Towards Spotting and Recognition of Handwritten Words in Indic Scripts

Kartik Dutta, Praveen Krishnan, Minesh Mathew and C.V. Jawahar
CVIT, IIIT Hyderabad, India
{kartik.dutta, praveen.krishnan, minesh.mathew}@research.iiit.ac.in and jawahar@iiit.ac.in

***Abstract*—Handwriting recognition (HWR) in Indic scripts is a challenging problem due to the inherent subtleties in the scripts, cursive nature of the handwriting and similar shape of the characters. Lack of publicly available handwriting datasets in Indic scripts has affected the development of handwritten word recognizers, and made direct comparisons across different methods an impossible task in the field. In this paper, we propose a framework for annotating large scale of handwritten word images with ease and speed. We also release a new handwritten word dataset for Telugu, which is collected and annotated using the proposed framework. We also benchmark major Indic scripts such as Devanagari, Bangla and Telugu for the tasks of word spotting and handwriting recognition using state of the art deep neural architectures. Finally, we evaluate the proposed pipeline on RoyDB, a public dataset, and achieve significant reduction in error rates.**

***Index Terms*—Indic scripts, Handwritten word spotting, Handwritten word recognition, Off-line handwritten documents.**

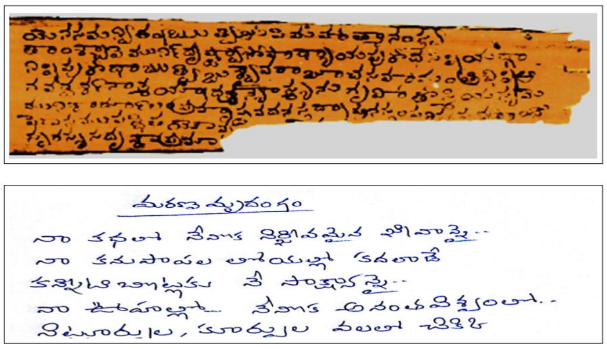


Fig. 1. Examples of handwritten Telugu manuscripts taken from (a) palm leaf text [1] and (b) contemporary writing.

I. Introduction

Understanding text written in handwritten documents enable access to the vast information present in the scanned historical manuscripts. It also provides an efficient organization and indexing of the handwritten content in numerous domains where either the digital technologies are yet to be fully adopted (e.g. court proceedings, medical transcripts, business transaction etc.) and the places where the medium of handwriting is still pervasive (e.g. writing personal notes, content managed in schools and educational institutions etc). Content level access of scanned handwritten documents written in native scripts has a profound value of interests for both historians and the local people. There are numerous handwritten manuscripts [1], notes and essays [2] with high literary content which are scanned as part of digital archives [3]. Fig. 1 shows two examples of handwritten Telugu script. Although there are many methods in the literature for recognizing handwritten content for Latin scripts [4], [5], the field is relatively new for Indic scripts.

Handwritten word recognition (HWR) [4] and spotting (HWS) [6] are the two broad approaches to achieve content level access to the individual words written in a document image. In HWR, given an word image, the task of the recognizer is to predict the character string either in a constrained setting using a lexicon or in an unconstrained setting. While in the case of spotting, given a word image, we project it to an appropriate feature space where nearby feature samples refer to the same lexical word irrespective to its style. Here the problem is formulated in a retrieval setting where given a query word image/string we retrieve all similar word images in a ranked order. Under both scenarios, the major challenges comes from the inherent variability in data. Each individual has a different style of writing and moreover, depending on the various underlying factors, even the style of a single person also changes in different instances of writings.

There are 22 official languages in India, with many more used for communication. Among these languages, Hindi (written in Devanagari script), Bangla and Telugu are the top three languages in terms of the percentage of native speakers [7]. In general there are some inherent features of Indic scripts that makes HWR more challenging as compared to Latin scripts. The number of possible distinct modified and conjunct characters which can be generated by all possible compositions of consonants and vowels is very large, unlike say English where there is no concept of modifiers or conjuncts. Compared to the 52 unique characters present in English, most Indic scripts have over 100 unique basic Unicode characters. For further details on Indic scripts the reader can refer [8].

Motivated by Dutta et. al. [9], in this work we propose a general scheme for gathering and annotating large scale handwritten data from multiple writers for low resource languages such as Indic scripts. We also release a large scale annotated handwritten word level dataset for Telugu and benchmark major Indic scripts for both text spotting and HWR using state of the art deep neural architectures. In our work, we also validate the role of using synthetic data and real handwriting data in an unrelated script (here Latin) in improving word recognition results for Indic scripts.

TABLE I
PUBLIC HANDWRITTEN DOCUMENT BENCHMARKS FOR INDIC SCRIPTS. HERE GT LEVEL REFERS TO THE MODALITY AT WHICH SEGMENTED LABELS WERE PROVIDED FOR THE DATASET.

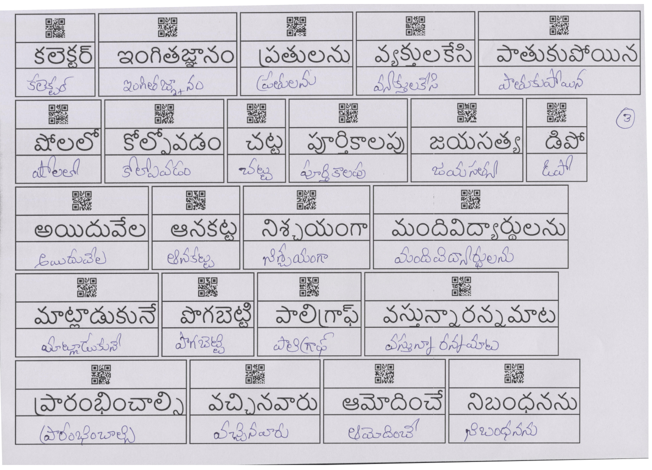
Name	Language	GT Level	#Writers	#Words
PBOK [19]	Bangla	Page	199	21K
	Oriya	Page	140	27K
	Kannada	Page	57	29K
CMATER [20]	Bangla	Line	40	30K
RoyDB [13]	Bangla	Word	60	17K
	Devanagari	Word	60	16K
LAW [21]	Devanagari	Word	10	27K
Tamil-DB [22]	Tamil	Word	50	25K
IIIT-HW-Dev [9]	Devanagari	Word	12	95K
IIIT-HW-Telugu (This Work)	Telugu	Word	11	120K

A. Related Works

Most of the earlier methods in the field of Indic script recognition were focused to the domain of printed documents. Various methods involving the k-nearest neighbors classifier, multi-layer perceptrons were tried out for the task. A summary of these works can be found in [8]. In this paper, we focus on handwritten word images which are more challenging to recognize than printed word images.

There are three popular ways of building handwriting word recognizers for Indic scripts. The first one is to use segmentation-free but lexicon dependent methods which train on recognizing or representing the whole word [10], [11]. Another approach is based on segmenting out the characters within the word image and then use an isolated symbol classifier such as SVM [12]. In [13], the authors segment Bangla and Devanagari word image into upper, middle and lower zones, using morphology and shape matching. The symbols present in the upper and lower zone are recognized using a SVM while a HMM was used to recognize the characters present in the middle zone. Finally, the results from all the three zones are combined. This approach suffers from the drawback that we have to use a script dependent character segmentation algorithm. The third approach treats word recognition as a seq-2-seq prediction problem where both the input and output are treated as a sequence of vectors and we have to maximize the probability of predicting the output label sequence given the input feature sequence [14], [15]. These methods don't require character level segmentation and are not bound to recognizing a limited set of words. In this work we formulate a similar seq-2-seq formulation for word recognition and present results under both a lexicon free (unconstrained) and lexicon based setting.

Similar to text recognition, most of the works in the domain of word spotting for Indic scripts also focused on printed documents [16] which used handcrafted features to represent word images. More recently CNN based features [17] are found to be highly effective for word spotting in Indic scripts. These features are taken from the penultimate layer of the deep CNN network which was trained to perform word classification task. For a general survey on word spotting in handwritten documents, the reader could refer to [18].



కలెక్టర్ ఇంగితజ్ఞానం ప్రతులను వ్యక్తులకేసి పాతుకుపోయిన

షోలలో కోల్పోవడం చట్ట పూర్తికాలపు జయసత్య డిపో ③

అయిదువేల ఆనకట్ట నిశ్చయంగా మందివిద్యార్థులను

మాట్లాడుకునే పొగబెట్టి పాలిగ్రాఫ్ వస్తున్నారన్నమాట

ప్రారంభించాల్సి వచ్చినవారు ఆమోదించే నిబంధనను

Fig. 2. An example of the filled and scanned A4 form used for creating the IIIT-HW-Telugu dataset.

II. INDIC HANDWRITTEN SCRIPTS DATASET

In the domain of off-line handwriting recognition for Indic scripts, there is a severe lack of publicly available datasets that are annotated at the word or line level. Table I lists different publicly available off-line handwritten Indic script datasets, to the best of our knowledge. We have ignored the datasets that only contain isolated characters or do not contain any ground truth. Compared to any standard Latin off-line handwriting recognition dataset like IAM [23], which have more than 100K annotated word images and are written by more than 500 writers, Indic scripts datasets are much smaller in size. Due to lack of standard datasets for any Indic scripts, it is hard to directly compare different methods.

A. IIIT-HW-Telugu Dataset [1]

In continuation to our ongoing efforts to create datasets for Indic scripts in handwritten domain, in this work we release a word level handwritten Telugu dataset which contains word images and its corresponding ground truth. The dataset is annotated using UTF-8, which is the dominant text encoding scheme across the Internet. We used a list of roughly 15K isolated Telugu words scraped from the Internet, courtesy [24]. After pruning out words with non Telugu UTF-8 characters, we have a vocabulary of 12,945 words. On an average each word consists of 9 basic Unicode characters. Each word in our vocabulary was given an unique id. We ensured that every character present in the Telugu UTF-8 range is present in our dataset.

Usually, most handwritten dataset are collected by methods similar to the one followed by [23], where annotators write paragraphs extracted from the text corpus on pages. While this approach is necessary for creating line level recognition datasets, it is sub-optimal for creating a word level dataset, as we have to perform first line and then word segmentation on the filled pages. Also, manual processing has to be done

[1]https://cvit.iiit.ac.in/research/projects/cvit-projects/indic-hw-data

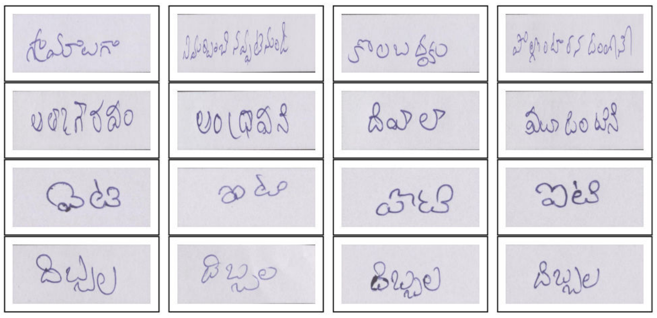


Fig. 3. Sample word images for IIIT-HW-Telugu dataset. The first two rows shows different words that have been written by the same writer. The last two rows shows examples of the same word that has been written by different writers. One can notice both the inter and intra class variability in writing in collected samples.

for correction of labelling errors. Instead, since we were only creating a dataset for word level recognition, we created special forms for data collection, an example of which is shown in Fig. 2.

The forms were generated by first randomly permuting our vocabulary of words and using each word's unique id we generate a QR code corresponding to that word and adding some whitespace at the bottom where the annotator could provide his/her sample. A constraint was added that each word in our vocabulary would occur exactly ten times across all the forms that were generated. After the forms were filled by human writers and scanned, the border outline of each box (see Fig. 2) were extracted from the scanned forms by using binarization followed by flood fill. Now, within each such box, the QR code, reference image and sample had equal width and were easy to segment. Thus, by using this procedure we were able to automatically segment and label the written word. However, the segmentation was not manually corrected. Fig. 3 shows a few sample word images that were collected as part of the dataset. We also manually performed evaluation of our automatic segmentation procedure on a few filled forms. Here we are able to obtain more than 99% word segmentation accuracy with our method as compared to < 95% word segmentation accuracy reported by [23]. Also, we did not require any manual processing for labelling, unlike [23] which required > 14 hours of human effort for label correction.

A total of 1,18,515 words samples were collected from 11 different individuals with different ages and educational backgrounds. Here each writer has written nearly 10K words on an average. The writers were free to write the words in their natural style instead of copying the word image that was shown to them. They were allowed to use any pens of their interest. The forms were scanned using a HP flatbed scanner in a resolution of 600 DPI in color. The train, validation and test sets were created such that there is no overlap of writers between any of the three sets. The training, validation and testing set contains 80637, 19980 and 17898 annotated word images respectively, roughly in the ratio of 70:15:15.

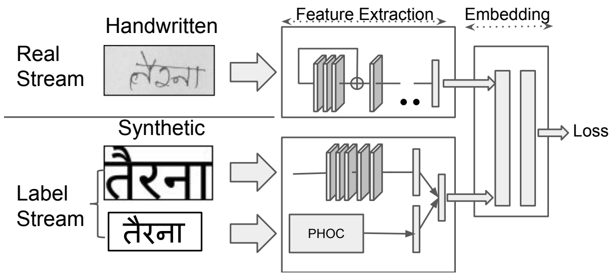


Fig. 4. End2End embedding network for word spotting.

III. Indic Script Word Spotting and Recognition

A. Word Spotting using End2End Embedding Network

Word spotting [6] refers to locating a keyword (e.g word image) given in the form of query from the underlying corpus of document images (segmentation-free) or word images (segmentation-based). Here the query could be either an exemplar word image or could be the corresponding text itself. The basic intuition is to represent word images using an appropriate feature space where the nearest neighbors belongs to same lexical word irrespective of variations from handwritten style and degradation.

We use the End2End deep network proposed in [25] for jointly learning the image and textual feature space which respects the lexical similarity across two modality. Fig. 4 presents the overall architecture. It consists of two streams (real and label) and two major components (feature extraction and embedding). The image stream consists of a deep residual network wherein the real handwritten images are fed and features are computed. While the label stream is further split into two networks: a convolutional network and PHOC [26] feature extractor. Here the convolutional network takes in the synthetic image of the current label and computes its feature representation which is later concatenated with the vectorial representation computed using PHOC. The features computed at both the streams are given to the label embedding layer which projects the two modalities into a common subspace where the similar word images and its corresponding text lies close to each other. The embedding layer is designed in a Siamese style architecture using a multi-layer perceptron. Here the weights across the two modalities are shared. The network uses a multi-task loss function which consists of three losses. The first two loss functions are cross entropy based classification losses for predicting the class of word from the image and label stream by having a softmax layer as last component of embedding layer. This constraints the features to remain discriminative. The third loss function is a cosine embedding loss function which is designed to maximize the correlation among similarly paired data across two modalities. This is computed from the features of the penultimate layer of the embedding layer of size 2048. Given such a representation space where both text and word images can be projected, the word spotting using either query-by-string or exemplar can be achieved using a nearest neighbor based search.

B. Word Recognition using CNN-RNN Hybrid Network

Word recognition [4] is the problem of converting the handwritten content present in an image into machine understandable text. In this work we use a CNN-RNN hybrid architecture, first proposed by [27]. The architecture that we use consists of a spatial transformer layer (STN) [28], followed by a set of residual convolutional blocks, proceeded by a stacked BLSTM (bidirectional LSTM) and ends with a linear layer for transcribing the labels. The role of the STN layer is to perform geometric transformation on the input, so as to correct the distortions that are present in handwriting due to variable hand movements [9]. The convolutional layers here are used for learning a sequence of feature maps, which are then passed on as input to the stacked BLSTMs. We use the CTC [29] loss function to train our network. It converts the predictions generated by the recurrent layers as a maximum probable sequence for the input. In Indic language, similar to syllables in Latin, there exists a basic unit of word called *akshara* which is made up of C*V, where C is a consonant and V is a vowel. Here we decided to use the output space as a series of unicode characters that are present in that dataset that we are currently using for training, instead of using *akshara's* [7]. Using *akshara's* we would have been unable to predict any unseen *akshara*, which is something likely to occur given the small vocabulary encompassed by any Indic datasets.

C. Use of Real Latin and Synthetic Data for Pre-Training

Deep learning architectures contains millions of parameters and in order to avoid over-fitting, availability of huge amounts of training data is crucial. Also as per [30] BLSTMs learn an implicit language model and with Indic scripts having a high number of basic characters, the need for training data is even more acute. We follow the pipeline used by [9] for rendering word images in our synthetic data. We use 60+ publicly available Unicode fonts for Bangla, Telugu and Devanagari to create our synthetic data. We used the same vocabulary that was used for the train set of any particular dataset. Here we render the word image in three different ways: without distortion, with a horizontal line at it's bottom or with a curved baseline. A varying amount of Gaussian noise and kerning were applied to the rendered images. In addition a small amount of rotation, shearing and padding were randomly applied to the generated images. Fig. 5 shows a few examples of synthetically generated Telugu word images. Over 1M such synthetically generated word images were used for pre-training for all experiments. In addition to synthetic data, we also use real data from an un-related script which in our case is Latin where there exists large annotated datasets (e.g. IAM [23]). Similar to works such as [9], [31], we pre-train our model on the IAM train set and present our analysis in Section IV-C for Indic word recognition.

IV. Experiments

A. Datasets

In addition to the IIIT-HW-Telugu dataset which is introduced as part of this work, we use the following public datasets

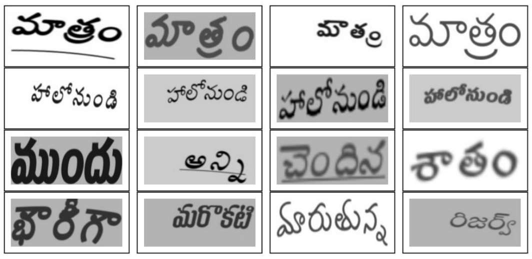


Fig. 5. Few word images that were used in the Telugu synthetic dataset. The first two rows shows variations for the same word. The last two rows shows variations across different words.

in our work:

- **IAM Handwriting Database [23]:** It includes contributions from over six hundred writers and comprises of 115,320 words in English. It is only used for pre-training purposes.
- **Indic Word Database / RoyDB [13]:** It contains samples from over sixty writers and consists of two tracks: Bengali and Devanagari. The Bengali track compromises of 17,901 binarized handwritten word images and the Devanagari track comprises of 16,128 handwritten gray-scale word images. On an average, the label corresponding to a word image in either track consists of 4 characters.
- **IIIT-HW-Dev [9]:** It contains samples from twelve writers and contains of 95,381 handwritten word image samples. On an average each word in the dataset consists of 8 basic Unicode characters. Unlike RoyDB this database is labeled using the UTF-8 encoding.

  We follow the standard partition for training, validation and testing for all datasets and use the standard lexicon for lexicon-based decoding.

B. Word Spotting Results

We follow the evaluation protocol for word spotting as presented in [26] using the train/val/test splits created for the Indic datasets. We conduct both query-by-string (QBS) and query-by-example (QBE) on the test corpus. For QBE setting, the queries are the subset of words taken from the test corpus which has a frequency of more than 1. However all the words were kept in the retrieval set. We also removed the top-1 retrieval since it is query image itself. For QBS scenario, we take the unique set of strings in the test set as queries. In both cases, we report the mean average precision value (mAP) which is standard measure for a retrieval task such as word spotting. Table II presents the results of word spotting on Hindi (Devanagari script), Telugu and RoyDB handwritten datasets. Here we observe quite high performance above 95% mAP which is better than the similar sized corpus in English. We believe this is because of the agglutinative nature of Indic languages which increases the vocabulary size by making longer words. These words gives larger context which is typically found better for word spotting systems to capture feature representation effectively. The above property is quite complementary to word recognition systems where with longer words the probability of making a mistake also

TABLE II
WORD SPOTTING PERFORMANCE USING END2END EMBEDDING ARCHITECTURE ON THE HINDI, TELUGU AND BANGLA DATASETS.

Dataset	Script	QBE	QBS
RoyDB [13]	Devanagari	0.9439	-
	Bangla	0.9627	-
IIIT-HW-Dev [9]	Devanagari	0.9626	0.9690
IIIT-HW-Telugu (This Work)	Telugu	0.9826	0.9843

TABLE III
HWR PERFORMANCE OF THE CNN-RNN HYBRID ARCHITECTURE ON THE IIIT-HW-TELUGU DATASET.

Method	WER	CER
CNN-BLSTM	37.92	9.15
SCNN-BLSTM	34.52	7.83
Synth-SCNN-BLSTM	27.61	5.22
IAM-SCNN-BLSTM	**23.98**	**4.58**
IAM-SCNN-BLSTM-Lexi	**1.07**	**0.3**

grows. We also evaluated the out-of-vocabulary (OOV) performance of our word spotting system. Here we obtained in an OOV mAP of 0.9767 for IIIT-HW-Telugu dataset. The comparable performance of OOV rate shows the robustness of the embedding.

C. Indic Word Recognition Results

For word recognition, in all experiments we report the word and character error rate value (WER & CER respectively), which are the standard metrics for the HWR task. Table III shows the recognition results of various variants of the CNN-RNN hybrid architecture on the IIIT-HW-Telugu dataset. All entries except the last one show results in a lexicon free decoding setting, where decoding is not restricted to any set of chosen words. The various models and their training strategies are mentioned below:

- CNN-BLSTM uses the architecture mentioned in Section III-B, without the STN module. It is trained only on the IIIT-HW-Telugu train set.
- SCNN-BLSTM uses the architecture mentioned in Section III-B (including the STN module). It is also trained only on the IIIT-HW-Telugu train set.
- Synth-SCNN-BLSTM uses the same architecture as above. It is first pre-trained on Telugu synthetic data and then fine-tuned on the IIIT-HW-Telugu train set.
- IAM-SCNN-BLSTM uses the same architecture as above. It is first pre-trained on the IAM train data, then fine-tuned on Telugu synthetic data and finally fine-tuned again on the IIIT-HW-Telugu train set.
- IAM-SCNN-BLSTM-Lexi has the same architecture and training pipeline as above but uses lexicon based decoding. Here the lexicon consists of the entire vocabulary used in the dataset.

From Table III we see the effectiveness of our architectural choice, the STN module and performing pre-training with both synthetic and IAM data.

We also conducted experiments on publicly available RoyDB [13] and IIIT-HW-Dev [9] datasets, using the data split mentioned in the respective dataset. Here we used the same pipeline as IAM-SCNN-BLSTM, just using the appropriate synthetic and real train data for each dataset. Table IV and V report the results for the RoyDB and IIIT-HW-Dev dataset respectively. As we can see, our method achieves the state of the art results. For all of the datasets, if we use a lexicon for decoding, we achieve far better results than the current state of the art in various Latin datasets such as IAM. This observation is most clear in Table III. One reason is that using a lexicon corrects a lot of errors caused due to confusion between modifiers and conjunct characters. To justify this, we take the lexicon of the IAM and IIIT-HW-Telugu dataset respectively and Fig. 7 shows the percentage of valid words that can be converted from one valid word to another, at certain edit distance values, for both datasets. It clearly shows that words in the lexicon of our Telugu dataset are further apart than the words in English. The avg. edit distance to convert a random valid word from the IAM dataset to another valid word is 7.32 while it's 9.41 for the IIIT-HW-Telugu dataset.

TABLE IV
HWR RESULTS ON THE ROYDB DATASET USING THE CNN-RNN HYBRID NETWORK. WE USED THE LEXICON THAT WAS RELEASED AS PART OF THE DATASET.

Method	**Seg.**	**Lexicon**	**Track**	**WER**	**CER**
Dutta et al. [15]	Word	Free	Bangla	10.71	3.49
This Work				**7.04**	**2.55**
Dutta et al. [9]			Devanagari	9.57	3.24
Dutta et al. [15]				12.23	5.17
This Work				**8.91**	**3.14**
Dutta et al. [15]		Based	Bangla	4.30	2.05
Adak et al. [14]				14.57	-
Roy et al. [13]				16.61	-
This Work				**2.85**	**1.39**
Dutta et al. [9]			Devanagari	4.32	2.07
Dutta et al. [15]				5.13	2.72
Roy et al. [13]				16.61	-
This Work				**3.78**	**2.01**

TABLE V
HWR ON THE IIIT-HW-DEV DATASET USING THE CNN-RNN HYBRID NETWORK. HERE WE USED THE LEXICON RELEASED AS PART OF THE DATASET.

Method	**Seg.**	**Lexicon**	**WER**	**CER**
Dutta et al. [9]	Word	Free	26.22	8.64
This Work			**19.52**	**6.41**
Dutta et al. [9]		Based	11.27	4.90
This Work			**3.40**	**1.52**

Fig. 6 shows the recognized outputs for a few sample word images, from all the 4 datasets used in the paper, using the IAM-SCNN-BLSTM-Lexi model in an unconstrained setting. As we can see, most of the errors were caused by ambiguities in the original word image, due to the subtle differences in the shape of characters.

V. CONCLUSION AND FUTURE WORKS

This work is in effort of streamlining the HWR and HWS methods for major Indic scripts. In this direction, we are currently working to release newer datasets in different scripts and also increase the number of writers. In future, we would

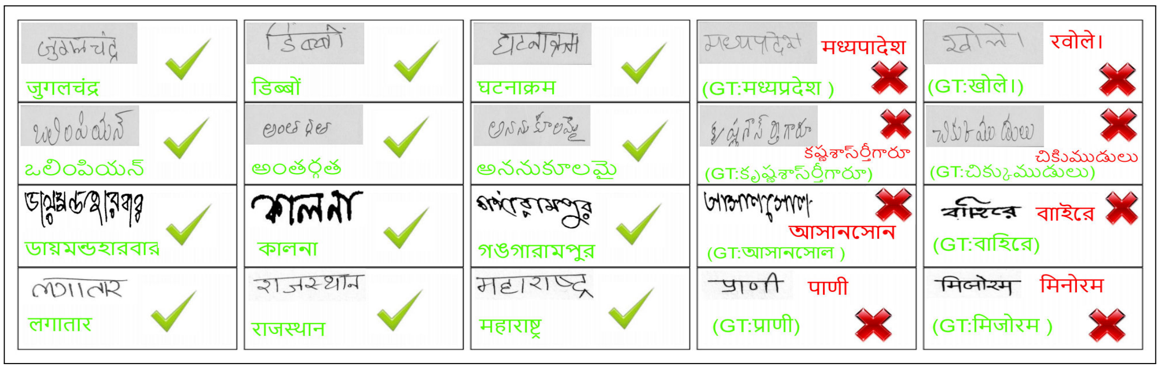


Fig. 6. Qualitative results of the cnn-rnn hybrid architecture on the IIIT-HW-Dev (1st row), iiit-hw-telugu (2nd row), RoyDB-Bangla (3rd row) and RoyDB-Devanagari (last row) datasets. Here GT refers to the ground truth.

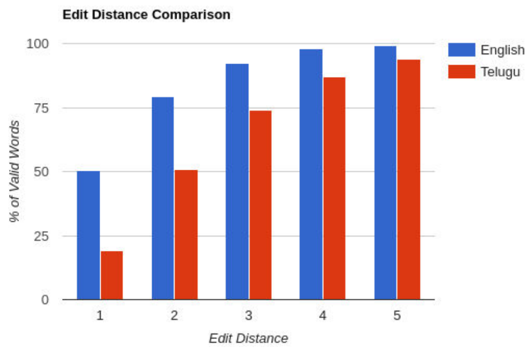


Fig. 7. Percentage of words that get converted to another valid word in Telugu and English. Here Telugu words are from the vocabulary of the IIIT-HW-Telugu dataset, while the English words are from the vocabulary of the IAM dataset.

also like to integrate language model based decoding to further enhance the recognition performance.

References

[1] T. V. Lakshmi, P. N. Sastry, and T. Rajinikanth, "A novel 3d approach to recognize telugu palm leaf text," *EST*, 2017.

[2] C. Adak and B. B. Chaudhuri, "Extraction of doodles and drawings from manuscripts," in *ICPRMI*, 2013.

[3] N. Balakrishnan, R. Reddy, M. Ganapathiraju, and V. Ambati, "Digital library of India: a testbed for Indian language research," *TCDL*, 2006.

[4] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *PAMI*, 2009.

[5] R. J. Milewski, V. Govindaraju, and A. Bhardwaj, "Automatic recognition of handwritten medical forms for search engines," *IJDAR*, 2009.

[6] T. M. Rath and R. Manmatha, "Word spotting for historical documents," *IJDAR*, 2007.

[7] V. Vinitha, "Error detection and correction in Indic OCRs," Master's thesis, International Institute of Information Technology Hyderabad, 2017.

[8] U. Pal and B. Chaudhuri, "Indian script character recognition: a survey," *PR*, 2004.

[9] K. Dutta, P. Krishnan, M. Mathew, and C. V. Jawahar, "Unconstrained handwriting recognition on devanagari script using a new benchmark dataset," in *DAS*, 2018.

[10] B. Shaw, U. Bhattacharya, and S. K. Parui, "Combination of features for efficient recognition of offline handwritten devanagari words," in *ICFHR*, 2014.

[11] B. Shaw, S. K. Parui, and M. Shridhar, "Offline handwritten devanagari word recognition: A holistic approach based on directional chain code feature and HMM," in *ICIT*, 2008.

[12] S. Arora, D. Bhattacharjee, M. Nasipuri, L. Malik, M. Kundu, and D. K. Basu, "Performance comparison of svm and ann for handwritten devnagari character recognition," *arXiv preprint arXiv:1006.5902*, 2010.

[13] P. P. Roy, A. K. Bhunia, A. Das, P. Dey, and U. Pal, "HMM-based Indic handwritten word recognition using zone segmentation," *PR*, 2016.

[14] C. Adak, B. B. Chaudhuri, and M. Blumenstein, "Offline cursive Bengali word recognition using CNNs with a recurrent model," in *ICFHR*, 2016.

[15] K. Dutta, P. Krishnan, M. Mathew, and C. V. Jawahar, "Towards accurate handwritten word recognition for Hindi and Bangla," in *NCVPRIPG*, 2017.

[16] R. Shekhar and C. V. Jawahar, "Word image retrieval using bag of visual words," in *DAS*, 2012.

[17] P. Krishnan and C. V. Jawahar, "HWNet v2: An efficient word image representation for handwritten documents," *arXiv preprint arXiv:1802.06194*, 2018.

[18] R. Ahmed, W. G. Al-Khatib, and S. Mahmoud, "A survey on handwritten documents word spotting," *IJMIR*, 2017.

[19] A. Alaei, U. Pal, and P. Nagabhushan, "Dataset and ground truth for handwritten text in four different scripts," *IJPRAI*, 2012.

[20] R. Sarkar, N. Das, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu, "Cmaterdb1: a database of unconstrained handwritten bangla and bangla–english mixed script document image," *IJDAR*, 2012.

[21] R. Jayadevan, S. R. Kolhe, P. M. Patil, and U. Pal, "Database development and recognition of handwritten devanagari legal amount words," in *ICDAR*, 2011.

[22] S. Thadchanamoorthy, N. Kodikara, H. Premaretne, U. Pal, and F. Kimura, "Tamil handwritten city name database development and recognition for postal automation," in *ICDAR*, 2013.

[23] U.-V. Marti and H. Bunke, "The iam-database: an english sentence database for offline handwriting recognition," *IJDAR*, 2002.

[24] D. Goldhahn, T. Eckart, and U. Quasthoff, "Building large monolingual dictionaries at the leipzig corpora collection: From 100 to 200 languages." in *LREC*, 2012.

[25] P. Krishnan, K. Dutta, and C. V. Jawahar, "Word spotting and recognition using deep embedding," in *DAS*, 2018.

[26] J. Almazán, A. Gordo, A. Fornés, and E. Valveny, "Word spotting and recognition with embedded attributes," *PAMI*, 2014.

[27] B. Shi, X. Bai, and C. Yao, "An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition," *PAMI*, 2016.

[28] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *NIPS*, 2015.

[29] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *ICML*, 2006.

[30] E. Sabir, S. Rawls, and P. Natarajan, "Implicit language model in LSTM for OCR," in *ICDAR*, 2017.

[31] T. Bluche and R. Messina, "Gated convolutional recurrent neural networks for multilingual handwriting recognition," in *ICDAR*, 2017.