2019 International Conference on Document Analysis and Recognition (ICDAR)

Segmentation-Free Bangla Offline Handwriting Recognition using Sequential Detection of Characters and Diacritics with a Faster R-CNN

Nishatul Majid
Department of Electrical and Computer Engineering
Boise State University
Boise, Idaho, USA
nishatulmajid@u.boisestate.edu

Elisa H. Barney Smith
Department of Electrical and Computer Engineering
Boise State University
Boise, Idaho, USA
ebarneysmith@boisestate.edu

***Abstract*—This paper presents an offline handwriting recognition system for Bangla script using sequential detection of characters and diacritics with a Faster R-CNN. This is an entirely segmentation-free approach where the characters and associated diacritics are detected separately with different networks named C-Net and D-Net. Both of these networks were prepared with transfer learning from VGG-16. The essay scripts from the Boise State Bangla Handwriting Dataset along with standard data augmentation techniques were used for training and testing. The F1 scores for the C-Net and D-Net networks are 89.6% and 93.2% respectively. Afterwards, both of these detection modules were fused into a word recognition unit with CER (Character Error Rate) of 11.2% and WER (Word Error Rate) of 24.4%. A spell checker further minimized the errors to 8.9% and 21.5% respectively. This same method is likely to be equally effective on several other Abugida scripts similar to Bangla.**

***Index Terms*—Bangla handwriting recognition, Offline Handwriting recognition, Segmentation-free handwriting recognition, Handwriting recognition using Faster R-CNN, Character spotting**

I. Introduction

This paper presents a segmentation-free offline recognition system for unconstrained Bangla handwritten text. Bangla (a.k.a. Bengali) is one of the most widely used languages in the world. It is the national and official language of Bangladesh, and the 2nd most widely spoken language of India. With 250-300 million total speakers, it is the 7th most spoken native language in the world by population. The Assamese script is nearly identical to the Bangla script and together they make the 5th most widely used writing system. It is a very literature-rich language with a millennium-old history and folk heritage.

From the handwriting recognition perspective, Bangla is a very tough and challenging script with its inherent cursive and connected nature. The fusion of characters into conjuncts, along with use of diacritics translates into more than two thousand almost inseparable glyphs. While in general people write in vividly different manners, there are several styles of writing which make many of the characters and conjuncts from the same class appear almost entirely different than each other. These are a few of the fundamental reasons why segmentation-based approaches are extremely complicated for scripts like Bangla. One good attempt was by Pal et al. with a lexicon driven segmentation-based Bangla word recognition with water reservoir based segmentation and modified quadratic discriminant function (MQDF) based likelihood computation of a character [1]. They used directional features and on a Bangla city-name test dataset it produced decent accuracy.

Segmentation-free document image processing has become the preferred approach in recent days and one of the fundamental application areas is word spotting. These word spotting processes are usually very flexible with respect to script. Rothacker et al. proposed an architecture using a Bag-of-Features (BoF) representation powered with SIFT descriptors to feed a Hidden Markov Model (HMM) for query based word spotting [2]. This not only produced excellent results with the George Washington dataset, but also turned out to be very successful with Arabic handwriting and Bangla machine printed documents [3], [4]. Wshah et al. proposed a script independent framework backed by an HMM to work with English, Arabic and Devanagari handwriting [5]. Das et al. presented a system for Indic script, primarily focused on Bangla and Devanagari, where Pyramid Histogram of Oriented Gradient (PHOG) backed by an HMM was used as the word spotting framework combining foreground and background features [6]. In recent years, deep convolutional networks have been outperforming other machine learning algorithms in various computer vision tasks, including document image analysis. Sudholt et al. introduced a Convolutional Neural Network (CNN) trained with Pyramidal Histogram of Characters (PHOC), namely PHOCNet which outperforms contemporary state-of-the-art approaches on various datasets for the task of word spotting [7]. Our proposed approach is analogous to the methods of word spotting in the sense that we intend to avoid segmentation by spotting the characters in a word image, rather than spot a word in a line or document.

Deep neural networks are extremely useful in segmentation-free handwriting transcription as well. Bluche et al. used the ROVER (Recognition Output Voting Error Reduction) scheme for combining four models and reported results on the IAM (English) dataset [8]. Two of them are based on

2379-2140/19/$31.00 ©2019 IEEE
DOI 10.1109/ICDAR.2019.00045

Bidirectional Long Short-Term Memory (BDLSTM) RNNs, and the other two are based on deep Multi-Layer Perceptrons (MLPs). Menasri et al. proposed a system that uses seven recognizers based on three different technologies - grapheme based hybrid HMM, Gaussian Mixture HMM and Recurrent Neural Networks (RNN) [9]. They presented their results with the RIMES (French) dataset. Stahlberg et al. proposed a method that uses fully connected deep neural networks for optical modeling with features extracted from raw pixel gray-scale intensity values of foreground segments [10] and presented their results with the IFN/ENIT (Arabic) dataset. Dutta et al. worked with the Devanagari script [11], which shares a close resemblance to Bangla. They used a CNN-RNN hybrid architecture and lexicon based decoding on the IIIT-HW-Dev dataset. This work recognizes whole words from a limited vocabulary. The work we propose is recognizing the letters, therefore does not depend on a fixed vocabulary. A few similar works have been done for Bangla with restricted vocabulary such as Bhowmik et. al. using HOG descriptor with MLP classifier [12], Adak et. al. using CNN with a recurrent model [13].

Rather than segmenting or using unsegmented word recognition, we establish a character/diacritic spotting mechanism from the word images. Subword based models are gaining popularity. Davis et al. [14] proposed a Computer Assisted Transcription (CAT) method with subword spotting for historical documents. Poznanski et al. used CNN-N-gram modeling [15] and obtained excellent results on the IAM, RIMES and IFN/ENIT datasets. N-gram based models usually work best for scripts with a relatively small alphabet, and Bangla is not one of them.

One of the primary challenges in this field has always been the availability of meticulously crafted datasets. For word recognition, many approaches require character or grapheme level ground truth tagging information. To initiate this process, we introduced the Boise State Bangla Handwriting dataset [16] with samples from 100 different writers and character level tagging information. This is a continuing process and right now this dataset contains documents from 150 writers yielding about 16,000 words or 55,000 characters. This dataset is freely available at [16]. This, however is still small for deep learning requirements, but is the only available public dataset for Bangla that contains sub-image tagging that can be used with standard object detection approaches.

In this paper, we present an offline segmentation-free handwriting recognition scheme. This is a simple, fast and effective approach which avoids character or zone separations and complex grammatical rules. The challenge was to overcome the limitation of data while dealing with the complicated nature of this script. Offline handwriting recognition is an extremely promising technology, which is specially relevant for developing countries like Bangladesh, where almost everywhere handwritten documents such as letters, applications, forms, agendas etc. are still being used predominantly.

II. Character/Diacritic spotting

In this paper, a Bangla offline handwriting recognition scheme is presented which starts with the image of a handwritten word and yields the transcription. Rather than segmenting the words followed by recognizing the characters, we look for each possible character element (like basic characters, diacritics, conjuncts, punctuation etc.) of the script inside the words and combine the results into text, Fig 1. This also means the recognition is not lexicon dependent.

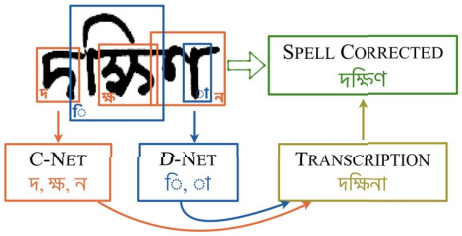

Fig. 1. Overview of the process of sequentially detecting characters and diacritics to obtain a transcription from handwritten text image.

The core foundation of this system is an object detection network with a Faster R-CNN (Regions with CNN features). Here, the objects are the glyphs, which are separated into two major classes, i.e. characters and diacritics. The character class is formed with the basic characters, punctuation and most of the conjunct characters available in the dataset. The diacritic class includes all the diacritics and a few conjuncts which have attributes like diacritics. The network is separately trained for these classes to obtain two specialized networks namely C-Net (Character Network) and D-Net (Diacritic Network). These networks are sequentially applied to word level images to identify their syllabic compositions with locations. This approach significantly reduces the complexity that can arise from the combinatory fusion property of the characters with diacritics and other characters in Bangla, as well as other similar scripts. The other challenges, such as the scarcity of data and large training time requirements were countered with usual methods, i.e. data augmentation and transfer learning. A word recognition unit was formed to fully utilize the C-Net and D-Net detection results in order to obtain a plain transcription. Furthermore, a basic spell corrector was developed to take advantage of easily correctable mistakes. The whole process is described in the following subsections.

Fig. 2. Sample of the dataset tagging labels and bounding boxes.

A. Developing the C-Net and the D-Net

The idea behind separately detecting characters and diacritics is rooted in the structure of the Boise State Bangla Handwriting Dataset. The Bangla script contains four types of elements - basic characters, conjuncts, vowel diacritics and punctuation marks. The vowel diacritics are a grapheme which is always paired with a basic character or conjunct. The arrangement of this diacritic can be in any position relative to the associated unit. That's why, it is only possible to have a pixel level segmentation for distinguishing each unit in a Bangla word, which is extremely complicated and requires such a massive effort that it can be considered impractical or often unnecessary. The same goes with the conjuncts; they can be formed with up-to 4 consonants and quite often these are merged in an irregular fashion. Some of the conjuncts have unique shapes, some appear like a vowel diacritic. Thereby, while preparing the dataset, we chose not to split the diacritics from their pairing, or the conjuncts from themselves, and have rectangular boxes for the units' ground truth information. That is, if a basic or conjunct character has an associated vowel diacritic, they are all considered together as a single unit tagged together with a single box in the ground truth file. An example image with ground truth tagging overlay is shown in Fig. 2. That's the fundamental motive for preparing two individual deep networks, where one looks for the characters and the other looks for the diacritics from a boxed unit.

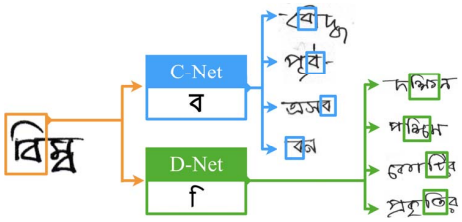

Fig. 3. C-Net and D-Net look for different classes in the words while being trained on same images.

While operating on the same image, the networks produces different detection outputs as illustrated in Fig. 3. C-Net looks for all the basic characters and conjuncts as single entries, with the exception of one basic character and a few conjuncts which looks more like a vowel diacritic extension rather than being a unique shape. Furthermore, the punctuation marks were also included in the C-Net classes. For this dataset, this was a 67 class problem. Similarly, D-Net looks for all vowel diacritics, including the earlier mentioned exceptions and results in a 15 class problem. Therefore, including the background class, the classification output layer becomes a fully connected layer with 68 and 16 nodes respectively for C-Net and D-Net. These class distributions are listed in Table I.

Thus, as a whole, the system can detect all possible combinations that can be obtained from the C-Net and D-Net classes, which includes all the basic consonants and the listed conjuncts with all their possible vowel diacritic presence. Furthermore, since the D-Net conjunct classes fuse with other consonants/conjuncts to form a cluster most of the time in the same way, this approach covers a major portion of the Bangla script without segmenting characters or words.

TABLE I
LIST OF C-NET AND D-NET CLASS DISTRIBUTIONS

C-Net Classes			D-Net Classes		
Basic	Conjunct	Punctuation	Diacritic	Basic	Conjunct
অ, আ, ই, ঈ, উ, ঊ, এ, ঐ, ও, ঔ, ক, খ, গ, ঘ, ঙ, চ, ছ, জ, ঝ, ট, ঠ, ড, ঢ, ণ, ত, থ, দ, ধ, ন, প, ফ, ব, ভ, ম, য, র, ল, শ, ষ, স, হ, ড়, ঢ়, য়, ৎ, ঃ, ঁ	ঙ, শ্চ, ঙ্গ, ঞ্ছ, ঞ্ঝ, ষ্ণ, ষ্ণ, ষ্ট, ষ্ম, ষ্প, স্ত, ন্ন, হ্ম, ক্ষ, ক্ষ, ত্ত, ষ্ণ	‘, (কমা)’, ‘। (দাড়ি)’, ‘- (হাইফেন)’	া, ি, ী, ু, ূ, ে, ৈ, ো, ৌ, ৃ	ঁ	ঁ, ্র, ্য, ্ব

B. Object Detection with Faster R-CNN

There are many types of object detection frameworks such as R-CNN, YOLO, Retina-Net, HMM based models etc. Here, we used a Faster R-CNN which is a faster variant and extension of R-CNN and Fast R-CNN. Faster R-CNN, introduced by Ren et al. [17], provides almost real time object detection and has been extensively used in many applications in machine learning including document image analysis. R-CNN and Fast R-CNN use a region proposal algorithm as a pre-processing step. In the case of Faster R-CNN, this issue is addressed by implementing the region proposal mechanism using the CNN, hence making the region proposal a part of the training and prediction. The predicted region proposals are then reshaped using a RoI (Region of Interest) pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes. Since, offline handwriting recognition can greatly benefit from a speedy behavior in most of its popular application areas, we used Faster R-CNN as the tool for character/diacritic spotting.

C. Transfer Learning from VGG-16

VGG-16 was introduced by the Visual Geometry Group (VGG) from the University of Oxford in 2014 [18]. Although this network is quite large and relatively slow to train, it has been very successful in numerous applications of deep learning. Here, the series VGG-16 architecture was transformed into a DAG (Direct Acyclic Graph) structure to obtain a Faster R-CNN network, shown in Fig. 4. with the pre-trained model weights. The features extracted from the ReLU5_3 (Rectified Linear Unit) layer were processed by a RoI (Region of Interest) pooling layer with 7×7 feature map output size replacing the last max pooling layer from the original architecture. Both the networks for character and diacritic detections, i.e. the C-Net and D-Net, were prepared using this approach.

D. Data Augmentation

Data augmentation is a widely known trick to mitigate the problem of training with smaller datasets, and almost

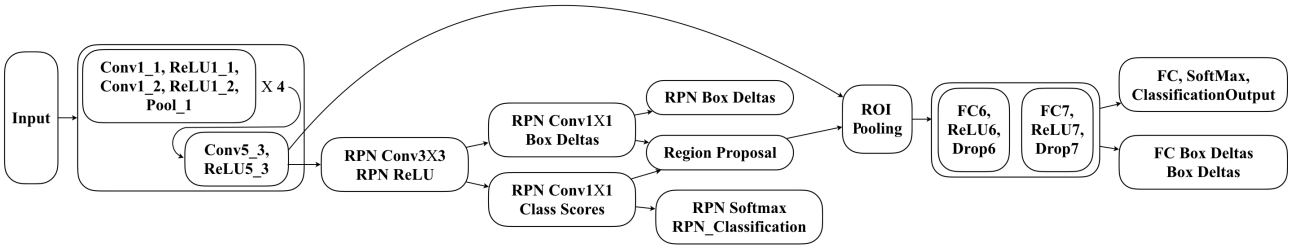

Fig. 4. Layer Graph of C-Net and D-Net, transformation of VGG-16 to a Faster R-CNN.

always it ensures better network training performance. Here, we applied three basic yet very effective techniques: 1) scaling along the X-axis (between 50 - 150% of the original image width), 2) shearing along the X-axis (between -5° and 5° and 3) rotation (between -5° and 5°)). Augmented samples were made by randomly drawing levels for all three of these distortions. For each word we generated three additional images, thus quadrupling the whole training set. Sample augmentation images are shown in fig. 5. This could be expanded to increase the recognition acuracy using more sophisticated augmentation techniques and using more augmented data than the 1:3 ratio we opted for, but at the cost of a much slower training process.

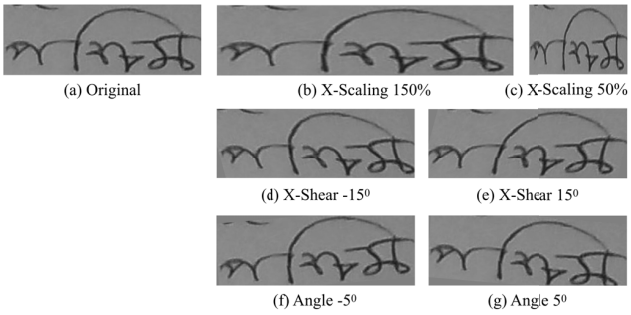

Fig. 5. Data augmentation: (a) original, (b)(c) augmentation with only X-Stretch of 150% and 50%, (d)(e) with only X-Shear of -15°and 15°and (f)(g) with only Angle -5°and 5°.

E. Training the C-Net and D-Net

Word images were extracted from the character level ground truth tagging information obtained from the Boise State Bangla Handwriting dataset [16]. The documents are originally stored in a random order with respect to demographics. Of the 15,656 word images from the first 150 writers, the first 80% were used for training, the next 10% for validation, and the remaining 10% for testing. The 12,525 training images were quadrupled to 50,100 images with data augmentation described earlier and these were used for training both the C-Net and D-Net. Each image was resized to 600 pixels at its smallest dimension during training. Stochastic Gradient Descent with Momentum (SGDM) was used for training with 0.9 momentum. The initial learning value was set to 0.001 with 10 maximum epochs. Negative trainings are done during the process by setting an overlap range with

$$\frac{area(A \cap B)}{area(A \cup B)} \tag{1}$$

where A and B are bounding boxes of the region proposal and actual value obtained from the ground truth file. Overlap ratios up-to 0.6 were used for negative training and higher values were considered positive. Lastly, the number of region proposals to randomly sample from each training image was set to 64. Increasing this number can produce higher training accuracy at the costs of increased memory usage and a slower training process.

C-Net and D-Net both were trained identically with these parameters, except for the difference in number of classes. We would like to acknowledge the high-performance computing support of the R2 Compute Cluster [19] provided by Boise State University's Research Computing Department.

F. Word Recognition using C-Net and D-Net

C-Net and D-Net find the individual elements in a word image, but combining the results to form a transcription requires some further work. The worst detection results from C-Net and D-Net were for the classes of ঊ (dirghô ū) and ঁ (ref) with mAPs of 0.66 and 0.79 respectively. The thresholds for detection confidence of 0.72 and 0.81 for C-Net and D-Net respectively were decided from the precision-recall graph of these worst classes with values that maximize the recalls.

One big issue with the detection was the overlapping of detection bounding boxes. The C-Net and D-Net detections are expected to overlap where there is a diacritic present. But we found individual C-Net and D-Net detections were also overlapping. This is because many Bangla characters/diacritics look like extensions of other characters/diacritics, and conjuncts look like individual characters merged together. A sample of C-Net detection overlaps is demonstrated in Fig. 6. We kept the largest bounding box, discarding the smaller detected bounding boxes that it encapsulates or overlaps with regardless of their confidence score. The overlap was computed using (1). and a value greater than 0.5 was considered an overlap.

Some of the diacritics appear before the character and some of them afterwards, but with Unicode all the diacritics are encoded after their primary character regardless of where they visually appear. Thereby, with all the overlapped results of C-Net and D-Net, the C-Net results were ordered first. There

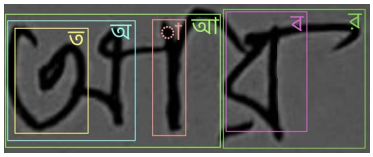

Fig. 6. Sample of the detection overlap issue. Green boxes are the proper detection and the others are detected look-alike sub-characters.

is one exception with the conjunct class '◌্' (Bangla conjunct 'ref'), which appears more like a diacritic and was processed with the D-Net. This actually appears before the associated character class in formation as well as Unicode encoding. Hence, for this case the D-Net result was placed prior to the C-Net one. The overlap computation and threshold were the same as described in the previous step. With this exception, all the detected results were sorted by the x_{min} values of their bounding boxes.

Lastly, some common language rules were applied to mitigate the detection errors. These are not script specific rules and are applicable to almost all other Abugida scripts. 1) A vowel can't have a vowel diacritic, therefore D-Net results overlapping with a vowel were eliminated. 2) A vowel can't form a conjunct, therefore if a C-Net result implied that it encapsulated a vowel with something else, based on confidence score one of them was eliminated. 3) There can't be two consecutive diacritics, so either one of them is a false positive, or there is a character in between with a miss in detection. This was assessed by the space gap between the detected bounding boxes. For overlap or tightly spaced detection, the diacritic with lower confidence was removed. For bigger gaps in between detection (set as 50% of the previous detection width), a blank character is placed in between to ensure we obtain a value closer to the actual elements in that word. 4) Two consecutive C-Net detections having a gap larger than 50% of the width of the first one is considered as a miss in detection, and thereby was filled in with a blank character, again to better assess the number of characters in a word.

Applying all these, the transcription of the words were recorded for performance evaluation. A few samples of the detection results are shown in Fig. 7.

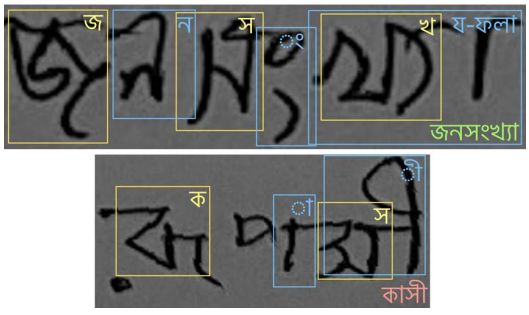

Fig. 7. Detection overlay, successful (top) and failed (bottom). Alternate colors are used to increase visibility.

G. Spell Checker

Although the base accuracy from the word recognition module was already admirable, to see the potential of this method a basic spell checker was prepared and applied to the detection results as a post processing module. The spell checker was developed assessing the individual detection performance of C-Net and D-Net. A Bangla word library containing 450,000 words was used in this process. With the language rules applied as discussed in section II-F, the number of C-Net characters in a word was being estimated with 97.62% accuracy. Therefore, the insertion or removal of a character (from C-Net classes) was excluded from the edit distance calculation. With that we had:

$$Edit\ Distance = \frac{I_D + R_D + S_{C+D}}{No.\ of\ Elements\ Detected} \quad (2)$$

where I_D and R_D are the number of insertions and removals required strictly from D-Net classes and S_{C+D} is the number of substitutions required from all classes. Words not included in the library were considered misspelled words and were replaced by the ones with the least edit distance from the detection.

III. Performance Evaluation

The C-Net and D-Net detection results are evaluated with MAP (Mean Average Precision) values and F1 scores (the geometric mean of Precision and Recall). Histograms of the F1 scores for C-Net and D-Net are shown in Fig. 8. The characters with a low number of occurrences in the dataset ended up having the poorest detection results, because the training simply wasn't adequete. This is specially true if that character has a visual similarity to other classes. The lowest three values we obtained are for 'ঊ' (dirghô ū), 'এ' (e) and 'শ্চ' (shô + chô) of 0.66, 0.71 and 0.73 respectively. These are all from C-Net classes; the D-Net scores are relatively stable and more uniform.

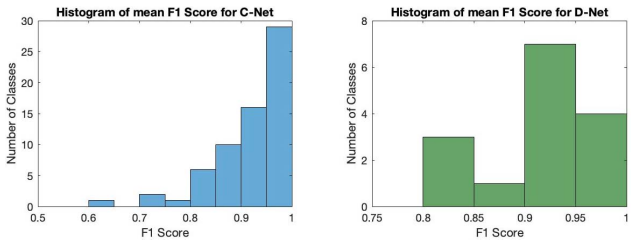

Fig. 8. Histogram of mean F1 score of C-Net and D-Net detection results.

The performance of the word recognition unit was evaluated with precision, recall, MAP, F1 score, CER (Character Error Rate) and WER (Word Error Rate). All the results are presented in Table I. The CER measures the Levenshtein distance normalized by the length of the true word and the WER is the ratio of the reading mistakes at the word level, among all test words.

One of the major problems with the overall detection was with the high number of false positives for the diacritic 'ি-কার'

(AA-kar). For handwriting, in most cases this just becomes a vertical line, hence, it is easy to false detect this inside any other characters/diacritics which include such a straight line.

TABLE I
PERFORMANCE SCORES AND ERROR RATES FOR ALL NETWORKS
(HIGH VALUES ARE DESIRED EXCEPT FOR THE ERROR RATES)

Networks	Parameters	Scores
C-Net	mAP	0.871
	F1 Score	0.896
D-Net	mAP	0.903
	F1 Score	0.932
Word Recognizer	Precision	0.883
	Recall	0.894
	mAP	0.884
	F1 Score	0.900
Word Recognizer Error Rates	CER	0.112
	WER	0.244
	CER (after spell check)	0.089
	WER (after spell check)	0.215

IV. CONCLUSION

While handwriting recognition is still far from being solved, the approach proposed here of changing it to a search problem is an effort to bring it closer to the solution. Bangla and other similar scripts are difficult scripts to work with, and most of the time they end up being approached with many fancy techniques of feature extraction, segmentation, zone-separation, grouping, combining multiple classifiers with sophisticated voting polls etc. Here, our effort is to present a simple yet effective way to deal with the complex nature of any Abugida script. Instead of segmentation and classification, the characters and diacritics were spotted sequentially to obtain a transcription. This not only produces an entirely lexicon independent handwriting recognition using a small and restricted dataset, it also saves the effort of pixel-precise ground truth tagging and allows just a rectangular tagging. The pre-trained VGG16 with a faster RCNN was used here, but in theory, any object detection framework should be compatible with this approach. This is also a very flexible process and is likely to work with many other Abugida scripts with minimal to no redesign. Scripts like Assamese, Tirhuta/Mithilakshar are very close to Bangla from all perspectives. Others, like Devanagari, Gurmukhi, Modi, Sylheti Nagari, etc. share the same formation as Bangla with different characters, and Ranjana/Lanydza, Gujarati, Tibetan etc. are close enough to assume that this methodology has strong potential to contribute to each of these also.

Although we had to work with the limited number of conjuncts that were available in the first release of the Boise State Bangla Handwriting Dataset, this can be easily expanded for the entire script in the same fashion. There are no other comparable transcription-level recognition results reported for Bangla at this time, therefore there was nothing to compare our results with. Also, this method works well with a limited amount of data and is likely to get better and more useable as the dataset grows.

REFERENCES

[1] U. Pal, K. Roy, and F. Kimura, "A lexicon driven method for unconstrained Bangla handwritten word recognition," in *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft, 2006.

[2] L. Rothacker, M. Rusinol, and G. A. Fink, "Bag-of-features HMMs for segmentation-free word spotting in handwritten documents," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, 2013, pp. 1305–1309.

[3] L. Rothacker, S. Vajda, and G. A. Fink, "Bag-of-features representations for offline handwriting recognition applied to Arabic script," in *International Conference on Frontiers in Handwriting Recognition (ICFHR), 2012*, 2012, pp. 149–154.

[4] L. Rothacker, G. A. Fink, P. Banerjee, U. Bhattacharya, and B. B. Chaudhuri, "Bag-of-features hmms for segmentation-free Bangla word spotting," in *Proceedings of the 4th International Workshop on Multilingual OCR*. ACM, 2013, p. 5.

[5] S. Wshah, G. Kumar, and V. Govindaraju, "Script independent word spotting in offline handwritten documents based on hidden markov models," in *2012 International Conference on Frontiers in Handwriting Recognition (ICFHR 2012)*. IEEE, 2012, pp. 14–19.

[6] A. Das, A. K. Bhunia, P. P. Roy, and U. Pal, "Handwritten word spotting in Indic scripts using foreground and background information," in *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*. IEEE, 2015, pp. 426–430.

[7] S. Sudholt and G. A. Fink, "PHOCNet: A deep convolutional neural network for word spotting in handwritten documents," in *Frontiers in Handwriting Recognition (ICFHR), 2016 15th International Conference on*. IEEE, 2016, pp. 277–282.

[8] T. Bluche, H. Ney, and C. Kermorvant, "A comparison of sequence-trained deep neural networks and recurrent neural networks optical modeling for handwriting recognition," in *International Conference on Statistical Language and Speech Processing*. Springer, 2014, pp. 199–210.

[9] F. Menasri, J. Louradour, A.-L. Bianne-Bernard, and C. Kermorvant, "The A2iA French handwriting recognition system at the Rimes-ICDAR2011 competition," in *Document Recognition and Retrieval XIX*, vol. 8297. International Society for Optics and Photonics, 2012, p. 82970Y.

[10] F. Stahlberg and S. Vogel, "The QCRI recognition system for handwritten Arabic," in *International Conference on Image Analysis and Processing*. Springer, 2015, pp. 276–286.

[11] K. Dutta, P. Krishnan, M. Mathew, and C. Jawahar, "Offline handwriting recognition on Devanagari using a new benchmark dataset," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, 2018, pp. 25–30.

[12] S. Bhowmik, M. G. Roushan, R. Sarkar, M. Nasipuri, S. Polley, and S. Malakar, "Handwritten bangla word recognition using hog descriptor," in *2014 Fourth International Conference of Emerging Applications of Information Technology*. IEEE, 2014, pp. 193–197.

[13] C. Adak, B. B. Chaudhuri, and M. Blumenstein, "Offline cursive Bengali word recognition using CNNs with a recurrent model," in *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2016, pp. 429–434.

[14] B. Davis, R. Clawson, and W. Barrett, "Flexible computer assisted transcription of historical documents through subword spotting."

[15] A. Poznanski and L. Wolf, "CNN-N-Gram for handwriting word recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2305–2314.

[16] N. Majid and E. H. Barney Smith, "Boise State Bangla Handwriting Dataset," https://doi.org/10.18122/saipl/1/boisestate, 2018.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, pp. 1137–1149, 2017.

[18] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[19] "Boise State's Research Computing Department. 2017. R2: Dell HPC Intel E5v4 (High Performance Computing Cluster. Boise, ID: Boise State University. DOI: 10.18122/B2S41H.)."