ICDAR 2023 Competition on Indic Handwriting Text Recognition

Ajoy Mondal[0000−0002−4808−8860] and C. V. Jawahar[0000−0001−6767−7057]

International Institute of Information Technology, Hyderabad, India
ajoy.mondal@iiit.ac.in and jawahar@iiit.ac.in

Abstract. This paper presents the competition report on Indic Handwriting Text Recognition (IHTR) held at the 17th International Conference on Document Analysis and Recognition (ICDAR 2023 IHTR). Handwriting text recognition is an essential component for analyzing handwritten documents. Several good recognizers are available for English handwriting text in the literature. In the case of Indic languages, limited work is available due to several challenging factors. (i) Two or more characters are often combined to form conjunct characters, (ii) most Indic scripts have around 100 unique Unicode characters, (iii) diversity in handwriting styles, (iv) varying ink density around the words, (v) challenging layouts with overlap between words and natural unstructured writing, and (vi) datasets with only a limited number of writers and examples.
With this competition, we motivate the researchers to continue researching Indic handwriting text recognition tasks to prevent the risk of vanishing Indic scripts/languages. In this competition, we use a training set of an existing benchmark dataset [4, 3, 6]. We create a new manually annotated validation set and test set for validation and testing purposes. A total of eighteen different teams around the world registered for this competition. Among them, only six teams submitted the results along with algorithm details. The winning team Upstage KR achieves an average 95.94% Character Recognition Rate (CRR) and 88.31% Word Recognition Rate (WRR) over ten languages.

Keywords: OCR · Indic handwriting text recognition · Indic language Indic script · conjunct characters.

1 Introduction

Optical Character Recognition (OCR) is the process of converting printed or handwritten images into machine-readable formats. OCR is an essential component in document image analysis. The OCR system usually consists of two main modules (i) text detection module and (ii) text recognition module. The text detection module locates all text blocks within an image, either at the word or line level. The text recognition module attempts to interpret the text image content and translate the visual signals into natural language tokens. Handwriting text recognition is more challenging than printed text because of the imbalanced differences in handwriting styles, content, and time. An individual's handwriting is always unique, and this property creates motivation and interest among researchers to work in this demanding and challenging area.

Many languages worldwide are disappearing due to their limited usage. We can easily use OCR and natural language processing techniques to stop the extermination of languages worldwide. Among 7000 languages[1], the handwritten OCR is available only for a few languages: English [7, 19,

[1] https://www.ethnologue.com/guides/how-many-languages

10], Chinese [23, 22, 18], Arabic [13, 8], and Japanese [12, 16]. Several Indian scripts and languages are at risk due to insufficient research efforts. So, there is an immense need for research on text recognition for Indic scripts and languages.

Among 22 languages in India, few are used only for communication purposes. Hindi, Bengali, and Telugu are the most spoken languages [9]. In most Indic scripts, two or more characters often combine to form conjunct characters [1]. These inherent features of Indic scripts make Handwriting Text Recognition (HWR) more challenging than Latin scripts. Compared to the 52 unique (upper case and lower case) characters in English, most Indic scripts have over 100 unique basic Unicode characters [17].

We organize a challenge on Indic handwriting text recognition tasks in ten different scripts: Bengali, Devanagari, Gujarati, Gurumukhi, Kannada, Malayalam, Odia, Tamil, Telugu, and Urdu. The competition is hosted on http://cvit.iiit.ac.in/ihtr2022/. This challenge motivates researchers to develop/design new algorithms for the above scripts. This challenge suggests a significant new direction for the community to recognize handwritten text in Indic scripts.

The paper is organized as follows. In Section 3, we give details about the dataset used for the competition. The submitted methods are discussed in Section 4. Section 5 shows the results of the competition. The conclusive remark is drawn in Section 6.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+090X	◌ँ	◌ं	◌ः	◌ऄ	अ	आ	इ	ई	उ	ऊ	ऋ	ऌ	ऍ	ऎ	ए	
U+091X	ऐ	ऑ	ऒ	ओ	औ	क	ख	ग	घ	ङ	च	छ	ज	झ	ञ	ट
U+092X	ठ	ड	ढ	ण	त	थ	द	ध	न	ऩ	प	फ	ब	भ	म	य
U+093X	र	ऱ	ल	ळ	ऴ	व	श	ष	स	ह	◌ऺ	◌ऻ	़	ऽ	◌ा	◌ि
U+094X	◌ी	◌ु	◌ू	◌ृ	◌ॄ	◌ॅ	◌े	◌ै	◌ॉ	◌ॊ	◌ो	◌ौ	◌्	◌ॎ	◌ॏ	
U+095X	ॐ					◌ॕ	◌ॖ	◌ॗ	क़	ख़	ग़	ज़	ड़	ढ़	फ़	य़
U+096X	ॠ	ॡ	◌ॢ	◌ॣ	।	॥	०	१	२	३	४	५	६	७	८	९
U+097X	॰	ॱ	ॲ	ऄ	आ	औ	अ॒	अ॓	ऱ	ऴ	ष॒	ग॒	ज॒	ॽ	ड॒	ब॒

Fig. 1. Shows a list of characters and corresponding Unicode characters in the Devanagari script.

2 Indic Languages

There are 22 official languages in India, Assamese, Bengali, Bodo, Dogri, Gujarati, Hindi, Kannada, Kashmiri, Konkani, Malayalam, Manipuri, Marathi, Maithili, Nepali, Odia, Punjabi, Sanskrit, Santali, Sindhi, Tamil, Telugu, and Urdu. Each language has its script. Multiple languages are also derived from a single script. Assamese and Bengali languages use Bengali script. Bodo, Dogri, Hindi, Kashmiri, Konkani, Marathi, Maithili, Nepali, Sanskrit, and Sindhi use the Devanagari script. Gujarati, Kannada, Malayalam, Odia, Punjabi, Tamil, Telugu, and Urdu languages use Gujarati, Kannada, Malayalam, Odia, Gurumukhi, Tamil, Telugu, and Urdu scripts, respectively. While Manipuri uses Bengali/Meitei script, and Santali uses Ol Chiki/Bengali/Odia scripts. Twenty-two languages are derived from 12 scripts. Each script has a set of unique characters and

digits. Fig. 1 and Fig. 2 show the groups of unique characters and corresponding Unicode characters in Bengali and Devanagari scripts, respectively.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
U+098X	ঀ	ঁ	ং	ঃ		অ	আ	ই	ঈ	উ	ঊ	ঋ	ঌ			এ
U+099X	ঐ			ও	ঔ	ক	খ	গ	ঘ	ঙ	চ	ছ	জ	ঝ	ঞ	ট
U+09AX	ঠ	ড	ঢ	ণ	ত	থ	দ	ধ	ন		প	ফ	ব	ভ	ম	য
U+09BX	র		ল				শ	ষ	স	হ			়	ঽ	া	ি
U+09CX	ী	ু	ূ	ৃ	ৄ			ে	ৈ			ো	ৌ	্	ৎ	
U+09DX								ৗ					ড়	ঢ়		য়
U+09EX	ৠ	ৡ	ৢ	ৣ			০	১	২	৩	৪	৫	৬	৭	৮	৯
U+09FX	ৰ	ৱ	৲	৳	৴	৵	৶	৷	৸	৹	৺	৻	□	□	□	

Fig. 2. Shows a list of characters and corresponding Unicode characters in Bengali script.

Compared to English, most Indic scripts consist of over 100 Unicode characters. Two or more characters are often combined to form a conjunct character. Fig. 3 shows a few examples of conjunct characters in Bengali and Hindi. It also shows how the unique characters are combined to create conjunct characters. It also presents a sequence of Unicode code characters corresponding to the unique characters to create conjunct characters. From Fig. 3, for the Bengali language, we observe that 1st and 3rd conjunct characters are obtained by combining three unique characters. At the same time, 2nd and 4th conjunct characters are created by combining five unique characters. In the case of the Hindi language, all four conjunct characters are obtained by combining three unique characters.

Words in Indic languages are created by combining characters from the corresponding script. Fig. 4 shows a few examples of words in Bengali and Hindi languages. It also shows how these (e.g., Bengali and Hindi) words are formed using unique characters from respective scripts (Bengali and Devanagari). Form Fig. 4, we observe 1st and 4th words have 'Upper Matra', 1st and 2nd words have 'Lower Matra'. In the case of Hindi words, all four words have 'Upper Matra', and 1st and 2nd words have both 'Upper Matra and Lower Matra'. The conjunct characters, 'Upper Matra' and 'Lower Matra' in Indic languages, differentiate them from Latin scripts/languages. Due to these special characteristics, the recognition of Indic words is more complex than Latin words.

3 Dataset

We use publicly available benchmark Indic handwriting text recognition datasets [4, 3, 6] for this competition. We use the training set mentioned in [4, 3, 6]. For evaluation purposes, we create a new validation set (VAL-IHW-ICDAR-2023) and test set (TEST-IHW-ICDAR-2023), which are manually annotated. For each of the ten languages, we manually annotate 6000 word images from handwriting pages written by 100 writers. Table 1 presents the statistics of training, validation, and test sets

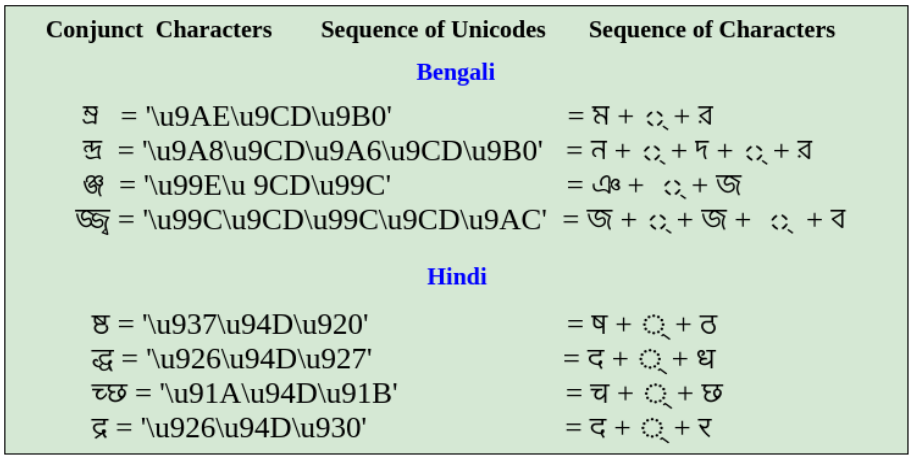

Fig. 3. Shows a few samples of conjunct characters in both Bengali and Hindi languages.

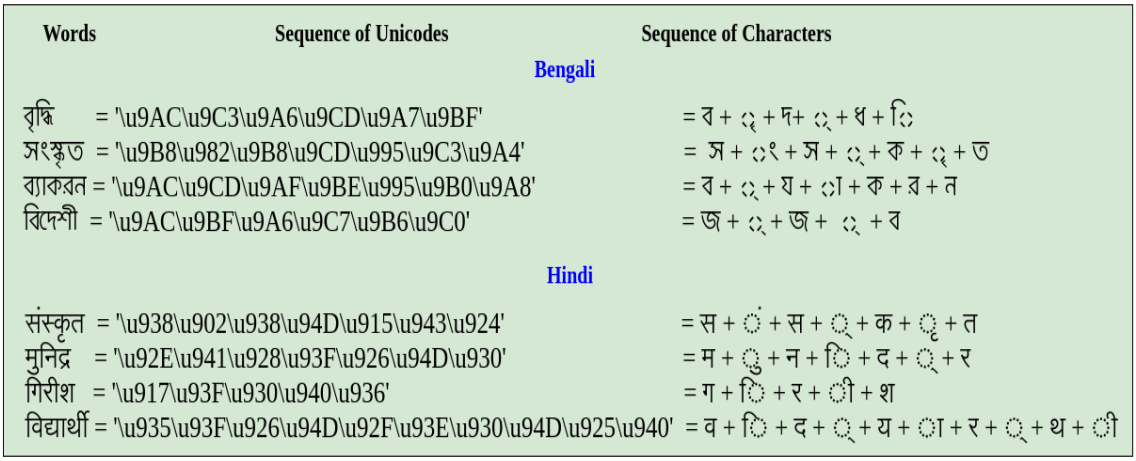

Fig. 4. Shows a few sample words from both Bengali and Hindi languages.

for this competition. It contains word-level images of ten Indic languages: Bengali, Devanagari, Gujarati, Gurumukhi, Kannada, Odia, Malayalam, Tamil, Telugu, and Urdu. We also create a list of unique words from the training set of a language which indicates the lexicon of this language. The lexicon can be used as a language model for post OCR error correction to improve OCR performance. Table 1 also shows the statistic of words in the dataset. The table shows that Malayalam has the most significant average word length among all other languages. Fig. 5 shows sample word images from the test set. The users in the competition are allowed to use additional real and/or synthetic data for training purposes. But they need to provide useful information about the additional dataset for training.

Script	Training Set				Validation Set				Test Set				#Lexicon
	#Image	Word Length			#Image	Word Length			#Image	Word Length			
		Max.	Min.	Avg.		Max.	Min.	Avg.		Max.	Min.	Avg.	
Bengali	82554	29	1	7	1000	13	1	5	5000	19	1	5	11295
Devanagari	69853	23	1	5	1000	13	1	4	5000	15	1	4	11030
Gujarati	82563	22	1	6	1000	15	1	5	5000	20	2	4	10963
Gurumukhi	81042	22	1	5	1000	11	1	4	5000	12	1	4	11093
Kannada	73517	32	1	9	1000	17	1	6	5000	23	1	7	11766
Malayalam	85270	41	1	11	1000	21	1	7	5000	24	1	8	13401
Odia	73400	29	1	7	1000	15	1	5	5000	18	3	5	13314
Tamil	75736	31	1	9	1000	24	1	7	5000	24	1	7	13292
Telugu	80693	27	1	8	1000	20	1	6	5000	21	1	7	12945
Urdu	69212	14	3	5	1000	11	3	5	5000	9	1	3	11936

Table 1. Division of dataset into training, validation, and test sets. **#:** indicates the number of word-level images. **#Lexicon:** shows the number of unique words from the training set of respective languages.

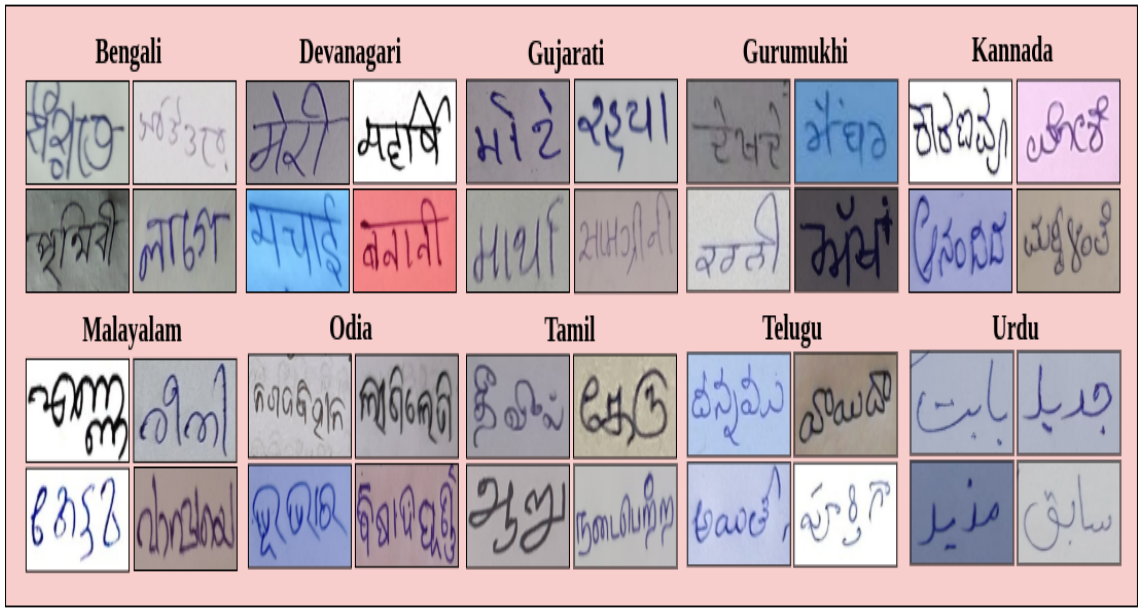

Fig. 5. Shows sample word images from the test set.

4 Methods

This section discusses each submitted method, including the baseline, in detail. Eighteen participants around the world registered for the competition. However, we got submissions from eight of them. These eight teams are (i) EE-Noobies, Indian Institute of Technology, Bombay, India, (ii) Upstage KR, Upstage KR, (iii) huanxiteam, (iv) GG-GradientGurus, Indian Institute of Technology, Delhi, India, (v) light, CCB Financial Technology Co. Ltd, China, (vi) PERO, Faculty of Information Technology, Brno University of Technology, (vii) SRUKR, Samsung R&D Institute

Ukraine, and (viii) LEAP-OCR, Indian Institute of Technology, Bombay, India. However, the teams huanxiteam and GG-GradientGurus are not interested to submitted algorithm details. Due to this, these teams are not included in the draft.

4.1 Baseline

We evaluated the method proposed by Gongidi and Jawahar [6] as our baseline. The baseline network consists of four modules: Transformation Network (TN), Feature Extractor (FE), Sequence Modeling (SM), and finally, Predictive Modeling (PM). The transformation network has six plain convolutional layers with 16, 32, 64, 128, 128, and 128 channels. Each layer has a filter size, stride, and padding size of 3, 1, and 1, followed by a 2×2 max-pooling layer with a stride of 2. The feature extractor module consists of ResNet architecture. The sequence modeling component consists of a 2 layer Bidirectional LSTM (BLSTM) architecture with 256 hidden neurons in each layer. The predictive modeling layer consists of Connectionist Temporal Classification (CTC) to decode and recognize the characters by aligning the feature and target character sequences. We resize input images into 96×256. We use the Adadelta optimizer with Stochastic Gradient Descent (SGD) for all the experiments. We set the learning rate to 1.0, batch size to 64, and momentum to 0.09. For more details, refer to [6][2].

4.2 EE-Noobies

Convolutional Recurrent Neural Networks (CRNN) have emerged as a robust end-to-end trainable architecture for various image-based sequence recognition problems, particularly scene text recognition. A detailed overview of the CRNN implementation pipeline is presented here, highlighting the key components and steps involved in training and inference.

CRNN Implementation Pipeline: The following section describes the key components and steps involved in the CRNN implementation pipeline for handwritten text recognition.

- **Pre-processing:** Input images are pre-processed by resizing them to a fixed height while maintaining their aspect ratios. This step ensures that the input images have a consistent format and size suitable for the CRNN model.
- **Convolutional Layers:** The pre-processed images are passed through convolutional layers, extracting feature maps from the input images. Pooling layers follow these layers to reduce spatial dimensions and capture local information. Batch normalization and activation functions, such as ReLU, are applied to improve training stability and performance.
- **Recurrent Layers:** The feature maps produced by the convolutional layers are fed into a series of recurrent layers, typically implemented using Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) cells. These layers capture the sequential information in the feature maps and model long-range dependencies. To convert the feature maps into a sequence format, a "Map-to- Sequence" operation is performed, which involves reshaping the feature maps by collapsing the spatial dimensions, resulting in a sequence of feature vectors.
- **Bidirectional RNN:** A bidirectional RNN often captures both forward and backward context information in the sequence. It is achieved using two RNNs, one processing the sequence in the forward direction and the other in the reverse order. The outputs of these RNNs are then concatenated at each time step.

[2] The code is available at https://github.com/sanny26/indic-htr

- **Transcription Layer:** The output from the recurrent layers is passed through a transcription layer that employs the Connectionist Temporal Classification (CTC) loss. The CTC layer allows the model to align the input sequences with the target label sequences without explicit character segmentation. It also helps the model recognize variable-length text by collapsing repeated characters and inserting blank tokens between characters.
- **Training:** The CRNN model is optimized during training using gradient-based optimization methods, such as Stochastic Gradient Descent (SGD) or Adam. The CTC loss is used as the objective function, and the gradients are back-propagated through the entire network, updating the weights of both the convolutional and recurrent layers.
- **Inference:** The CRNN model takes an input image and generates an output sequence for inference. The CTC layer outputs a probability distribution over the possible label sequences. A decoding algorithm, such as the best path decoding or beam search decoding, is applied to find the most likely label sequence to obtain the final prediction.

4.3 Upstage UK

To solve the Indic Handwritten Text Recognition task, the Upstage UK team uses the PARSeq framework [2], changing the vision encoder to SwinV2 [11]. The training details are as follows.

- The Upstage UK team uses an in-house synthetic data generator derived from the open-source SynthTiger [24] to generate English synthetic data. It uses Pango Text Renderer to generate 100k Indic synthetic data for each language for pre-training purposes.
- The team gathers a large pool of real-world in-house Korean scene text datasets, English handwriting dataset (IAM) [14], and various Indic handwriting datasets on Kaggle [3] to perform real-world data pre-training.
- The team used the competition data for ten languages to train a multilingual model from the second pre-training stage. Lastly, they train single language models for each of the ten languages respectively, using only the corresponding competition data for that language, starting from the multilingual model in the third stage. For the final submission, they use an ensemble of multiple models per language, whose combination is determined by utilizing the validation accuracy to pick the best ensemble candidates.
  In the processing stage, they cropped the text image to remove the empty area of the image using the Otsu threshold algorithm. In the post-processing stage, they remove the ',' character from Urdu words.

4.4 light

The team light participated in the Indic Handwriting Text Recognition Competition using the TrOCR algorithm. In terms of data, they used GAN networks to perform data augmentation for each language category, increasing the robustness of the models.

Text Recognition Model TrOCR: TrOCR [10] is an end-to-end OCR model based on Transformer. Unlike existing methods, TrOCR is simple and efficient, does not use CNN as the backbone network but instead splits the input text image into image slices and then inputs them into the image Transformer. The encoder and decoder of TrOCR use standard Transformer structures and

[3] https://www.kaggle.com/competitions/bengaliai-cv19/

self-attention mechanisms, with the decoder generating word pieces as the recognition text of the input image. To train the TrOCR model more effectively, the researchers used pre-training models in ViT and BERT modes to initialize the encoder and decoder.

Data Augmentation Method: To increase the diversity of training data and improve the robustness of the model, the team utilized various techniques such as image compression, image warping, stretching, random cropping, and grid mask perspective overlay on the input images.

Additionally, the team attempted to use the ScrabbleGAN [5] generative network for data generation. The team generated 100,000 training data for each language class during model training. However, after comparing the accuracy of the training data with and without data augmentation, the team observed a 1.6% reduction in accuracy. This decrease in accuracy can be attributed to the limited simulation of the generated data. Consequently, the team discarded the data augmentation method.

Training Implementation: First, the team initializes the model to a custom training dataset and generates a character set lexicon file for the current data. Then, the team initializes the custom model weights based on the lexicon file and pre-trained model weights. Finally, the team loads the training data and initializes the custom model weights for model training.

4.5 PERO

The OCR system consists of an optical model (OM) and a language model (LM). For each script, the individual OM and LM are trained. Besides the provided competition datasets, the team also used the original IIIT-INDIC-HW-WORDS validation and test datasets, which differ from the corresponding competition datasets, as additional training data.

The OM is a CRNN-based neural network comprising convolutional blocks and LSTM layers, and it is trained using the CTC loss function. Each OM is trained for 100k iterations with an initial learning rate of 2×10^{-4} and was halved after 80k iterations. After every 2k iterations of training, the model is evaluated on the validation dataset. As the final model, the team selected the one with the lowest character error rate on the validation dataset. The team used data augmentations during training, including color changing, binarization, affine transformations, adding noise, blurring, and masking.

The LM is an LSTM network trained to predict the next character in a sequence. As training data, the team used vocabulary scraped from internet sources (i.e., each word type occurring once in the training data). The team used the transcriptions of the training data from the challenge to select the best model by perplexity, exploring different model widths and dropout rates. The language model is combined with the optical model during prefix beam search over the OM outputs.

4.6 SRUKR

This report describes the approach used for offline handwritten text recognition of Indic languages. It is based on Convolutional Recurrent Neural Network (CRNN) with Connectionist Temporal Classification (CTC) objective function, which allows the processing of image-based sequence-to-sequence data and is widely used in handwriting recognition [20, 21, 26]. There is no need for character segmentation or horizontal scale normalization.

The developed solution is trained on the IHTR2022 and ICDAR2023 datasets. Training scripts were implemented using PyTorch. An essential step of offline handwriting recognition is pre-processing because images can have different sizes, and writing styles differ in the characters' skew, slant, height, and thickness. With pre-processing, the recognition rate is significantly higher. Pre-processing algorithms [21] used in the current recognition solution is described below.

The first pre-processing step is image binarization. For this, we are taking grey-scale images and using Otsu's method. After that, we perform skew and slant correction, then crop and normalize the image height to 64 px.

Each handwriting has its structure of the character's shape with different thicknesses of writing trajectory. To process such samples, we extract the skeleton of these binary images to a skeletal remnant using skeletonization. For the Urdu language, the team flips images to account for right-to-left writing direction. Also, the team performed a set of experiments with blurring and data augmentation to extend the training dataset.

The architecture of the CRNN consists of three components: convolutional layers (CNN), recurrent layers (RNN), and a transcription layer. The CNN extracts feature from the image based on MobileNet-V3 and EfficientNet-V2. The team modifies strides to reduce the sequence length by the X-axis 8 times. It provides a sufficient number of frames necessary for recognition. EfficientNet-V2 gives better results, and at the same time, it has more weight and works longer. The team uses a 1-layer bidirectional GRU with 128 cells as the RNN. Detailed configuration of every layer for trained recognition CRNN is given below. The number of CNN parameters is 5326200, and the number of RNN parameters is 198144.

Configuration of trained CRNN:
Type Configuration
Input width x 64 x 1
CNN (EfficientNet-V2) channels, kernel, stride, layers
Conv2d 24, 3x3, (2, 2), 1
Fused-MBConv 24, 3x3, (1, 1), 2
Fused-MBConv 48, 3x3, (2, 2), 4
Fused-MBConv 64, 3x3, (2, 2), 4
MBConv 128, 3x3, (2, 1), 6
MBConv 160, 3x3, (2, 1), 9
Reshape the image to a sequence
Conv1d in: 320, out: 128, kernel: 1, stride: 1
RNN
Bidirectional GRU hidden units: 128
Transcription
Linear (256 + 1(bias)) x (chars + 1 (blank))
Output width / 8 x (chars + 1 (blank))

In the recognition phase for decoding, the team uses an adapted version of the token passing algorithm [25]. It allows the utilization of a language model and improves the recognition rates. An adapted token-passing algorithm finds the most probable sequence of complete words, given the class-based 3-gram language model. The dictionary contains about 150 thousands of words.

4.7 LEAP-OCR

Handwriting recognition is one of the most widely researched problems. Handwritten characters have many variations and are available in many scripts and languages, making this problem more challenging. Furthermore, handwritten text in Indic languages also has concerned complexities of conjunct consonants, which further adds to the challenges. Deep learning is widely used to recognize handwriting. The team proposed a Convolutional Recurrent Neural Network (CRNN) based text recognition model as CRNNs have proven beneficial for image-based sequential identification tasks.

Method: The team uses a CRNN [20] based recognition model for Indic Language Handwritten Text Recognition as proposed in the DocTR [15] framework. The competition dataset is pre-processed in the format required by DocTR, which involves creating a JSON file for image names and corresponding ground truths. The image pre-processing also includes making all of them equal sizes (maintaining aspect ratio) by necessary padding, color inversion for a few images, and adding random noise to make the model more robust. The processed images from the competition dataset for the selected language and the ground are fed to the randomly initialized CRNN model. Similarly, the team trained the CRNN models language-wise, each composed of a distinct set of vocabulary. Once the concerned epochs are completed, we use the learned models for handwriting recognition. Along with the trained model, the language of the handwritten data is also given as input for the inference stage to carry out proper predictions in inference stage.

5 Evaluation

5.1 Evaluation Metrics

Two popular evaluation metrics such as 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

$$ER = \frac{S + D + I}{N}, \tag{1}$$

where S indicates the number of substitutions, D indicates the number of deletions, I indicates the number of insertions, and N number of instances in reference text. In the case of CER, Eq. (1) operates on the 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 case of CRR, Eq. (2) operates on the character level and in the case of WRR, Eq. (2) operates on word level.

We assign a rank to a team based on WRR for each language. Each team will get points based on their rank on each script: rank 1 gets 5 points, rank 2 gets 4 points, rank 3 gets 3 points, rank 4 gets 2 points, and rank 5 and below get 1 point for participation. Not submitting results for a language gets a point 0. Since the competition has ten languages, each team may get different ranks and corresponding points for ten languages. Therefore, finding the winner and runner-up team from the list of groups for ten languages is tricky. To solve this issue, we calculate the final point for a team by summing all points corresponding to the languages of that team. We declare the winner and runner-up teams based on the final points.

5.2 Competition Results

Among eighteen registered participants, six teams submitted the results of ten languages (except EE-Noobies submitted results corresponding to three languages: Devanagari, Tamil, and Urdu. Tables 2, 3, 4, 5, 6, 7, 8, 9, 10, 11 show obtained results by different methods for Bengali, Devanagari, Gujarati, Gurumukhi, Kannada, Malayalam, Odia, Tamil, Telugu, and Urdu, respectively. While Table 12 shows the average CRR and WRR over ten languages for all methods, including the baseline.

For Bengali, only five teams submitted results. Upstage KR obtains the best CRR (98.99%) and WRR (96.10%) among the teams. It is because of using additional synthetic and real datasets and multi-stage training strategies. The second highest scoring team, PERO, achieves CRR (98.11%) and WRR (92.02%), which is significantly closer to the best CRR (98.99%) and WRR (96.10%). The team PERO also used additional real data for training and language models to correct prediction. The team light obtains CRR (97.54%) and WRR (91.62%) and is in the third position. During training, this team also used additional synthetic data (10M word images, 1M word images per language). LEAP-OCR is the minor performer. From the results of Bengali, we observed that using additional training data and language models improves CRR and WRR.

For Devanagari, the team Upstage KR achieves the best CRR (98.02%) and WRR (93.16%). The other three groups, PERO (CRR 97.64%, WRR 91.16%), light (CRR 97.24%, WRR 91.16%), and SRUKR (CRR 96.97%, WRR 91.98%), obtain significantly closer CRR and WRR. All these four teams used additional data for training. However, multi-stage training helps Upstage KR to achieve the best CRR and WRR. On the other hand, teams LEAP-OCR and EE-Noobies obtained the least CRR and WRR as they have not used any additional data during training.

Table 2. Shows comparison of results obtained by several methods on **Bengali**.

Method	**CRR**	**WRR**	**Rank (Point)**
Baseline [6]	93.46	75.34	-
Upstage KR	**98.99**	**96.10**	**1 (5)**
PERO	98.11	92.02	2 (4)
LEAP-OCR	74.58	40.98	5 (1)
SRUKR	96.01	88.06	4 (2)
light	97.54	91.62	3 (3)
EE-Noobies	0.0	0.0	0

Table 3. Shows comparison of results obtained by several methods on **Devanagari**.

Method	**CRR**	**WRR**	**Rank (Point)**
Baseline [6]	92.99	74.72	-
Upstage KR	**98.02**	**93.16**	**1 (5)**
PERO	97.64	91.16	3 (3)
LEAP-OCR	80.68	58.16	4 (2)
SRUKR	96.97	91.98	2 (4)
light	97.24	91.16	3 (3)
EE-Noobies	76.04	41.30	5 (1)

Table 4. Shows comparison of results obtained by several methods on **Gujarati**.

Method	CRR	WRR	Rank (Point)
Baseline [6]	54.70	23.75	-
Upstage KR	83.88	61.4	4 (2)
PERO	**84.38**	61.96	3 (3)
LEAP-OCR	50.12	21.25	5 (1)
SRUKR	82.82	62.38	2 (4)
light	84.08	**62.80**	**1 (5)**
EE-Noobies	0.0	0.0	0

Table 5. Shows comparison of results obtained by several methods on **Gurumukhi**.

Method	CRR	WRR	Rank (Point)
Baseline [6]	90.88	71.42	-
Upstage KR	98.52	**95.28**	**1 (5)**
PERO	**98.62**	95.0	3 (3)
LEAP-OCR	83.97	56.66	5 (1)
SRUKR	97.06	90.22	4 (2)
light	98.44	95.16	2 (4)
EE-Noobies	0.0	0.0	0

Table 6. Shows comparison of results obtained by several methods on **Kannada**.

Method	CRR	WRR	Rank (Point)
Baseline [6]	89.16	56.08	-
Upstage KR	98.8	93.62	2 (4)
PERO	**99.06**	**94.54**	**1 (5)**
LEAP-OCR	85.72	50.46	5 (1)
SRUKR	97.77	91.36	4 (2)
light	98.6	92.58	3 (3)
EE-Noobies	0.0	0.0	0

Table 7. Shows comparison of results obtained by several methods on **Malayalam**.

Method	CRR	WRR	Rank (Point)
Baseline [6]	95.79	77.20	-
Upstage KR	**99.47**	**97.16**	**1 (5)**
PERO	99.19	94.88	2 (4)
LEAP-OCR	73.78	43.24	5 (1)
SRUKR	97.26	87.22	4 (2)
light	98.98	94.46	3 (3)
EE-Noobies	0.0	0.0	0

Table 8. Shows comparison of results obtained by several methods on **Odia**.

Method	CRR	WRR	Rank (Point)
Baseline [6]	89.83	66.34	-
Upstage KR	94.16	81.0	4 (2)
PERO	**94.77**	**83.38**	1 (5)
LEAP-OCR	82.07	46.68	5 (1)
SRUKR	94.04	81.48	3 (3)
light	94.37	82.96	2 (4)
EE-Noobies	0.0	0.0	0

Table 9. Shows comparison of results obtained by several methods on **Tamil**.

Method	CRR	WRR	Rank (Point)
Baseline [6]	97.88	88.48	-
Upstage KR	99.61	97.92	2 (4)
PERO	99.59	97.54	3 (3)
LEAP-OCR	90.63	60.90	6 (1)
SRUKR	99.15	95.62	4 (2)
light	**99.63**	**98.08**	**1 (5)**
EE-Noobies	92.24	66.92	5 (1)

For Gujarati, Gurumukhi, Kannada, Odia, and Tamil languages, the four teams, Upstage KR, PERO, SRUKR, and light, obtained much closer CRR and WRR. For these languages, additional synthetic and real data help the models to predict higher and closer CRR and WRR.

In the case of the Malayalam language, using additional real data for the teams Upstage KR, PERO, light, and SRUKR help the corresponding models achieve good performance. Using synthetic and real data makes the Upstage KR team achieve the highest CRR and WRR.

For Telugu, two teams, Upsatege KR (CRR 98.53, WRR 91.18) and PERO (CRR 98.63, WRR 91.44), obtain significantly closer output. While the other two teams SRUKR (CRR 97.5, WRR

Table 10. Shows comparison of results obtained by several methods on **Telugu**.

Method	CRR	WRR	Rank (Point)
Baseline [6]	95.82	76.53	-
Upstage KR	98.53	91.18	2 (4)
PERO	**98.63**	**91.44**	**1 (5)**
LEAP-OCR	81.07	36.12	6 (1)
SRUKR	97.5	88.84	3 (3)
light	97.61	86.62	4 (2)
EE-Noobies	84.2	40.48	5 (1)

Table 11. Shows comparison of results obtained by several methods on **Urdu**.

Method	CRR	WRR	Rank (Point)
Baseline [6]	76.23	36.78	-
Upstage KR	89.38	76.3	2 (4)
PERO	89.79	73.34	3 (3)
LEAP-OCR	59.1	22.42	5 (1)
SRUKR	**91.68**	**82.12**	**1 (5)**
light	78.32	48.5	4 (2)
EE-Noobies	0.0	0.0	0

Table 12. Shows comparison of average results over ten languages obtained by several methods.

Method	CRR	WRR	Points
Baseline [6]	87.67	64.66	-
Upstage KR	95.94	**88.31**	**40**
PERO	**95.98**	87.53	38
light	94.48	84.39	34
SRUKR	95.03	85.93	29
LEAP-OCR	76.17	43.69	11
EE-Noobies	84.16	49.57	3

88.84) and light (CRR 97.61, WRR 86.62), also get substantially closer results. The additional synthetic and real data helps the model better recognize all these four teams. The teams LEAP-OCR and EE-Noobies are inferior CRR and WRR because of no other synthetic and real data for training.

For Urdu, the writing sequence is different from other languages. It is written from right-to-left, while the word is written from left-to-right for other languages. All four teams, Upstage KR, PERO, light, and SRUKR, used additional data for training, reasonable CRR, and WRR obtained. As the words are written from right-to-left, the results of the teams Upstage KR, PERO, and light are lesser than that of SRUKR. The team SRUKR obtained the best CRR (91.68%) and WRR (82.12%). The team SRUKR flips word images to account for right-to-left writing direction for Urdu. Because of this strategy, the SRUKR obtains the best CRR and WRR and 6% margin in WRR than the 2nd best team Upstage KR.

From the experiments, we observed that additional real and synthetic data helped to learn the representation of word images for recognition. Multi-stage learning further improves recognition accuracy. The language model helps to rectify and correct wrongly recognized words. Though four methods, Upstage KR, PERO, light, and SRUKR, achieve more than 84% WRR over ten languages on average, several wrongly recognized words still need to be corrected. Fig. 6 shows a few sample word images recognized by the submitted methods.

Fig. 6. Shows sample word images recognized by different techniques.

6 Conclusion

This competition motivates the researchers to continue researching Indic handwritten text recognition tasks to prevent the risk of vanishing a few Indic scripts/languages. Eighteen teams registered for this competition. Among them, only six teams submitted results along with algorithm details. The Upstage KR team won the competition and achieved an average CRR of 95.94% and WRR of 88.31% over all languages. At the same time, team PERO (average CRR 95.98% and WRR 87.53% over ten languages) won the runner-up position in this competition. Four of the six teams, Upstage KR, PERO, light, and SRUKR, obtain much closer recognition results. The following factors (i) additional synthetic and/or real training data, (ii) pre-processing of the training data, and (iii) language model for recognition error correction help these teams to achieve high recognition scores. In the case of the Urdu language, words are written from right-to-left direction, totally different from the writing direction from left-to-right for other Indic languages. For Urdu, flipping word images helps the SRUKR team to achieve the best performance (CRR 91.68% and WRR 82.12%). All these four methods set a new benchmark for the Indic handwriting text recognition tasks. These methods open a direction for solving Indic handwriting recognition tasks.

In the future, we will continue this challenge to enrich the literature on Indic handwriting text recognition tasks with methods and datasets. The challenge impacts the OCR community in building better models and creating complex datasets.

Acknowledgement

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

References

1. Script Grammar. for Indian languages (Accessed March 26 2020), http://language.worldofcomputing.net/grammar/script-grammar.html.
2. Bautista, D., Atienza, R.: Scene text recognition with permuted autoregressive sequence models. In: European Conference on Computer Vision. pp. 178–196 (2022)
3. Dutta, K., Krishnan, P., Mathew, M., Jawahar, C.V.: Towards spotting and recognition of handwritten words in Indic scripts. In: ICFHR. pp. 32–37 (2018)
4. Dutta, K., Krishnan, P., Mathew, M., Jawahar, C.: Offline handwriting recognition on Devanagari using a new benchmark dataset. In: DAS. pp. 25–30 (2018)
5. Fogel, S., Averbuch-Elor, H., Cohen, S., Mazor, S., Litman, R.: ScrabbleGAN: Semi-supervised varying length handwritten text generation. In: IEEE/CVF conference on computer vision and pattern recognition. pp. 4324–4333 (2020)
6. Gongidi, S., Jawahar, C.: IIIT-INDIC-HW-WORDs: A dataset for Indic handwritten text recognition. In: ICDAR. pp. 444–459 (2021)
7. Graves, A., Schmidhuber, J.: Offline handwriting recognition with multidimensional recurrent neural networks. In: NIPS (2008)
8. Jemni, S.K., Ammar, S., Kessentini, Y.: Domain and writer adaptation of offline Arabic handwriting recognition using deep neural networks. Neural Computing and Applications (2022)
9. Krishnan, P., Jawahar, C.V.: HWNet v2: An efficient word image representation for handwritten documents. IJDAR (2019)
10. Li, M., Lv, T., Cui, L., Lu, Y., Florencio, D., Zhang, C., Li, Z., Wei, F.: Trocr: Transformer-based optical character recognition with pre-trained models. arXiv (2021)
11. Liu, Z., Hu, H., Lin, Y., Yao, Z., Xie, Z., Wei, Y., Ning, J., Cao, Y., Zhang, Z., Dong, L., et al.: Swin transformer v2: Scaling up capacity and resolution. In: IEEE/CVF conference on computer vision and pattern recognition. pp. 12009–12019 (2022)
12. 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: ICFHR (2018)
13. Maalej, R., Kherallah, M.: Improving the DBLSTM for on-line Arabic handwriting recognition. Multimedia Tools and Applications (2020)
14. Marti, U.V., Bunke, H.: The IAM-database: an english sentence database for offline handwriting recognition. International Journal on Document Analysis and Recognition **5**, 39–46 (2002)
15. Mindee: doctr: Document text recognition. https://github.com/mindee/doctr (2021)
16. Nguyen, K.C., Nguyen, C.T., Nakagawa, M.: A semantic segmentation-based method for handwritten Japanese text recognition. In: ICFHR (2020)
17. Pal, U., Chaudhuri, B.: Indian script character recognition: a survey. Pattern Recognition (2004)
18. 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)
19. Pham, V., Bluche, T., Kermorvant, C., Louradour, J.: Dropout improves recurrent neural networks for handwriting recognition. In: ICFHR (2014)
20. Shi, B., Bai, X., Yao, C.: An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(11), 2298–2304 (2016)
21. Viatchaninov, O., Dziubliuk, V., Radyvonenko, O., Yakishyn, Y., Zlotnyk, M.: Calliscan: on-device privacy-preserving image-based handwritten text recognition with visual hints. In: The Adjunct Publication of the 32nd Annual ACM Symposium on User Interface Software and Technology. pp. 72–74 (2019)
22. Wu, Y.C., Yin, F., Chen, Z., Liu, C.L.: Handwritten Chinese text recognition using separable multi-dimensional recurrent neural network. In: ICDAR (2017)
23. Xie, Z., Sun, Z., Jin, L., Feng, Z., Zhang, S.: Fully convolutional recurrent network for handwritten Chinese text recognition. In: ICPR (2016)

24. Yim, M., Kim, Y., Cho, H.C., Park, S.: SynthTIGER: synthetic text image GEneratoR towards better text recognition models. In: International Conference on Document Analysis and Recognition. pp. 109–124 (2021)
25. Young, S.J., Russell, N., Thornton, J.: Token passing: a simple conceptual model for connected speech recognition systems. Citeseer (1989)
26. Zhelezniakov, D., Zaytsev, V., Radyvonenko, O.: Acceleration of online recognition of 2d sequences using deep bidirectional LSTM and dynamic programming. In: International Work-Conference on Artificial Neural Networks. pp. 438–449 (2019)