Printed OCR for Extremely Low-resource Indic Languages

Alik Sarkar, Ajoy Mondal, Gurpreet Singh Lehal, and C. V. Jawahar

CVIT, International Institute of Information Technology, Hyderabad, India
sarkaralikphy@gmail.com, gs.lehal@research.iiit.ac.in,
{ajoy.mondal,jawahar}@iiit.ac.in

Abstract. Indic languages represent a significant aspect of India's cultural heritage, embodying collective knowledge, traditions, and customs. Preserving this heritage is crucial. Optical Character Recognition (OCR) technology aids in simplifying text recognition tasks by extracting text from images. This study uses an established OCR model to digitize document images of extremely low-resource Indian languages, which previous OCR efforts did not focus on. Preparing corpora for such languages is challenging due to the scarcity of expert linguists and the required time and resources. We introduce a synthetic dataset, *Mozhi-LR(S)* and a real dataset, *Mozhi-LR(R)*, comprising word level images with textual transcriptions for these nine languages. The model is trained using synthetic datasets and fine-tuned with real ones, achieving high accuracy on synthetic and real datasets. We also offer APIs for our OCR models and web-based applications that incorporate these APIs. This integration facilitates the digitization of Indic printed documents in extremely low-resource languages. The trained models, code, and datasets are publicly available at https://github.com/ALIKSARKAR/Printed-OCR-for-Extremely-Low-resource-Indic-Languages.

Keywords: OCR · Indic language · Indic script · low-resource language · printed text.

1 Introduction

India is rich in culture, most reflected in its languages. Languages are crucial for maintaining Indian culture, as they communicate knowledge and identity. With over sixteen hundred languages spoken, India has one of the highest linguistic diversities globally. Preserving cultural heritage is vital to representing collective knowledge, traditions, and customs. Preserving ancient languages like Sanskrit provides insights into India's rich cultural history and appreciation of diverse traditions and customs. Optical Character Recognition (OCR) is crucial for preserving these languages by digitizing documents.

OCR is a technology that extracts text from images and serves various practical purposes. The OCR process typically involves two main steps: text detection and text recognition, which can be executed separately or concurrently [26]. Text

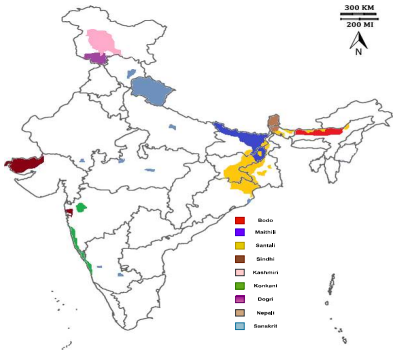

Fig. 1. Displays the regions on the map of India highlighted with colors where low-resource languages are spoken. Best view in Zoom.

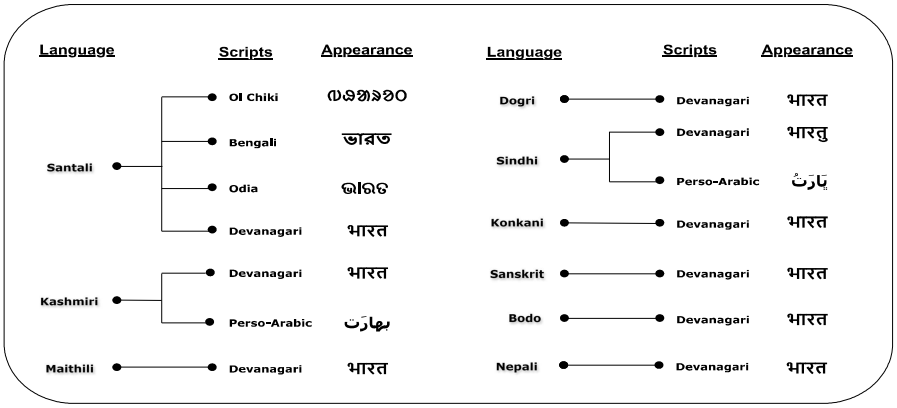

Fig. 2. We study printed text recognition of nine Indian low-resource languages (four different scripts). Languages such as Bodo, Dogri, Konkani, Maithili, Nepali, and Sanskrit use the common script Devanagari. We show how the name "Bharat" is written in all nine languages.

detection identifies text regions within an image, followed by text recognition, where the identified text is transcribed. The recognition task may vary based on the image type, such as handwritten recognition (HWR), document OCR, and scene text recognition (STR). The challenges encountered in text recognition depend on factors such as the language or script used, the text's rendering (handwritten, printed, or typewritten), and how the document is captured (scanned, photographed with a mobile device, or born-digital).

The 2011 official census of India [1] lists thirty Indian languages with more than a million native speakers. Twenty-two of them are granted scheduled (constitutionally recognized) language status, written in (at least) 13 different ex-

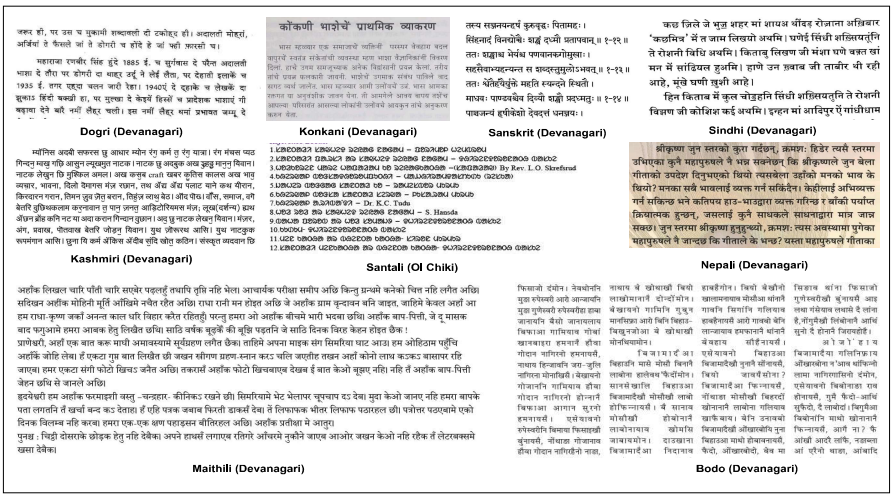

Fig. 3. Showcases a selection of cropped images from real document pages featuring the nine low-resource languages, each represented in its respective script. Best view in Zoom.

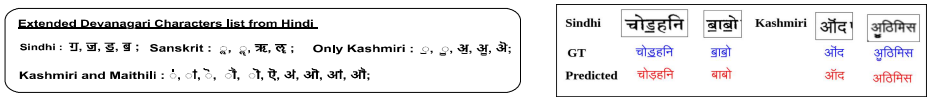

Fig. 4. (a) Shows characters which are used in Sindhi, Maithili, and Kashmiri. (b) The Hindi model is unable to recognize words with extended characters that are not present in Hindi words.

tant scripts and several other variations of them. These twenty-two languages belong to three different language families: *Indo-European, Dravidian, and Sino-Tibetan*. Among those languages, thirteen are high-resource (the languages for which documents or other resources are readily available): *Assamese, Bengali, Gujarati, Hindi, Kannada, Malayalam, Manipuri, Marathi, Odia, Punjabi, Tamil, Telugu, and Urdu*. Many of them share common linguistic and grammatical structures. However, the script remains very different, except for a few languages. The remaining nine languages are low-resource: *Bodo, Dogri, Kashmiri, Konkani, Maithili, Nepali, Sanskrit, Santali, and Sindhi*. Fig. 1 highlights the regions of India where low-resource languages are spoken. While most of these low-resource languages use Devanagari as their script, some use different scripts. Santali uses Bengali, Odia, Devanagari, and Ol Chiki as their script. Kashmiri uses Devanagari and Perso-Arabic scripts. Fig. 4(a) shows characters that are used in Sindhi, Maithili, and Kashmiri. Fig. 4(b) illustrates that the Hindi OCR model struggles to recognize words in Sindhi and Kashmiri due to the presence

of extended characters not found in Hindi words. These experiments underscore the necessity of developing separate OCR models tailored to these low-resource languages. Fig. 2 shows how the word "Bharat" is written in these languages. Fig. 3 depicts a sample of cropped images from nine languages from our newly created *Mozhi-LR(R)* dataset.

Despite extensive endeavors spanning from the 1970s to date [40,8,4], achieving satisfactory results across multiple languages and document types remained challenging. The complexity of scripts and languages and the scarcity of large-scale annotated data presented formidable barriers to progress in Indian language OCR. While significant attention has been devoted to high-resource Indic languages [38,23,21,24,27,28,36], only some studies [6,35,11] have addressed low-resource languages due to their limited resources. To fill the gap in research for low-resource languages, we leverage an established OCR model to digitize documents in nine such languages. We create a synthetic data set of word level images with their textual transcription of nine languages. We also create a real dataset of manually annotated word level images with their textual transcription. The model is trained with synthetic data sets and fine-tuned with real datasets. Finally, it is evaluated on a real and synthetic dataset and achieves high accuracy. The APIs corresponding to our developed models are integrated into Bhasini[1] for public use.

The contribution of this work is as follows:

- Generate a synthetic dataset named *Mozhi-LR(S)*, comprising word level images and their corresponding textual transcriptions for the nine languages — *Bodo, Dogri, Kashmiri, Konkani, Maithili, Nepali, Sanskrit, Santali, and Sindhi* (refer Table 1 and Fig. 6).
- Generate a real dataset named *Mozhi-LR(R)*, consisting of manually annotated word level images and their corresponding textual transcriptions from real documents (refer Table 1 and Fig. 8).
- We develop OCR models for low-resource Indic languages and evaluate their performance on both synthetic and real datasets, demonstrating significant accuracy improvements across all languages. These findings highlight the effectiveness of our approach in overcoming the challenges posed by low-resource languages (see Table 2 for details).
- Provide APIs for our OCR models and web-based applications that seamlessly integrate these APIs, facilitating the digitization of Indic printed documents in low-resource languages.

2 Related Work

2.1 OCR on High-resource Indic Languages

Initially, OCR systems [8,40] for Indian languages typically follow a template-matching approach to match characters, relying on intuitive features like shape

[1] https://bhashini.gov.in/

and water reservoir. Pal and Chaudhuri [34] provides a comprehensive overview of the methods developed during this era. In the initial stage of OCR, various methods such as [40,2,8,32,33,25] follow the pipeline — segmentation of words into characters, which are then classified using various classifiers.

In the subsequent phase of OCR development, statistical and data-driven methods gained prominence, incorporating techniques like Discrete Cosine Transform (DCT) and Principal Component Analysis (PCA). Support Vector Machines (SVM) and Artificial Neural Networks (ANN) emerged as classifiers. Several works [3,5,42,29,37,37,22,31] have been done in this directions. Arya *et al.* [4] provide a comparative analysis of leading OCR systems developed during this period.

Recent advancements in OCR for Indian scripts have focused on segmentation-free methods, which directly generate label sequences from word or line images. Sankaran *et al.* [38] introduced the use of Connectionist Temporal Classification (CTC)-based sequence modeling for recognizing Indian printed text. They employed an RNN encoder and CTC transcription to map feature sequences from Devanagari word images to class label sequences. This approach was further refined in [30], where the feature sequence from the word image is directly mapped to the Unicode sequence, eliminating the need for rule-based Akshara to Unicode mapping. Adopting CTC-based transcription provided a solution for sub-word segmentation challenges in Indic scripts, enabling direct transcription of word images into machine-readable Unicode sequences. Krishnan *et al.* [23] employed profile-based features and CTC-based models similar to [30] for recognizing seven Indian languages. Their evaluation of extensive document image datasets per language demonstrated the effectiveness of a unified CTC transcription framework for multi-language recognition, eliminating the need for language-specific components.

Hasan *et al.* [41] introduced an RNN+CTC model for recognizing printed Urdu text, which directly generates a Unicode sequence from a text line image, with lines as the recognition unit. Chavan et al.[9] conducted a comparative study evaluating the performance of an RNN encoder and a multidimensional RNN (MDRNN) [14] encoder in conjunction with CTC transcription. They utilized HOG (Histogram of Gradients) features with the RNN encoder and raw pixels with the MDRNN. Their findings indicate that the MDRNN encoder outperforms the RNN encoder. Paul *et al.* [36] proposed an RNN+CTC transcription model for recognizing Bengali script. Additionally, Kundaikar and Pawar [24] investigated the robustness of CTC-based Devanagari OCR to font and font size variations. Significant efforts have been directed towards building OCR models for high-resource languages in India, whereas relatively few works focus on low-resource languages.

2.2 OCR on Extremely Low-resource Indic Languages

While research in resource-scarce Indian languages remains limited, some efforts have been made in languages derived from the Devanagari script. Dwivedi *et al.* [11] proposed an encoder-decoder model for recognizing Sanskrit texts.

Hasan *et al.* [16] explored transformer models for recognizing Nepali text written in the Devanagari script. Additionally, a few studies have been conducted in languages such as Sindhi [15], Kashmiri [7], and Sanskrit [39], albeit the number of works in these languages remains limited. The scarcity of research in this domain motivates us to explore and investigate further.

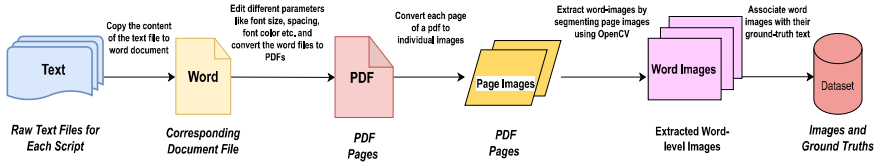

Fig. 5. Shows a pipeline for generation of synthetic word level images and their corresponding textual transcriptions.

3 Dataset

Synthetic Dataset: Real-world data collection is time-consuming and challenging, requiring millions of images for specific tasks. Additionally, the annotation process is prone to human errors despite efforts to minimize them. Synthetic data has emerged as an alternative source to address these limitations. Generated by computers with minimal effort, synthetic data is cost-effective and scalable [18,19]. For this study, we create a dataset *Mozhi-LR(S)*, by generating a large number of synthetic word level images and their corresponding textual transcriptions for each of the nine languages. We follow the pipeline for creating the synthetic dataset illustrated in Fig. 5.

(i) *Text File:* We start with the collection of raw text files for each of the ten languages, each representing a distinct linguistic dataset. (ii) *Word File:* copy the content of each text file to a corresponding Microsoft Word document to create a Word file. In these Word files, we edit the text by changing font sizes, font colors, and background colors. It introduces variation in the text's appearance, enhancing the dataset's diversity. (iii) *PDF:* we transform the edited Word documents into PDF files. (iv) *Page Image:* we convert the PDF files into a combination of page images. (v) *Word Level Image:* using OpenCV, we extract word-level images from these page images. (vi) *Mapping:* We establish a mapping between each word-level image and its corresponding text to ensure alignment between the visual representation and the original text content. (vii) *Synthetic Dataset:* gather the word images and corresponding textual transcriptions, assembling a comprehensive synthetic dataset suitable for creating and evaluating OCR models. Table 1 shows the statistics of the created synthetic dataset. Fig. 6 shows a few sample word level images and corresponding textual transcriptions from the generated synthetic dataset.

Script	Language	Mozhi-LR(R)			Mozhi-LR(S)		
		Train	Val	Test	Train	Val	Test
Devanagari	Maithili	2358	336	675	91849	13121	26243
	Sindhi	3567	509	1021	183212	26174	52346
	Dogri	3260	465	933	20225	2891	5780
	Konkani	5282	754	1511	74028	10578	21152
	Sanskrit	4767	681	1363	-	-	-
	Nepali	5028	718	1437	41152	5931	11861
	Bodo	35037	5004	1014	22415	3203	6405
	Kashmiri	3899	557	1115	-	-	-
Ol Chiki	Santali	-	-	-	19811	2830	5661

Table 1. Presents the statistics of the created real *Mozhi-LR(R)* and synthetic *Mozhi-LR(S)* datasets.

Fig. 6. Shows synthetic word samples for three languages, Santali, Sindhi, and Maithili.

Real Dataset: Our real dataset, *Mozhi-LR(R)*, consists of 20-25 document images per language sourced from various books and scanned using a flatbed scanner at 300 DPI. These pages typically feature single-column text arranged in paragraphs with simple layouts. For languages *Bodo, Dogri, Kashmiri, Konkani, Maithili, Nepali, Sanskrit, and Sindhi*, pages are in *Devanagari* script. Each page is manually annotated with word bounding boxes and corresponding text transcriptions for the word level images. Fig. 8 showcases samples from *Mozhi-LR(R)*, highlighting the dataset's diversity in terms of fonts, text sizes, colors, orientations, lighting conditions, noises, styles, and backgrounds.

4 Baseline for Text Recognition

We utilize the network architecture proposed by Gongidi *et al.* [12] depicted in Fig. 9, as the baseline for our experiment. This network consists of four main modules: the Transformation Network Module (TM), Feature Extractor Module (FEM), Sequence Modeling Module (SMM), and Predictive Modeling Module (PMM).

Transformation Network Module (TM): This module transforms the input image X into the normalized image $\tilde{X}$. Printed text images often exhibit font

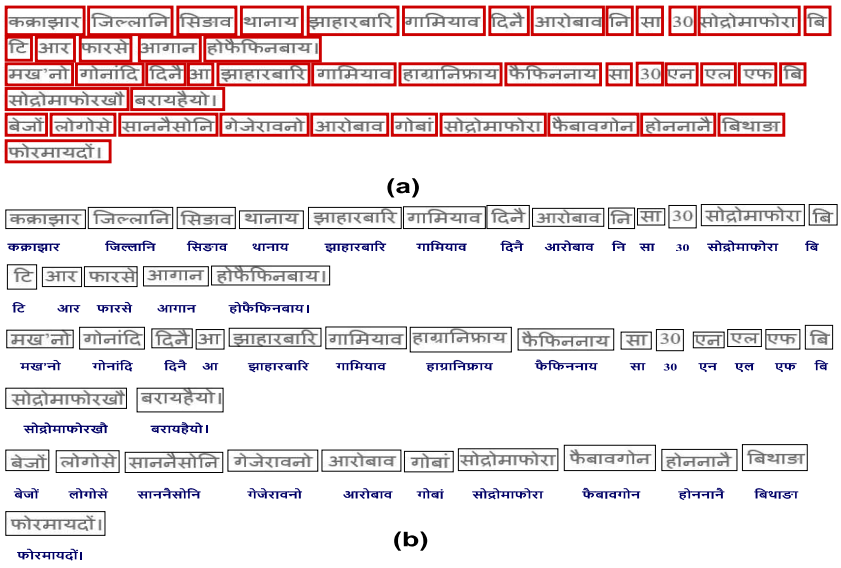

Fig. 7. Shows how we annotated real word level images and corresponding textual transcriptions. (a) Cropped page image with bounding boxes, and (b) Cropped word images extracted by bounding boxes and their corresponding manually annotated ground truth highlighted in blue.

Fig. 8. Shows real word samples for two scripts. Devanagari script for Bodo, Maithili and Sindhi languages.

styles, sizes, and orientations variations, posing challenges for accurate recognition. If these input images are used without alteration, the subsequent feature extraction stage must learn to account for these variations. A transformation block is employed to apply input-specific geometric transformations to simplify the text recognition task. Thin-plate spline (TPS) [20] and affine transformations (ATN) are commonly utilized methods to rectify input images. The Affine module adjusts the scale, translation, and shear, while TPS applies a non-rigid transformation by identifying fiducial points along the upper and bottom edges of the word region.

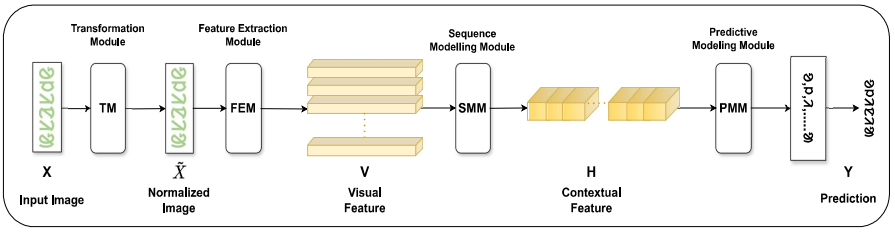

Fig. 9. Shows a standard pipeline for text recognition is depicted below, illustrating the process from input image X to text prediction Y. The example showcased here is written in the Ol Chiki script.

Feature Extraction Module (FEM): A convolutional neural network, such as ResNet [17], processes an input image (i.e., X or $\tilde{X}$) to produce a visual feature map $V = v_i$, where i ranges from 1 to I (the number of columns in the feature map). Each column in the feature map corresponds to a distinct receptive field along the horizontal axis of the input image. These features are then utilized to predict the character associated with each receptive field.

Sequence Modelling Module (SMM): The features extracted from the feature extraction module (FEM) are restructured into a sequence V, where each column v_i represents a frame of the sequence. However, this sequence may lack contextual information. To address this issue, similar to previous works [10,41,38,23], we adopt a 2-layer BiLSTM architecture with 256 hidden neurons in each layer as the sequence modeling (SM) module in our experiments to make better sequence $H = \text{Seq}(V)$.

Predictive Modeling Module (PMM): This module decodes a character sequence from the contextual feature H. Using the input H, the module predicts a sequence of characters $Y = y_1, y_2, \ldots, y_n$. Connectionist Temporal Classification (CTC) [13] is a commonly used method for achieving this task. CTC predicts a character in each column $h_i \in H$ and transforms the entire character sequence into a variable length stream by removing repeated characters and blanks.

5 Experiments and Results

5.1 Implementation Details

In our experiments, we standardized the cropped words to 32 pixels, converted them to grayscale, and adjusted the aspect ratio to 96 $\times$ 256. Data sets are randomly split into a 4:1 ratio for train and test sets, respectively, for both synthetic and real data across all languages. Additionally, we allocated 12.50% of the train split for validation, ensuring its representation of the training data while keeping the test set distinct. The character set comprised language-specific characters, symbols, and digits, with varying character counts for training depending on the

language. Our model employed a bi-directional LSTM with 256 hidden units per direction over two layers, yielding an output size of 2×256 at each time step. Implementation was done using PyTorch, based on an existing CRNN architecture [12], with training conducted on a single Nvidia GeForce 1080 Ti GPU. For fine-tuning Devanagari script-based languages (*Sindhi, Maithi, Dogri, Kashmiri, Konkani, Nepali, Sanskrit, and Bodo*), we used the AdaDelta with a 0.95 decay rate. The training was performed for 50 epochs using the pre-trained Hindi model on real and synthetic datasets. The batch size and learning rate were set to 32 and 1.0, respectively, with gradient clipping applied at a magnitude of 5.

Script	**Language**	**Mozhi-LR(R)**		**Mozhi-LR(S)**	
		WRR	**CRR**	**WRR**	**CRR**
Devanagari	Maithili	92.74	97.85	96.23	98.12
	Sindhi	80.22	94.05	94.58	98.83
	Dogri	91.85	97.30	96.76	98.94
	Konkani	89.28	97.29	95.27	97.75
	Sanskrit	86.57	94.96	-	-
	Nepali	82.60	95.13	93.07	96.73
	Bodo	93.84	97.22	94.95	97.14
	Kashmiri	87.71	93.80	-	-
Ol Chiki	Santali	-	-	90.60	96.58

Table 2. Shows results on our created datasets using baseline.

5.2 Training and Testing Details

We use the baseline model to train and evaluate the performance on the Ol Chiki script in the *Mozhi-LR(S)* dataset for Santali. For the remaining eight Devanagari-based languages, we fine-tune the Hindi pre-trained model on both the *Mozhi-LR(S)* and *Mozhi-LR(R)* datasets. For Santali (Ol Chiki script), we only showcase the results for our *Mozhi-LR(S)* dataset due to the limited number of real word images collected for *Mozhi-LR(R)* for evaluation. We evaluate our model solely on the *Mozhi-LR(R)* dataset for Sanskrit and Kashmiri.

5.3 Evaluation Metrics

Two popular evaluation metrics — *Character Recognition Rate (CRR)* (alternatively *Character Error Rate, CER*) and *Word Recognition Rate (WRR)* (alternatively *Word Error Rate, WER*) are used to evaluate the performance of recognizers. *Error Rate (ER)* is defined as

$$ER = \left(\frac{S + D + I}{N}\right), \tag{1}$$

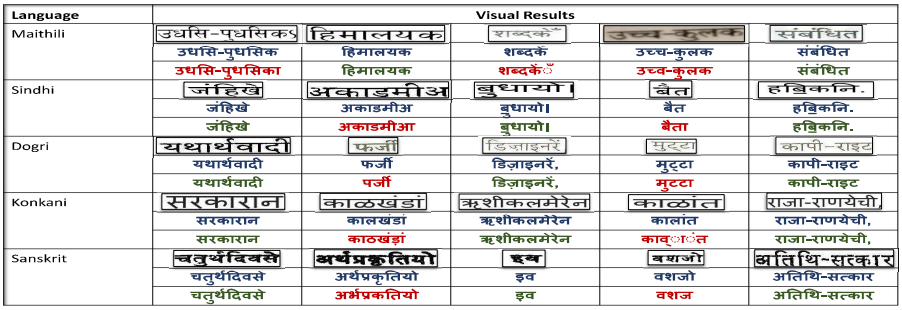

Fig. 10. Shows visual results on word level images for Maithili, Sindhi, Dogri, Konkani, and Sanskrit. In each language, the first row displays the input word level image, the second row (text in blue) represents the ground truth, and the third row shows the predicted text. Correct predictions are highlighted in green, while incorrect predictions are highlighted in red.

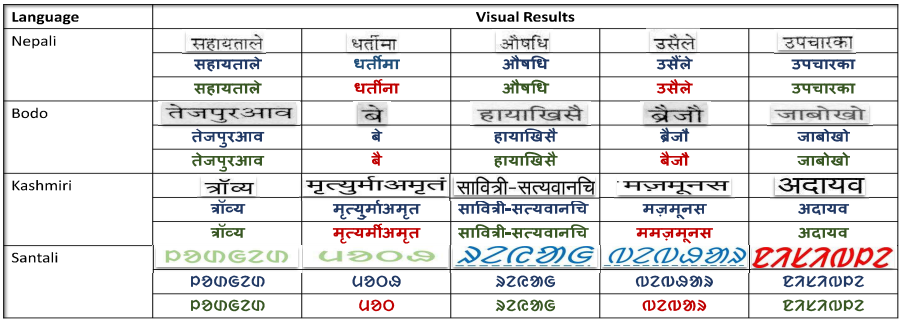

Fig. 11. Shows visual results on word level images for Nepali, Bodo, Kashmiri, and Santali. In each language, the first row displays the input word level image, the second row (text in blue) represents the ground truth, and the third row shows the predicted text. Correct predictions are highlighted in green, while incorrect predictions are highlighted in red.

where S indicates the number of substitutions, D indicates the number of deletions, I indicates the number of insertions, and N is the number of instances in reference text. In the case of *CER*, Eq. (1) operates on character level, and in the case of *WER*, Eq. (1) operates on word level. *Recognition Rate (RR)* is defined as

$$RR = (1 - ER)\,. \tag{2}$$

In the case of *CRR*, Eq. (2) operates on character level, and in the case of *WRR*, Eq. (2) operates on word level.

6 Result Analysis

6.1 Quantitative Results:

The quantitative results from synthetic and real datasets are summarized in Table 2. Across the eight languages using the Devanagari script — Maithili, Dogri, Bodo, Sindhi, Konkani, Sanskrit, Nepali, and Kashmiri — the Word Recognition Rate (WRR) varies. Maithili, Dogri, and Bodo exhibit WRRs exceeding 90%, with 92.74%, 91.85%, and 93.84%, respectively. Conversely, the remaining languages — Sindhi, Konkani, Sanskrit, Nepali, and Kashmiri — show WRRs ranging between 80-90%. Bodo stands out with the highest WRR, likely due to its larger training data volume in the dataset. This discrepancy suggests a correlation between dataset size and recognition accuracy, emphasizing the importance of sufficient training data for achieving optimal performance.

6.2 Visual Results:

Languages (Script)	Santali (Ol Chiki)	Maithili (Devanagari)	Sindhi (Devanagari)	Dogri (Devanagari)	Konkani (Devanagari)	Sanskrit (Devanagari)	Nepali (Devanagari)	Bodo (Devanagari)	Kashmiri (Devanagari)
Original Image									
ChatGPT 4o	ᱥᱚᱢᱵᱚᱫᱷᱤᱛ	संबंधित	हब्बिकनि.	डिज़ाइनरें,	काळंत	अर्थप्रकृतियः	औषधि	तेजपुरआव	सत्य
Proposed Method	ᱥᱚᱢᱵᱚᱫᱷᱤᱛ	संबंधित	हब्बिकनि	डिज़ाइनरें,	काळांत	अर्थप्रकतियो	औषधि	तेजपुरआव	सत्य

Fig. 12. Shows comparative examples for one sample for each language of how our proposed method performs against LLM like ChatGPT 4o. The first row displays the language and the corresponding script, the second row shows the input word level images, the third row shows the output ChatGPT 4o, and the last row shows the output of our proposed method. Correct predictions are highlighted in black, while incorrect predictions are highlighted in red.

The visual outcomes depicted in Fig. 10 and Fig. 11 showcase the results generated by our proposed method on *Mozhi-LR(R)* and *Mozhi-LR(S)* datasets. Correctly predicted words are highlighted in green, while incorrect predictions are highlighted in red. We can see that our proposed approach also gives the correct output for conjugate characters in all languages. For Maithili, we can see from Fig. 10 that our recognition module has difficulty recognizing text images that are a little noisy and blurry. In Sanskrit, the model produces the wrong text output for complex characters with noise. Particularly in Santali, our approach encounters difficulty recognizing the character across various input image qualities. In Kashmiri, our model is struggling to recognize some additional special characters as we show in the last column of Fig. 12.

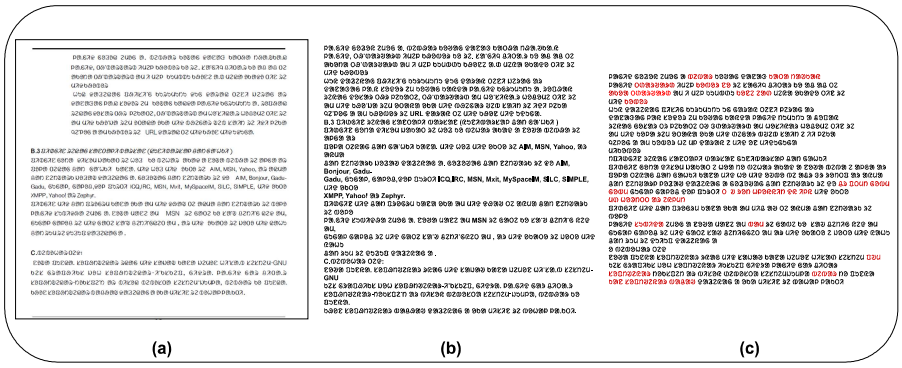

Fig. 13. Shows the result of result of page level API performance of our model for Ol Chiki script. fig (a) is the original page image, fig(b) is the ground truth of the page image, and fig (c) shows the result of the API.

7 Web-based Tool for End-to-End OCR Evaluation

In page level OCR, the objective is to transcribe the text within a document image by segmenting it into words and then recognizing the text at the word level. We focus only on text recognition, excluding layout analysis and reading order identification. To build an end-to-end page OCR pipeline, we combine existing text detection methods with our baseline model for recognition. Transcriptions from individual segments are arranged in the detected reading order. We develop a web-based tool[2] for this purpose. Users can upload document pages in Indic low-resource languages in the tool and get recognized text outputs. Fig. 13 depicts visual results at the page level using only our approach. Since existing OCR tools like current Tesseract and GoogleOCR do not support Ol Chiki script for Santali. Panel (a) presents the original document page image, while panel (b) displays the ground truth, and panel (c) shows the predicted text by our approach. Wrongly recognized texts are highlighted in red. This figure emphasizes that our approach is sufficiently good to recognize text in the Ol Chiki script.

8 Conclusions

This study introduces the *Mozhi-LR(R)* and *Mozhi-LR(S)* datasets, incorporate nine extremely low-resource Indian languages: *Maithili, Sindhi, Dogri, Konkani, Nepali, Sanskrit, Bodo, and Santali*, alongside high-performing OCR models. While the Devanagari script is employed for eight languages, Santali uses the Ol Chiki script. Additionally, we provide APIs for our page level OCR models and integrate them into the web-based tool for digitizing Indic low-resource printed documents. Our study, datasets, and accessible APIs are expected to foster research on OCR of Indian low-resource languages.

[2] `https://ilocr.iiit.ac.in/accurateocr/`

References

1. Census 2011. https://censusindia.gov.in/2011-Common/CensusData2011.Html, accessed on 1 November 2021
2. Antani, S., Agnihotri, L.: Gujarati character recognition. In: ICDAR (1999)
3. Aparna, K., Ramakrishnan, A.: A complete tamil optical character recognition system. In: DAS (2002)
4. Arya, D., Jawahar, C., Bhagvati, C., Patnaik, T., Chaudhuri, B., Lehal, G., Chaudhury, S., Ramakrishna, A.: Experiences of integration and performance testing of multilingual ocr for printed indian scripts. In: joint workshop on multilingual OCR and analytics for noisy unstructured text data (2011)
5. Ashwin, T., Sastry, P.: A font and size-independent ocr system for printed kannada documents using support vector machines. Sadhana **27**, 35–58 (2002)
6. Avadesh, M., Goyal, N.: Optical character recognition for sanskrit using convolution neural networks. In: DAS (2018)
7. Bashir, M., Goyal, V., Giri, K.J.: Challenges in recognition of kashmiri script. In: Recent Innovations in Computing: Proceedings of ICRIC (2022)
8. Chaudhuri, B.B., Pal, U.: A complete printed bangla ocr system. Pattern recognition **31**(5), 531–549 (1998)
9. Chavan, V., Malage, A., Mehrotra, K., Gupta, M.K.: Printed text recognition using blstm and mdlstm for indian languages. In: ICIIP (2017)
10. Cheng, Z., Bai, F., Xu, Y., Zheng, G., Pu, S., Zhou, S.: Focusing attention: Towards accurate text recognition in natural images. In: ICCV (2017)
11. Dwivedi, A., Saluja, R., Sarvadevabhatla, R.K.: An ocr for classical indic documents containing arbitrarily long words. In: CVPRW (2020)
12. Gongidi, S., Jawahar, C.: IIIT-INDIC-HW-Words: A dataset for indic handwritten text recognition. In: ICDAR. pp. 444–459 (2021)
13. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: ICML (2006)
14. Graves, A., Fernández, S., Schmidhuber, J.: Multi-dimensional recurrent neural networks. In: ICANN. pp. 549–558 (2007)
15. Hakro, D.N., Talib, A.Z.: Printed text image database for sindhi ocr. TALLIP **15**(4), 1–18 (2016)
16. Hasan, S., Dhakal, A., Mehedi, M.H.K., Rasel, A.A.: Optical text recognition in nepali and bengali: A transformer-based approach. arXiv preprint arXiv:2404.02375 (2024)
17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
18. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Synthetic data and artificial neural networks for natural scene text recognition. arXiv preprint arXiv:1406.2227 (2014)
19. Jaderberg, M., Simonyan, K., Vedaldi, A., Zisserman, A.: Reading text in the wild with convolutional neural networks. IJCV **116**, 1–20 (2016)
20. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. NeurIPS (2015)
21. Jain, M., Mathew, M., Jawahar, C.: Unconstrained ocr for urdu using deep cnn-rnn hybrid networks. In: ACPR (2017)
22. Jawahar, C., Kumar, M.P., Kiran, S.R.: A bilingual ocr for hindi-telugu documents and its applications. In: ICDAR (2003)

23. Krishnan, P., Sankaran, N., Singh, A.K., Jawahar, C.: Towards a robust ocr system for indic scripts. In: DAS (2014)
24. Kundaikar, T., Pawar, J.D.: Multi-font devanagari text recognition using lstm neural networks. In: ICTSCI (2020)
25. Lehal, G.S., Singh, C.: A gurmukhi script recognition system. In: ICPR (2000)
26. Liu, X., Meng, G., Pan, C.: Scene text detection and recognition with advances in deep learning: a survey. IJDAR **22**, 143–162 (2019)
27. Mathew, M., Jain, M., Jawahar, C.: Benchmarking scene text recognition in devanagari, telugu and malayalam. In: ICDAR (2017)
28. Mathew, M., Singh, A.K., Jawahar, C.: Multilingual ocr for indic scripts. In: DAS (2016)
29. Natarajan, P.S., MacRostie, E., Decerbo, M.: The bbn byblos hindi ocr system. In: Document Recognition and Retrieval XII. vol. 5676, pp. 10–16 (2005)
30. Naveen Sankaran, T., Neelappa, A., Jawahar, C.: Devanagari text recognition: A transcription based formulation. In: ICDAR (2013)
31. Neeba, N., Jawahar, C.: Empirical evaluation of character classification schemes. In: ICAPR (2009)
32. Negi, A., Bhagvati, C., Krishna, B.: An ocr system for telugu. In: ICDAR (2001)
33. Pal, U., Sarkar, A.: Recognition of printed urdu script. In: ICDAR (2003)
34. Pal, U., Chaudhuri, B.: Indian script character recognition: a survey. pattern Recognition **37**(9), 1887–1899 (2004)
35. Pant, N., Bal, B.K.: Improving nepali ocr performance by using hybrid recognition approaches. In: IISA (2016)
36. Paul, D., Chaudhuri, B.B.: A blstm network for printed bengali ocr system with high accuracy. arXiv preprint arXiv:1908.08674 (2019)
37. Sanjeev Kunte, R., Sudhaker Samuel, R.: A simple and efficient optical character recognition system for basic symbols in printed kannada text. Sadhana **32**(5), 521–533 (2007)
38. Sankaran, N., Jawahar, C.: Recognition of printed devanagari text using blstm neural network. In: ICPR (2012)
39. Shah, R., Gupta, M.K., Kumar, A.: Ancient sanskrit line-level ocr using opennmt architecture. In: ICIIP (2021)
40. Sinha, R., Mahabala, H.: Machine recognition of devanagari script. IEEE Trans. on SMC **9**(8), 435–441 (1979)
41. Ul-Hasan, A., Ahmed, S.B., Rashid, F., Shafait, F., Breuel, T.M.: Offline printed urdu nastaleeq script recognition with bidirectional lstm networks. In: ICDAR (2013)
42. Vijay Kumar, B., Ramakrishnan, A.: Machine recognition of printed kannada text. In: DAS (2002)