the Adam optimizer. 2019 international joint conference on neural networks (IJCNN), IEEE, pp. 1–8.
5. **Castro, D., Zanchettin, C., Amaral, L. A. N. (2024).** On the improvement of handwritten text line recognition with octave convolutional recurrent neural networks. International Journal on Document Analysis and Recognition (IJDAR), pp. 1–15.
6. **Chooi, S. L., Ab Ghafar, A. S. B. (2021).** Handwritten character recognition using convolutional neural network. Progress in Engineering Application and Technology, Vol. 2, No. 1, pp. 593–611.
7. **Das, D., Dash, R., Majhi, B. (2018).** Optimization based feature generation for handwritten Odia-numeral recognition. 2018 Fourteenth International Conference on Information Processing (ICINPRO), IEEE, pp. 1–5.
8. **Das, D., Nayak, D. R., Dash, R., Majhi, B. (2019).** An empirical evaluation of extreme learning machine: application to handwritten character recognition. Multimedia Tools and Applications, Vol. 78, pp. 19495–19523.
9. **Das, D., Nayak, D. R., Dash, R., Majhi, B. (2020).** MJCN: Multi-objective jaya convolutional network for handwritten optical character recognition. Multimedia Tools and Applications, Vol. 79, pp. 33023–33042.
10. **Dash, K. S., Puhan, N., Panda, G. (2016).** A sparse concept coded spatio-spectral feature representation for handwritten character recognition. 2016 International Conference on Signal Processing and Communications (SPCOM), IEEE, pp. 1–5.
11. **Dash, K. S., Puhan, N. B., Panda, G. (2015).** Handwritten numeral recognition using non-redundant stockwell transform and bio-inspired optimal zoning. IET Image processing, Vol. 9, No. 10, pp. 874–882.
12. **Dash, K. S., Puhan, N. B., Panda, G. (2017).** Odia character recognition: a directional review. Artificial Intelligence Review, Vol. 48, pp. 473–497.
13. **Dash, K. S., Puhan, N. B., Panda, G. (2020).** Sparse concept coded tetrolet transform for unconstrained Odia character recognition. arXiv preprint arXiv:2004.01551.
14. **Dash, P. R., Balabantaray, R. C., Dey, R. (2024).** An approach for handwritten alphanumeric character recognition: Leveraging cnn for accurate recognition. 2024 1st International Conference on Cognitive, Green and Ubiquitous Computing (IC-CGU), IEEE, pp. 1–6.
15. **Dehghanian, A., Ghods, V. (2018).** Farsi handwriting digit recognition based on convolutional neural networks. 2018 6th International Symposium on Computational and Business Intelligence (ISCBI), IEEE, pp. 65–68.
16. **Dey, R., Balabantaray, R. C., Mohanty, S. (2022).** Offline Odia handwritten character recognition with a focus on compound characters. Multimedia Tools and Applications, Vol. 81, No. 8, pp. 10469–10495.

17. **Dey, R., Balabantaray, R. C., Piri, J., Singh, D. (2021).** Offline natural scene character recognition using vgg16 neural networks. 2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA), IEEE, pp. 946–951.

18. **D'souza, R. N., Huang, P.-Y., Yeh, F.-C. (2020).** Structural analysis and optimization of convolutional neural networks with a small sample size. Scientific reports, Vol. 10, No. 1, pp. 834.

19. **Gysel, P., Motamedi, M., Ghiasi, S. (2016).** Hardware-oriented approximation of convolutional neural networks. arXiv preprint arXiv:1604.03168.

20. **Hochuli, A. G., Oliveira, L. S., Britto Jr, A., Sabourin, R. (2018).** Handwritten digit segmentation: Is it still necessary?. Pattern Recognition, Vol. 78, pp. 1–11.

21. **Huh, J.-H. (2020).** Surgery agreement signature authentication system for mobile health care. Electronics, Vol. 9, No. 6, pp. 890.

22. **Ignat, A., Aciobanitei, B. (2016).** Handwritten digit recognition using rotations. 2016 18th International symposium on symbolic and numeric algorithms for scientific computing (SYNASC), IEEE, pp. 303–306.

23. **Jindal, T., Bhattacharya, U. (2013).** Recognition of offline handwritten numerals using an ensemble of mlps combined by adaboost. Proceedings of the 4th International Workshop on Multilingual OCR, pp. 1–5.

24. **Khan, S., Hafeez, A., Ali, H., Nazir, S., Hussain, A. (2020).** Pioneer dataset and recognition of handwritten Pashto characters using convolution neural networks. Measurement and Control, Vol. 53, No. 9-10, pp. 2041–2054.

25. **Kulkarni, S. R., Rajendran, B. (2018).** Spiking neural networks for handwritten digit recognition—supervised learning and network optimization. Neural Networks, Vol. 103, pp. 118–127.

26. **Li, P., He, X., Song, D., Ding, Z., Qiao, M., Cheng, X., Li, R. (2021).** Improved categorical cross-entropy loss for training deep neural networks with noisy labels. Pattern Recognition and Computer Vision: 4th Chinese Conference, PRCV 2021, Beijing, China, October 29–November 1, 2021, Proceedings, Part IV 4, Springer, pp. 78–89.

27. **Maalej, R., Kherallah, M. (2020).** Improving the DBLSTM for online Arabic handwriting recognition. Multimedia Tools and Applications, Vol. 79, pp. 17969–17990.

28. **Majhi, B., Pujari, P. (2018).** On development and performance evaluation of novel odia handwritten digit recognition methods. Arabian Journal for Science and Engineering, Vol. 43, pp. 3887–3901.

29. **Mohapatra, R. K., Mishra, T. K., Panda, S., Majhi, B. (2015).** OHCS: A database for handwritten atomic Odia character recognition. 2015 Fifth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), IEEE, pp. 1–4.

30. **Otsu, N. (1979).** A threshold selection method from gray-level histograms. IEEE transactions on systems, man, and cybernetics, Vol. 9, No. 1, pp. 62–66.

31. **Panda, R., Dash, S., Padhy, S., Nayak, M. (2022).** CNN based handwritten Odia character recognition. 2022 International Conference on Machine Learning, Computer Systems and Security (MLCSS), pp. 267–273. DOI: `10.1109/MLCSS57186.2022.00056`.

32. **Ram, S., Gupta, S., Agarwal, B. (2018).** Devanagri character recognition model using deep convolution neural network. Journal of Statistics and Management Systems, Vol. 21, No. 4, pp. 593–599.

33. **Rodríguez, P., Bautista, M. A., Gonzalez, J., Escalera, S. (2018).** Beyond one-hot encoding: Lower dimensional target embedding. Image and Vision Computing, Vol. 75, pp. 21–31.

34. **Roy, K., Pal, T., Pal, U., Kimura, F. (2005).** Oriya handwritten numeral recognition system.

Eighth International Conference on Document Analysis and Recognition (ICDAR'05), IEEE, pp. 770–774.

35. **Sarangi, P. K., Ahmed, P., Ravulakollu, K. K., et al. (2014).** Naïve Bayes classifier with lu factorization for recognition of handwritten odia numerals. Indian Journal of Science and Technology, Vol. 7, No. 1, pp. 35–38.

36. **Sethi, R. K., Mohanty, K. K. (2020).** Optical odia character classification using cnn and transfer learning: A deep learning approach. International Research Journal of Engineering and Technology (IRJET), Vol. 7, No. 07, pp. 3885–3890.

37. **Sethy, A., Patra, P. K. (2019).** Off-line odia handwritten character recognition: an axis constellation model-based research. International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol. 8, No. 9S2, pp. 788–793.

38. **Sethy, A., Patra, P. K., Nayak, D. R. (2018).** Off-line handwritten Odia character recognition using DWT and PCA. Progress in Advanced Computing and Intelligent Engineering: Proceedings of ICACIE 2016, Volume 1, Springer, pp. 187–195.

39. **Sethy, A., Patra, P. K., Nayak, S., Jena, P. M. (2017).** Symmetric axis based off-line odia handwritten character and numeral recognition. 2017 3rd International Conference on Computational Intelligence and Networks (CINE), IEEE, pp. 83–87.

40. **Sethy, A., Patra, P. K., Nayak, S. R. (2020).** Offline handwritten numeral recognition using convolution neural network. Machine Vision Inspection Systems: Image Processing, Concepts, Methodologies and Applications, Vol. 1, pp. 197–212.

41. **Singh, P. K., Sarkar, R., Das, N., Basu, S., Kundu, M., Nasipuri, M. (2018).** Benchmark databases of handwritten Bangla-Roman and Devanagari-Roman mixed-script document images. Multimedia Tools and Applications, Vol. 77, pp. 8441–8473.

42. **Tripathy, N., Panda, M., Pal, U. (2003).** System for Oriya handwritten numeral recognition. Document recognition and retrieval XI, SPIE, Vol. 5296, pp. 174–181.

43. **Tushar, A. K., Ashiquzzaman, A., Afrin, A., Islam, M. R. (2018).** A novel transfer learning approach upon Hindi, Arabic, and Bangla numerals using convolutional neural networks. Computational Vision and Bio Inspired Computing, Springer, pp. 972–981.

44. **Wu, F., Zhu, C., Xu, J., Bhatt, M. W., Sharma, A. (2022).** Research on image text recognition based on canny edge detection algorithm and k-means algorithm. International Journal of System Assurance Engineering and Management, Vol. 13, No. Suppl 1, pp. 72–80.

45. **Yue, X., Wang, Z., Ishibashi, R., Kaneko, H., Meng, L. (2024).** An unsupervised automatic organization method for professor Shirakawa's hand-notated documents of oracle bone inscriptions. International Journal on Document Analysis and Recognition (IJDAR), pp. 1–19.

Article received on 23/04/2025; accepted on 22/07/2025.
**Corresponding author is Pragnya Ranjan Dash.*