Enhancing Accuracy in Indic Handwritten Text Recognition

Evani Lalitha, Ajoy Mondal, and C. V. Jawahar

CVIT, International Institute of Information Technology, Hyderabad, India
lalitha.e@research.iiit.ac.in
{ajoy.mondal,jawahar}@iiit.ac.in

Abstract. Handwritten Text Recognition (HTR) presents a significant challenge in computer vision due to various factors such as individual writing styles, noise, blur, and other imperfections in the text. This challenge is further exacerbated when dealing with languages using Indian scripts, which are characterized by complex character structures, extensive character inventories, and specific cultural nuances. In this study, we address these challenges by focusing on enhancing handwritten text recognition for ten Indic languages: *Hindi, Bengali, Telugu, Tamil, Gujarati, Gurumukhi, Oriya, Kannada, Malayalam, and Urdu*. We aim to improve recognition accuracy by leveraging the Permuted Autoregressive Sequence Model (PARSeq), an extension of the transformer-based model. Our results demonstrate the superiority of the PARSeq model over existing approaches, particularly in achieving state-of-the-art performance across most languages. Additionally, we investigate the efficacy of transfer learning from printed text to handwritten text, revealing its potential to enhance recognition performance. The trained models and code are publicly available at https://github.com/LalithaEvani/Indic-HTR-CVIP-2024.

Keywords: Indic handwritten text · transfer learning · recognizer · transformer · pre-train · PARSeq.

1 Introduction

Handwritten Text Recognition (HTR) is a sub-domain in Optical Character Recognition (OCR), which focuses on automatically converting handwritten text into digital text, mimicking the capabilities of human readers. Recognizing Handwritten text is one of the important and challenging problems in computer vision. One of the main challenges of HTR is the uniqueness of the writing. In most cases, text written by one writer is unique to the others. Hence, the ability to recognize the same text written by different writers is complex, and it is even utilized in forensic handwriting analysis to identify the person based on the handwriting. The uniqueness in handwriting style is also used in graphology which attempts to assess a person's personality through handwriting. One such application is

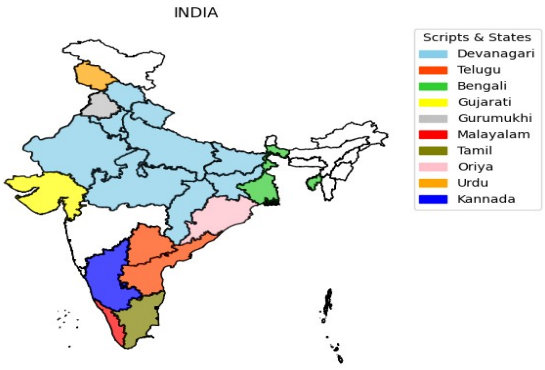

Fig. 1. India map representing the ten scripts dominantly used in several states.

to identify the profession of a person via handwriting [15]. This characteristic makes developing models that generalize well across different styles challenging. Other challenges include noise, blur, smudges, incomplete characters, variation in the density of the ink, the orientation of characters, and scaling of characters.

There have been numerous attempts at recognizing handwritten text, and with the evolution of deep learning, the accuracy of the recognizers drastically increased [9, 7, 1]. Language plays a crucial role in handwriting recognition, as recognizers are typically designed to be language-specific. Therefore, a recognizer must be trained specifically for the language it is intended to recognize. Generating handwritten data manually imposes limitations on the data collection process, affecting the ability to enhance accuracy. In modern deep learning models, selecting suitable models is as critical as gathering sufficient data.

Handwritten text recognizers are primarily seen in prevalent languages such as English [12, 23, 17], Chinese [30, 29, 22], Arabic [19, 13], and Japanese [18, 21]. The ability to build recognizers on languages other than these is essential because, if not, thousands of languages spoken worldwide are at risk of slowly becoming extinct. Indic languages, prevalent in the Indian subcontinent, originate mainly from the Brahmi script and encompass hundreds of dialects. The reliance on training data hinders building effective recognizers for these languages; thus, the scarcity of extensive datasets is a significant drawback. Furthermore, research and development efforts focusing on Indic languages are relatively limited. These languages present unique complexities compared to scripts like Latin, including intricate character structures and a more extensive character inventory, intensifying recognition challenges. Handling diacritics and ligatures, variations in writing styles, and cultural nuances further complicate recognizer development, making achieving accurate results daunting.

Building upon prior research in handwritten text recognition for Indic languages, this paper focuses on enhancing recognizers for ten of the 22 major languages recognized under the "8th schedule" of the Indian constitution. These languages include *Hindi, Bengali, Telugu, Tamil, Gujarati, Gurumukhi, Oriya, Kannada, Malayalam, and Urdu*. Fig. 1 illustrates the states where these scripts are predominantly used. This paper aims to enhance the accuracy of handwritten text recognition in Indic languages using a transformer-based model extension known as the Permuted Auto-regressive Sequence Model (PARSeq), proposed by Darwin *et al.* [4]. The PARSeq models are trained on handwritten text data in Indic languages and compared with previous results, highlighting the novelty of the transformer-based approach compared to CNN and RNN-based methods utilized in previous studies.

We summarise contributions as follows:

- Implement a transformer-based model using PARSeq [4] to enhance recognition accuracy for Indic handwritten text.
- Showcase the models state-of-the-art performance across most languages by comparing our method with prior approaches. (refer Table 2).
- Highlighting the effectiveness of transfer learning by investigating its impact from printed to handwritten text. (refer Table 4).

2 Related Work

In the context of Indic scripts, handwritten text recognition typically employs three main methods. The first method involves segmentation, where characters within a word image are segmented, and individual characters are recognized using isolated symbol classifiers like Support Vector Machine (SVM) [3]or Artificial Neural Network (ANN) [16, 2]. For instance, Roy *et al.* [24] segmented Indic language word images into three zones (lower, middle, and upper) with the utilization of morphological operations, shape matching and other such image processing techniques. Support Vector Machines were then applied to recognize the upper and lower zones, while Hidden Markov Models were employed for the middle zone. At last, the results that were obtained from three zones were combined.

The second method is developing recognizers using methods that are segmentation-free, which focus on recognizing the entire word or obtaining a holistic representation [27, 26, 14]. For example, Shaw *et al.* [27] used histogram of chain-code directions to extract features from word images, by by scanning the image strips from left to right using a sliding window. To recognize Devanagari handwritten words, a continuous density Hidden Markov Model (HMM) was proposed. Another study done by Shaw *et al.* [26], introduced a novel approach for holistic recognition of offline handwritten word images by combining two feature vectors. However, these methods are limited in their ability to recognize a diverse lexicon due to their reliance on predefined lexicons.

The third category of methods involves sequence modeling, where the handwritten text recognition task is transformed into a sequence-to-sequence prediction task, where both the input and output are treated as sequences of vectors. This approach is commonly addressed using Recurrent Neural Networks (RNN) [11, 25]. Sequence-to-sequence models are designed to optimize the likelihood of generating the output label based on the input feature sequence [1, 7, 8]. It overcomes the limitations of previous methods by eliminating the need for explicit symbol segmentation and allowing for the recognition of variable-length lexicons. For example, Garain *et al.* [9] proposed a recognizer that uses Bidirectional Long-Short-Term Memory (BLSTM) with a Connectionist Temporal Classification (CTC) layer to recognize Bengali handwritten words without constraints offline. Adak *et al.* [1] developed a Bengali handwritten text recognizer using a Convolutional Neural Network (CNN) integrated with LSTM and a CTC layer. Dutta *et al.* [7, 8] constructed handwritten text recognizers for Devanagari, Bengali, and Telugu languages using a CNN-RNN hybrid model trained end-to-end. Santoshini *et al.* [10] built a recognizer consisting of a Spatial Tranformer Network in addiction to the hybrid CNN-RNN and CTC layer for eight Indic scripts, including Urdu, Bengali, Tamil, Gujarati, Malayalam, Gurumukhi, Kannada, and Odia. Additionally, various data augmentations were employed to enhance the recognizer's accuracy.

As deep learning continues to evolve, the adoption of transformer architectures [28], has become increasingly prominent which was initially designed for natural language processing (NLP) tasks. Unlike previous methods, these architectures capture relationships among various elements of a sequence by leveraging self-attention mechanisms. As a result of transformers being a success in NLP, they were extended into the vision domain, resulting in the development of Vision Transformers [5]. These models eliminate the need for CNNs and RNNs by segmenting the images into a sequence of patches, which are then processed by the transformer. One notable extension of the transformer architecture is PARSeq [4], initially developed for Scene Text Recognition (STR). This paper uses the PARSeq model to enhance handwritten text recognizers for Indic languages.

3 Methodology

In this study, we employ PARSeq, a Permuted auto-regressive Sequence Model based on transformer architecture. Initially designed for Scene Text Recognition, PARSeq is adept at recognizing text from cropped regions within a scene. Leveraging its permutation capability and transformer extension, we apply PARSeq to recognize handwritten text in Indic languages.

PARSeq comprises an encoder and a decoder, the encoder comprising 12 layers and the decoder featuring a single layer, the architecture of which is shown in Fig. 2. This configuration is chosen for computational efficiency. The Encoder is ViT Encoder. ViT implements transformers on images. But in PARSeq all the output tokens z of the encoder are given as input to the decoder. Here x is the

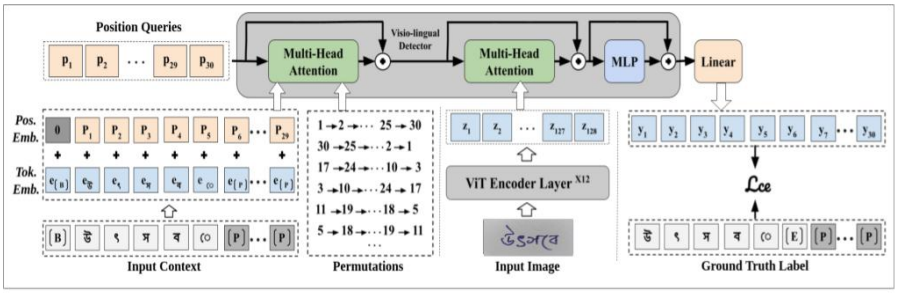

Fig. 2. PARSeq architecture. [P], [B], and [E], are padding tokens, *beginning-of-sequence*(BOS), and *end-of-sequence* (EOS), respectively. $\mathcal{L}_{ce}$ is the cross-entropy loss.

input image, H and W are height and width of the image, respectively, divided into $p_w \times p_h$ patches, while d_{model} being the dimension.

$$\mathbf{z} = \text{Enc}(x) \in \mathbb{R}^{\frac{WH}{p_w p_h} \times d_{\text{model}}} \tag{1}$$

There are two MHA modules, they are used for context-position attention and image-position attention respectively. The context-position attention is given by:

$$\mathbf{h_c} = \mathbf{p} + \text{MHA}(\mathbf{p}, \mathbf{c}, \mathbf{c}, \mathbf{m}) \in \mathbb{R}^{(T+1) \times d_{\text{model}}}, \tag{2}$$

where $\mathbf{p} \in \mathbb{R}^{(T+1) \times d_{\text{model}}}$ are the position tokens, T is the context length, $\mathbf{c} \in \mathbb{R}^{(T+1) \times d_{\text{model}}}$ are the context embeddings with positional information, and $\mathbf{m} \in \mathbb{R}^{(T+1) \times (T+1)}$ is the optional attention mask.

The image-position attention is given by:

$$\mathbf{h_i} = \mathbf{h_c} + \text{MHA}(\mathbf{h_c}, \mathbf{z}, \mathbf{z}) \in \mathbb{R}^{(T+1) \times d_{\text{model}}} \tag{3}$$

The output logits are given by:

$$\mathbf{y} = \text{Linear}(\mathbf{h}_{\text{dec}}) \in \mathbb{R}^{(T+1) \times (S+1)}, \tag{4}$$

where S is the size of the character set used for training, and h_{dec} is the last decoder hidden state.

The decoder function is given by:

$$\mathbf{y} = \text{Dec}(\mathbf{z}, \mathbf{p}, \mathbf{c}, \mathbf{m}) \in \mathbb{R}^{(T+1) \times (S+1)}. \tag{5}$$

The main feature of PARSeq is permutation language modeling, which trains the model on all T factorization of the likelihood, where T is the number of tokens in the output sequence. Considering the standard Vision Transformers, it is nothing but a particular case of PLM where one of the permutations $[1, 2, \ldots, T]$ is used. It can be stated as:

$$\log p(\mathbf{y}|\mathbf{x}) = \mathbb{E}\mathbf{z} \sim \mathcal{Z}_T \left[\sum_{t=1}^{T} \log p_\theta(y_{z_t}|\mathbf{y}_{\mathbf{z}<t}, \mathbf{x}) \right]. \tag{6}$$

Due to computational requirements, the model is only trained on some T factorization but K of the T permutations. This K is chosen so that the first half of the permutations are left-to-right randomly sampled permutations, and the other half are right-to-left permutations, which is the flipped version of the former. The loss thus calculated is given by:

$$\mathcal{L} = \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}_{\text{ce}}(\mathbf{y}_k, \hat{\mathbf{y}}), \tag{7}$$

where $\hat{\mathbf{y}}$ is the ground truth label and $\mathbf{y}_k = \text{Dec}(\mathbf{z}, \mathbf{p}, \mathbf{c}, \mathbf{m_k})$

As mentioned in the previous section, the previous works focussed on CNN based models to improve the accuracy of the recognizers. Transformer based approaches are latest advancements in the deep learning domain. PARSeq is especially chosen because of its permutation capabilities. It is capable of context-free and context-aware decoding, and iterative refinement. It combines various decoding schemes into a single model and leverages the parallel computation capabilities of Transformers. It also uses attention extensively, demonstrating the robustness on vertical and rotated text in images[1].

4 Experiments and Results

4.1 Implementation Details

All PARSeq models were trained on four Nvidia GeForce GTX 1080 Ti GPUs for approximately 160,000 iterations, employing a batch size of 254. Pre-training utilized a 1-cycle learning rate scheduler, while training employed the SWA scheduler with the Adam optimizer. Consistent with the original PARSeq model, a permutation count of K=6, a patch size of 8×4, a drop out rate of 0.1 and a learning rate of 7e-4, were employed for the entire pre-training and most of the training. For some of the languages, namely, Bengali, Gujarati, Kannada, Oriya, and Malayalam fine-tuning was done at a permutation count of K=14, dropout rate of 0.4 and a learning rate of 7e-6. The maximum label length across all languages was set to 35, chosen based on the dataset to accommodate significant words during future inference. The character set included language-specific characters, special symbols, and digits, with the number of characters used for training varying depending on the language.

4.2 Training/Testing Details

Training is conducted through two approaches. The first approach involves training all languages on the *IIIT-INDIC-HW-WORDS* dataset, utilizing the 1-cycle

[1] For more information on PARSeq, please refer [4].

learning rate scheduler during initial training and the SWA scheduler a few iterations before training completion. The second approach employs transfer learning, where the model undergoes initial pre-training on a printed dataset for all languages, followed by training on the *IIIT-INDIC-HW-WORDS* dataset. Three optimal model checkpoints, determined by achieving the minimum validation loss, were saved and tested on the test set of *IIIT-INDIC-HW-WORDS* dataset across all languages.

4.3 Dataset

We used *IIIT-INDIC-HW-WORDS* dataset for experimental purposes. It includes handwritten word level images of ten languages — *Hindi, Bengali, Telugu, Tamil, Kannada, Gurumukhi, Gujarati, Oriya, Malayalam and Urdu.* Table 1 shows the statistics of this dataset, and Fig. 3 shows a few sample word level images from this dataset.

Table 1. Shows the statistics of *IIIT-INDIC-HW-WORDS* dataset used in this experiment.

Script	#Writers	#Word Instances	Lexicon Size	#Train Instances	#Val Instances	#Test Instances
Devanagari	12	95K	11,030	69,853	12,708	12,869
Telugu	11	120K	12,945	80,637	19,980	17,898
Bengali	24	113K	11,295	82,554	12,947	17,574
Gujarati	17	116K	10,963	82,563	17,643	16,490
Gurumukhi	22	112K	11,093	81,042	13,627	17,947
Kannada	11	103K	11,766	73,517	13,752	15,730
Odia	10	101K	13,314	73,400	11,217	16,850
Malayalam	27	116K	13,401	85,270	11,878	19,635
Tamil	16	103K	13,292	75,736	11,597	16,184
Urdu	8	100K	11,936	71,207	13,906	15,517

4.4 Evaluation Metrics

The performance of the model is assessed with the help of Word Error Rate (WER), which calculates the ratio of incorrectly classified words to the total number of words. A word is considered correct if all its characters are predicted accurately; otherwise, it is deemed incorrect. We also used Character Error Rate (CER), which calculates the error at the character level. In general, Error Rate (ER) is the ratio of the total number of errors to the total number of predictions.

$$\mathrm{ER} = \frac{\mathrm{S} + \mathrm{D} + \mathrm{I}}{\mathrm{N}} \tag{8}$$

, where S indicates the number of substitutions, D indicates the number of deletions, I indicates the number of insertions and N the number of instances in

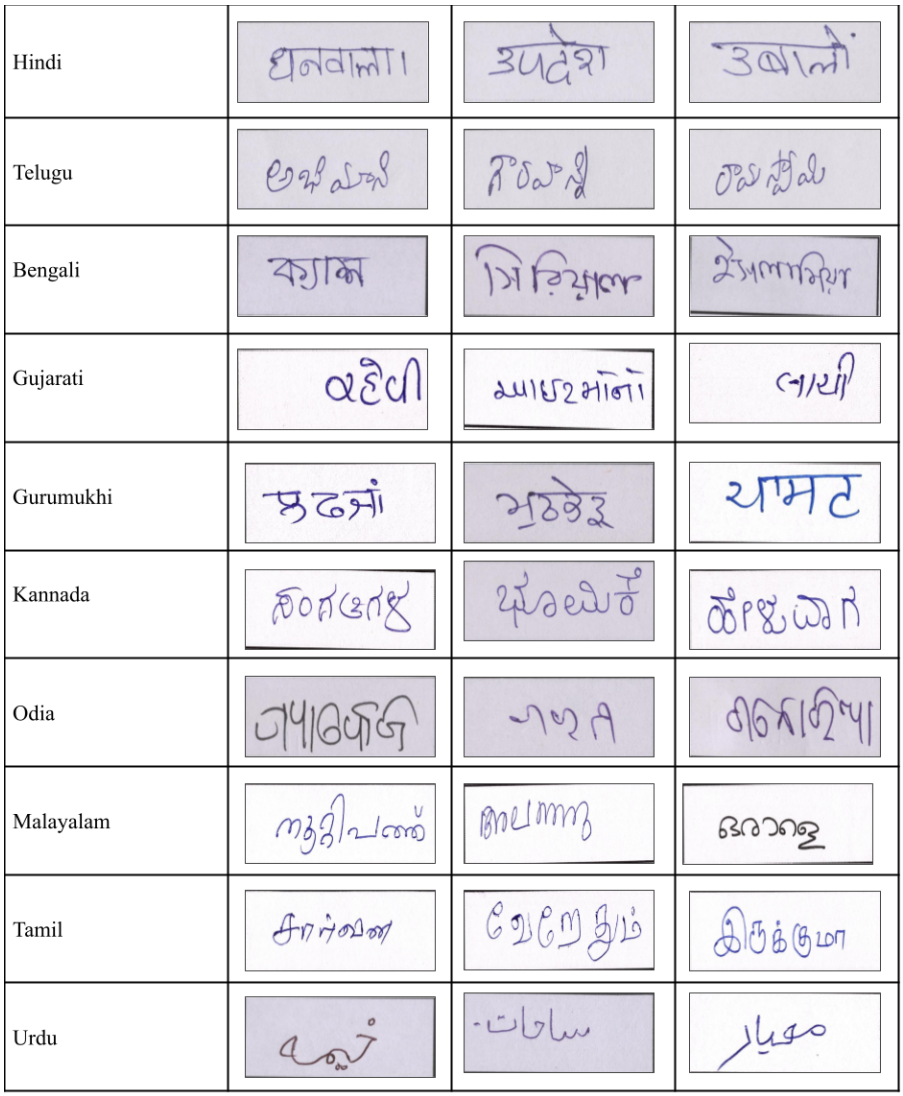

Fig. 3. Some sample word images from the dataset used in this experiment.

reference text. In the case of CER, the Eq. (8) is applied at the character level, while in the case of WER, it is used on the word level.

Table 2. Presents WER and CER comparisons across different languages. ↓ denotes that better performance is represented by a smaller value.

Language	CRNN [10, 6]		Mondal *et al.* [20]		Ours	
	WER↓	**CER↓**	**WER↓**	**CER↓**	**WER↓**	**CER↓**
Hindi	26.22	3.17	9.06	**1.98**	**6.93**	2.93
Telugu	23.98	3.18	12.11	**2.15**	**10.37**	2.50
Bengali	15.71	4.85	**12.34**	**2.35**	16.35	4.17
Gujarati	18.59	2.39	**9.21**	**1.19**	12.04	2.74
Gurumukhi	18.37	3.42	**10.77**	**2.1**	11.92	3.24
Kannada	7.65	1.79	8.57	**1.01**	**6.55**	1.13
Odia	19.19	3	**12.32**	**1.32**	14.86	3.38
Malayalam	10.23	1.92	9.37	1.12	**5.97**	**0.98**
Tamil	**7.82**	1.92	9.18	**1.25**	8.02	1.43
Urdu	24.11	5.07	18.76	**3.89**	**17.81**	5.51

4.5 Results Analysis

Quantitative Results: We compare the results obtained using PARSeq with previous works [10, 6, 20], as presented in Table 2. When contrasting with Dutta *et al.* [6], it's evident that for Hindi, our achieved WER is substantially lower at 6.93 compared to previous 26.22, indicating significant improvement without the use of a lexicon. Similarly, for Telugu, our WER of 10.37 surpasses the previous 23.98. In comparison with Santoshini *et al.* [10], for the remaining eight languages, our results show higher WER values in Gurumukhi, Gujarati, and Urdu, with differences exceeding 6% from the previous WER. Additionally, our WER is higher by more than 4% for Malayalam and Odia, and more then 1% for Kannada. However, there is minimal change in WER for Tamil and Bengali. Overall, as the WER decreases compared to [10, 6], it suggests that transformer based models outperform CNN-RNN based models. In the case of CER, when compared with Dutta *et al.* [6] and Santoshini *et al.* [10], it can be seen that our CER is lower for most of the languages with the lowest being in Malayalam and Tamil.

Furthermore, comparing the results with Mondal *et al.* [20], the PARSeq model demonstrates satisfactory performance in Hindi, Telugu, Malayalam, Kannada, Tamil, and Urdu. However, for Gurumukhi, the WER is slightly higher for PARSeq. Significant differences in WER are observed in the Bengali, Gujarati, and Odia languages. In the case of CER, our CER is higher for most of the languages or similar otherwise. One of the reasons for a low CER in [20] could be that, there is an additional module, called semantic module, which predicts the semantic information which is then given as additional input to the decoder, thus leveraging the accuracy of predicting the characters.

Fig. 4. Shows selected samples showcasing qualitative results obtained using PARSeq across ten Indic languages. Text highlighted in blue refers to the Ground truth and text highlighted in red refers to text recognized incorreclty.

Fig. 5. Shows qualitative comparison between CNN-RNN [20] model and PARSeq model across ten Indic languages. Text highlighted in blue refers to the Ground truth and text highlighted in red refers to text recognized incorreclty.

Qualitative Analysis: Fig. 4 depicts the qualitative outcomes achieved through PARSeq across the ten Indic languages. The accurate predictions are displayed in the initial three columns, while the subsequent two columns showcase incorrect predictions. Wrongly recognized characters within the words are highlighted in red for clarity.

Another comparison is made between the CNN-RNN model [10, 6, 20] and PARSeq model across the ten Indic languages as shown in Fig. 5. The first column contains words that are correctly predicted by both the CNN-RNN model and the PARSeq model; the second column includes words that the CNN-RNN model wrongly predicts and correctly predicted by the PARSeq model, and the last column contains the words that both models wrongly predict. PARSeq could predict words across languages that the CNN-RNN model could not predict correctly. In the case of wrong predictions, both the models wrongly predicted almost the same character across languages.

Table 3. Presents post-OCR results across ten Indic languages. ↓ denotes that better performance is represented by a smaller value.

Language	**Recall 1↓**		**Recall 2↓**		**Recall 3↓**		**Recall 4↓**		**Recall 5↓**	
	WER	**CER**	**WER**	**CER**	**WER**	**CER**	**WER**	**CER**	**WER**	**CER**
Hindi	4.14	2.03	3.06	1.59	2.52	1.34	2.16	1.15	1.93	1.05
Telugu	2.84	1.54	1.68	1.05	1.31	0.86	1.13	0.76	1.04	0.07
Bengali	7.77	3.66	5.6	2.8	4.6	2.32	4.06	2.08	3.68	1.83
Gujarati	4.41	1.92	2.52	1.26	1.87	0.97	1.55	0.83	1.38	0.74
Gurumukhi	6.76	2.87	4.07	2.06	3.61	1.6	3.04	1.38	2.67	1.21
Kannada	1.64	0.61	0.87	0.41	0.71	0.33	0.49	0.25	0.39	0.21
Odia	6.08	2.62	3.61	1.78	2.72	1.43	2.32	1.26	2.08	1.34
Malayalam	1.29	0.54	0.71	0.36	0.53	0.29	0.44	0.24	0.35	0.2
Tamil	1.6	0.78	0.96	0.54	0.8	0.47	0.66	0.41	0.56	0.37
Urdu	14.75	5.98	11.98	4.79	10.34	4.12	9.04	3.64	8.01	3.29

Post-OCR Error Correction: As the name suggests, Post-OCR Error Correction is implemented after obtaining predictions from the model. Errors in these predictions are identified and corrected using several methods. A lexicon is created by concatenating the training, validation, and test datasets. The edit distance between the predicted words and the lexicon is then calculated to identify the top five words with the least edit distances, which are considered potentially correct words. For Recall 1, the final word chosen is the word which has the least edit distance, and the CER (Character Error Rate) and WER (Word Error Rate) are calculated. For Recall 2, the top two words are considered potential correct words, and the one with the least edit distance to the ground truth is used to calculate CER and WER. This process is repeated for Recall 3, Recall 4, and Recall 5. Our paper applies this post-OCR error correction method to all

ten languages considered, with the CER and WER results presented in Table 3. For each language, CER and WER are calculated up to Recall 5.

Table 4. Highlights the impact of transfer learning on the performance of the recognizer. Bold value indicates the best results.

Language	**Word Error Rate (WER)↓**	
	Trained model	(Pre-trained + Trained) model
Bengali	23.69	**19.08**
Gujarati	17.69	**12.32**
Gurumukhi	14.63	**11.92**
Kannada	11.59	**7.28**
Odia	21.03	**16.78**
Malayalam	10.49	**5.97**
Tamil	9.82	**8.02**

Ablation Study: An experiment was conducted to assess the impact of transfer learning on model training. In this experiment, the PARSeq model underwent training in two distinct approaches, both with the same hyperparameters. Initially, it was trained on handwritten data from the *IIIT-INDIC-HW-WORDS* dataset. At the same time, the latter method involved applying transfer learning by pre-training the model on printed data before training it on handwritten data. Table 4 illustrates the comparison of results obtained for select languages. For Bengali, Gujarati, Kannada, Odia and Malayalam the WER achieved by the pre-trained model is significantly lower than that of the model without pre-training, with the difference exceeding 4%. Similarly, for Gurumukhi, the WER is reduced by 3% and for Tamil, it reduced by 2%. Thus, it can be inferred that leveraging learned representations from printed data as a starting point for training on handwritten data can substantially enhance the recognizer's performance.

5 Conclusions

This study focuses on enhancing handwritten text recognition by employing the Permutated Autoregressive Sequence Model (PARSeq), an extension of transformer-based models. Trained on the *IIIT-INDIC-HW-WORDS* dataset, PARSeq produces models for ten major Indic languages. Comparative analysis with existing approaches demonstrates state-of-the-art performance across most languages. Post-OCR error correction showcases the advantage of using lexicon in correcting the predicted words, which can significantly improve the accuracy across all languages. Additionally, the study investigates the impact of transfer learning by pre-training models on printed data before training them on handwritten data. The findings suggest that models trained with transfer learning exhibit superior performance compared to those trained solely on handwritten data.

Acknowledgments

This work is supported by MeitY, Government of India, through the NLTM-Bhashini project.

References

1. Adak, C., Chaudhuri, B.B., Blumenstein, M.: Offline cursive bengali word recognition using cnns with a recurrent model. In: 2016 15th International conference on frontiers in handwriting recognition (ICFHR). pp. 429–434. IEEE (2016)
2. Alonso-Weber, J.M., Sesmero, M., Sanchis, A.: Combining additive input noise annealing and pattern transformations for improved handwritten character recognition. Expert systems with applications **41**(18), 8180–8188 (2014)
3. Arora, S., Bhattacharjee, D., Nasipuri, M., Malik, L., Kundu, M., Basu, D.K.: Performance comparison of svm and ann for handwritten devnagari character recognition. arXiv preprint arXiv:1006.5902 (2010)
4. Bautista, D., Atienza, R.: Scene text recognition with permuted autoregressive sequence models. In: European conference on computer vision. pp. 178–196. Springer (2022)
5. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
6. Dutta, K., Krishnan, P., Mathew, M., Jawahar, C.: Offline handwriting recognition on devanagari using a new benchmark dataset. In: 2018 13th IAPR international workshop on document analysis systems (DAS). pp. 25–30. IEEE (2018)
7. Dutta, K., Krishnan, P., Mathew, M., Jawahar, C.: Towards accurate handwritten word recognition for hindi and bangla. In: Computer Vision, Pattern Recognition, Image Processing, and Graphics: 6th National Conference, NCVPRIPG 2017, Mandi, India, December 16-19, 2017, Revised Selected Papers 6. pp. 470–480. Springer (2018)
8. Dutta, K., Krishnan, P., Mathew, M., Jawahar, C.: Towards spotting and recognition of handwritten words in indic scripts. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 32–37. IEEE (2018)
9. Garain, U., Mioulet, L., Chaudhuri, B.B., Chatelain, C., Paquet, T.: Unconstrained bengali handwriting recognition with recurrent models. In: 2015 13th international conference on document analysis and recognition (ICDAR). pp. 1056–1060. IEEE (2015)
10. Gongidi, S., Jawahar, C.: iiit-indic-hw-words: A dataset for indic handwritten text recognition. In: Document Analysis and Recognition–ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part IV 16. pp. 444–459. Springer (2021)
11. Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A novel connectionist system for unconstrained handwriting recognition. IEEE transactions on pattern analysis and machine intelligence **31**(5), 855–868 (2008)
12. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. Advances in neural information processing systems **21** (2008)

13. Jemni, S.K., Ammar, S., Kessentini, Y.: Domain and writer adaptation of offline arabic handwriting recognition using deep neural networks. Neural Computing and Applications **34**(3), 2055–2071 (2022)
14. Kaur, H., Kumar, M.: On the recognition of offline handwritten word using holistic approach and adaboost methodology. Multimedia Tools and Applications **80**(7), 11155–11175 (2021)
15. Kumar, P., Gupta, M., Gupta, M., Sharma, A.: Profession identification using handwritten text images. In: Computer Vision and Image Processing: 4th International Conference, CVIP 2019, Jaipur, India, September 27–29, 2019, Revised Selected Papers, Part II 4. pp. 25–35. Springer (2020)
16. Labani, M., Moradi, P., Ahmadizar, F., Jalili, M.: A novel multivariate filter method for feature selection in text classification problems. Engineering Applications of Artificial Intelligence **70**, 25–37 (2018)
17. Li, M., Lv, T., Chen, J., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., Wei, F.: Trocr: Transformer-based optical character recognition with pre-trained models. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 13094–13102 (2023)
18. Ly, N.T., Nguyen, C.T., Nakagawa, M.: Training an end-to-end model for offline handwritten japanese text recognition by generated synthetic patterns. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 74–79. IEEE (2018)
19. Maalej, R., Kherallah, M.: Improving the dblstm for on-line arabic handwriting recognition. Multimedia Tools and Applications **79**, 17969–17990 (2020)
20. Mondal, A., Jawahar, C.: Enhancing indic handwritten text recognition using global semantic information. In: International Conference on Frontiers in Handwriting Recognition. pp. 360–374. Springer (2022)
21. Nguyen, K.C., Nguyen, C.T., Nakagawa, M.: A semantic segmentation-based method for handwritten japanese text recognition. In: 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 127–132. IEEE (2020)
22. Peng, D., Jin, L., Ma, W., Xie, C., Zhang, H., Zhu, S., Li, J.: Recognition of handwritten chinese text by segmentation: a segment-annotation-free approach. IEEE Transactions on Multimedia (2022)
23. Pham, V., Bluche, T., Kermorvant, C., Louradour, J.: Dropout improves recurrent neural networks for handwriting recognition. In: 2014 14th international conference on frontiers in handwriting recognition. pp. 285–290. IEEE (2014)
24. Roy, P.P., Bhunia, A.K., Das, A., Dey, P., Pal, U.: Hmm-based indic handwritten word recognition using zone segmentation. Pattern recognition **60**, 1057–1075 (2016)
25. Sankaran, N., Neelappa, A., Jawahar, C.: Devanagari text recognition: A transcription based formulation. In: 2013 12th International Conference on Document Analysis and Recognition. pp. 678–682. IEEE (2013)
26. Shaw, B., Bhattacharya, U., Parui, S.K.: Combination of features for efficient recognition of offline handwritten devanagari words. In: 2014 14th International conference on frontiers in handwriting recognition. pp. 240–245. IEEE (2014)
27. Shaw, B., Parui, S.K., Shridhar, M.: Offline handwritten devanagari word recognition: A holistic approach based on directional chain code feature and hmm. In: 2008 International Conference on Information Technology. pp. 203–208. IEEE (2008)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

29. Wu, Y.C., Yin, F., Chen, Z., Liu, C.L.: Handwritten chinese text recognition using separable multi-dimensional recurrent neural network. In: 2017 14th IAPR international conference on document analysis and recognition (ICDAR). vol. 1, pp. 79–84. IEEE (2017)
30. Xie, Z., Sun, Z., Jin, L., Feng, Z., Zhang, S.: Fully convolutional recurrent network for handwritten chinese text recognition. In: 2016 23rd International Conference on Pattern Recognition (ICPR). pp. 4011–4016. IEEE (2016)