Offline Handwriting Recognition on Devanagari using a new Benchmark Dataset

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

***Abstract*—Handwriting recognition (HWR) in Indic scripts, like Devanagari is very challenging due to the subtleties in the scripts, variations in rendering and the cursive nature of the handwriting. Lack of public handwriting datasets in Indic scripts has long stymied the development of offline handwritten word recognizers and made comparison across different methods a tedious task in the field. In this paper, we release a new handwritten word dataset for Devanagari, IIIT-HW-Dev to alleviate some of these issues.**

We benchmark the IIIT-HW-Dev dataset using a CNN-RNN hybrid architecture. Furthermore, using this architecture, we empirically show that usage of synthetic data and cross lingual transfer learning helps alleviate the issue of lack of training data. We use this proposed pipeline on a public dataset, RoyDB and achieve state of the art results.

***Keywords*-CNN-RNN Hybrid Network, Devanagari Dataset, Handwriting Recognition, Benchmarking.**

I. Introduction

Handwritten text recognition is the process of automatic conversion of handwritten text into machine-encoded text. It has been a popular research area for many years due to various applications such as digitizing handwritten manuscripts [1], postal automation [2], etc. When considering a limited lexicon size or a limited number of writers, robust solutions such as [3] have already been found. However, not enough work has been done in the space of unconstrained text recognition in Indic scripts. In this work, we address the challenges associated with creating a handwritten word recognizer for the most popular Indic script – Devanagari. Devanagari is the most popular Indic script, used by nearly 400 million people in northern India [4]. It is used to write many languages such as Hindi, Sanskrit, etc. It is read from left to right. Figure 1 shows two examples of handwritten Devanagari script.

One of the main reason for the neglect of Devanagari and other Indic scripts in the field of handwriting recognition is the lack of a large publicly available handwriting datasets, which inhibits the training of modern deep learning architectures. These modern architectures contain millions of parameters and require substantial amount of data for training. Another issue caused due to the lack of a publicly available benchmark is that nearly every research paper uses a different dataset, making it impossible to compare any two methods that are present in literature. This contrasts with works in handwriting recognition of English where every method is benchmarked against the IAM [5] dataset. To alleviate the above mentioned issues, we release a new dataset, IIIT-HW-Dev, for handwritten word recognition in Devanagari. We also experiment with various fine-tuning strategies to overcome the lack of real training data.

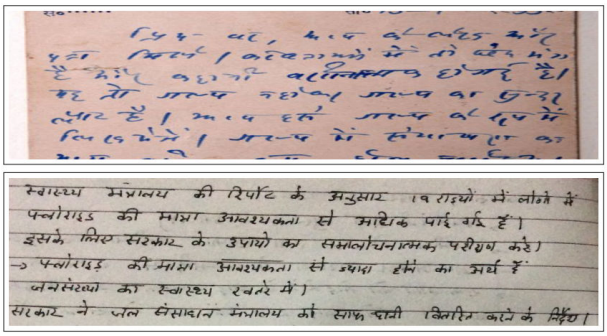

Figure 1. Examples of Handwritten Devanagari from (a) letter written by Munshi Premchand (courtesy Nehru Memorial Museum) and (b) contemporary writing.

A. Issues in Devanagari Script Recognition

The Devanagari script consists of 11 vowels and 37 consonants. A horizontal line called the *Shirorekha* is present in the script from which the characters hang. When multiple characters are written together the *Shirorekha* gets extended. When a consonant is followed by a vowel in Devanagari script, the shape of consonant character gets modified and such a character is referred as a *modified* character or modifier. When a consonant is followed by another consonant and a diacritic called *virama*, a new character gets formed which has an orthographic shape and is called a *compound* character. Unlike Latin scripts, Devanagari script has no concept of lowercase and uppercase letters. For more details about the Devanagari script, the reader can refer to [6].

Beside the aforementioned characters, there are various characters from ancient languages such as Avestan, Sanskrit, etc. that are part of the Devanagari script in order to ensure compatibility of the script with ancient manuscripts. Even if we ignore modifiers and conjuncts, the number of characters in the script is more than double of the number of characters present in English. With the inclusion of modifiers and conjuncts, the number of distinct characters in Devanagari script is well over a thousand [7].

In addition to the script level challenges, there are various issues associated with how the Devanagari script is written, in the case of compound characters. The same compound character can be correctly written in two different ways, one in which the *virama* is hidden and another where the *virama* is explicitly written. Any Devanagari

handwriting recognition system, for all possible compound characters, must map both these variations to the same compound character.

Also, a handwritten word recognizer for Devanagari scripts has to deal with challenges associated with recognizing the varying styles of different writers and the cursive nature of the handwriting. The presence of circular arcs in the character shape causes distortions such as skew while writing along with merging up of adjacent characters.

B. Related Works

Earlier works in the field of Devanagari script recognition were limited to the domain of printed documents. Various methods such as KNN, multi-layer perceptrons were tried out for the recognition task. A summary of these methods can be found in [8]. In this work, we focus on handwritten word images which are more challenging than printed words.

There are three popular ways of building handwritten word recognizers in Devanagari. The first approach consists of segmenting out the various characters and then to use an isolated character (symbol) classifiers like SVM's [9] or lately CNN's [10]. In [11], Roy et al. segment the Devanagari word image into three regions, namely the upper, middle and lower zone, using image processing techniques such as morphological operations and shape matching. The upper and lower zones are recognized by using support vector machines while the middle zone was recognized using a HMM decoded with a lexicon of middle zone characters. Finally, the results from the recognizers in all three zones are combined. The second approach for building recognizers is to use segmentation free methods which train on recognizing the whole word or find a holistic representation [12], [13]. The limitation of both first and second method is that they are limited to recognizing a limited size lexicon.

The third and most popular approach, uses Recurrent Neural Networks (RNN) [14], [15], which defines the input as a sequence of feature vectors. They do not require explicit symbol segmentation for recognition and are not bound to recognizing a limited size lexicon. In this work, we use the above approach, using a CNN-RNN hybrid network and do a sequence to sequence transcription. In this work, we present results in both lexicon-based and lexicon-free (unconstrained) setting. Our proposed method gives the state of the art results on the publicly available Devanagari track of the RoyDB [11] dataset.

The paper is structured as follows: In Section II, we talk about the IIIT-HW-Dev dataset. In Section III, we discuss the CNN-RNN hybrid architecture & its training. In Section IV, we benchmark IIIT-HW-Dev dataset to the CNN-RNN hybrid network, along with discussing the results of the various transfer learning experiments on CNN-RNN hybrid architecture. Section V concludes our work.

II. IIIT-HW-Dev Dataset

In the field of offline handwriting recognition for the Devanagari script, there is a lack of publicly available datasets annotated at the word or line level. Table I lists all the publicly available offline handwriting datasets for all Indic scripts. Here we have ignored datasets that only contain isolated segmented characters or do not have any ground truth data. Clearly, compared to an English dataset like IAM [5], which has more than 100k annotated word images, and written by 600 writers, Indic datasets are much smaller in size. Few of the datasets which are available in Devanagari either use a customized character annotation scheme [11], or only contain images from a small specialized vocabulary, say legal words [16].

To overcome these issues, we have created the IIIT-HW-Dev dataset. This dataset is annotated using UTF-8, which is the dominant character encoding scheme across the web. It contains a vocabulary of 9,540 Devanagari words chosen such that most of the characters present in the dataset are from the UTF-8 Devanagari range. Almost all the words in the vocabulary have the same number of samples in the dataset. On an average, each word in the dataset consists of 8 basic Unicode characters.

A total of 95,381 word samples were collected from 12 different individuals with different educational backgrounds and ages. The writers were free to use pens of their choice and write the words in their natural style. Forms containing well separated boxes were used to collect the data. Each box had a reference word image and space to write the shown word image. The collected data was digitized using a flatbed scanner in a resolution of 600 DPI in color and stored in JPEG format. Using morphological operations, we were able to extract and label the handwritten word images. However, the segmentation was not manually corrected. Figure 2 shows few sample images that were extracted from the dataset.

The train, validation and testing split has been done such that there is no overlap of writers between the 3 sets. The training, validation and testing set are roughly in a ratio of 70:15:15 and contain nearly 70k, 12k & 12k annotated word images respectively. The IIIT-HW-Dev dataset introduced in this research work will be made freely available for academic research purposes.

Table I
Public handwritten document datasets 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 [17]	Bangla	Page	199	21k
	Oriya	Page	140	27k
	Kannada	Page	57	29k
CMATER [18]	Bangla	Line	40	30k
RoyDB [11]	Bangla	Word	60	17k
	Devanagari	Word	60	16k
LAW [16]	Devanagari	Word	10	27k
Tamil-DB [19]	Tamil	Word	50	25k
IIIT-HW-Dev	Devanagari	Word	12	95k

Figure 2. Few sample word images from the IIIT-HW-Dev dataset. The first two rows shows different words that have been written by the same writer. The last two rows shows the same word images that have been written by different writers.

III. Method

In this paper, we use a CNN-RNN Hybrid architecture, first proposed in [20]. Figure 3 illustrates the proposed architecture, which consists of a spatial transformer layer (STN) [21], followed by a set of residual convolutional blocks, which is proceeded by stacked bi-directional LSTM layers and ends with CTC layer for transcribing the labels.

A. CNN-RNN Hybrid Architecture

Our CNN-RNN hybrid architecture consists of a set of convolutional layers, followed by recurrent neural network (BLSTM) layers, whose output is given to a transcription layer, modeled using connectionist temporal classification (CTC) [22]. In general, CNN's have been found to produce excellent spatially discriminative and translation invariant features, while RNN's can perform sequence to sequence transcription on an input of a sequence of feature vectors. Here the input to the RNN is a sequence of feature vectors, constructed from the feature maps of the last convolutional layer by reshaping the 3 dimensional tensor to a 2 dimensional shape. For example, given the feature map of size $\alpha \times \beta \times \gamma$, it is reshaped into a sequence of γ feature vectors, each $\in \mathbb{R}^{\alpha \times \beta}$. Here α, β and γ refer to the the width, height and channel size of the last convolutional layers feature map respectively. This sequence of feature vectors is then forwarded to a stacked set of recurrent layers,here a bi-directional LSTM [14] network. The BLSTM network produces a prediction from the label set at each time step. In our case, the label set consists of all the characters present in UTF-8 for Devanagari, plus a blank symbol. The CTC layer converts the predictions generated by the BLSTM output layer as a maximum probable label sequence for the input. One of the key advantages of the above framework is that the input images need not be resized to a fixed size, thus avoiding distortion in the aspect ratio, since both convolutional and recurrent layers can operate with variable size images and feature sequences respectively.

Recent works in deep learning [23], [24] have shown that adding more convolutional layers to a neural network leads to an improvement in classification accuracy. However, merely increasing the number of convolutional layers can actually cause a decrease in classification accuracy, as shown by [25]. Also upon increasing the number of layers, we encounter the issue of exploding/vanishing gradients and internal covariate shifts [26]. In our architecture, we take into account some of the recent proposed solutions in the literature to overcome these problems. Batch Normalization [26] is applied before each convolutional layer to prevent internal covariate shifts. The convolutional filters are arranged into multiple residual blocks with skip connections as proposed in [25]. Also, as suggested in [27] we apply dropout at the rate of 0.2 at the BLSTM layers in our network.

B. Spatial Transformer Network (STN)

The spatial transformer network [21] is an end-to-end trainable layer which performs an explicit geometric transformation on the input. As shown by [28], this layer transforms the input feature map such that the geometric distortion is removed from the input and the corrected input is forwarded to the network. It has three main components – the localization network, the grid generator and the sampler, as seen in Figure 3. A feature map is given to the localization network as input and it outputs λ, the transformation parameters to be applied to the input feature map. The localization network is a neural network having a $|\lambda|$ dimensional fully connected layer at the end. The grid generator generates a grid of coordinates in the input feature map corresponding to each pixel from the output feature map. Finally, the sampler generates the output feature map using an interpolation scheme after applying the learnt transformation. In the case of handwritten images, the STN layer is useful, as it corrects the distortions caused due to the variable hand movements.

C. Synthetic Data

For successful training of deep learning architecture, availability of huge amounts of training data is crucial, as any typical architecture contains millions of parameters. We follow a pipeline similar to [29], for rendering word images to create the synthetic data. We use nearly 100 publicly available Unicode fonts for Devanagari and used the same vocabulary that was used in creating the IIIT-HW-Dev dataset. The word is rendered using one of the three following ways: with a bottom horizontal line or following a curve or without either of the two distortions. A varying amount of kerning and gaussian noise was applied to the rendered images. We also applied a random amount of rotation (+/- 5 degrees), shearing (+/- 0.5 degrees along the horizontal direction) and translation along all four directions. The synthetic Devanagari train data used in this work consisted of 1 million word images. Figure 4 shows few examples of synthetic word images in Devanagari.

D. Cross Lingual Transfer Learning

Works such as [24], [30] have shown that the filters of lower convolutional layers learn generalized features acting as edge and shape detectors. As we go to the higher layers, the layers learn more specialized features specific to the current task. More recently, [31] reported state of the art word recognition results on the IAM dataset by firstly

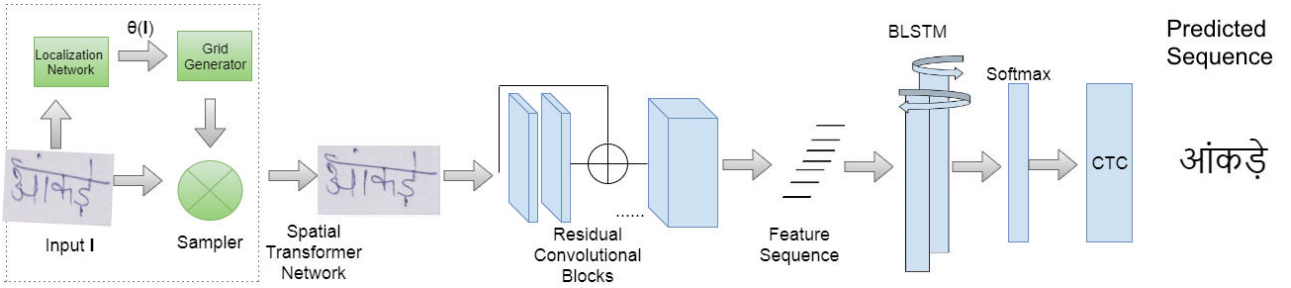

Figure 3. Visualization of the CNN-RNN hybrid network. The various important components of the architecture are highlighted such as the spatial transformer network, residual convolutional blocks, BLSTM layers and the CTC loss function.

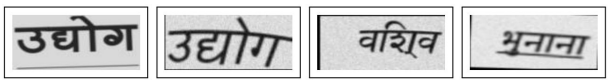

Figure 4. Few word images that were used in the Devanagari synthetic dataset.

training their CRNN network using a dataset that contained Latin languages such as French, English, etc. and Russian, a Slavic language. After training their network to convergence on this mixed dataset, the authors fine-tuned the model for each language separately, by reinitializing its softmax layer from scratch, with its size now only being the size of that particular language's character set plus a blank label. Inspired by these results, we conduct a few experiments to empirically check how the recognition performance is affected by the addition of real data from an unrelated script (here English) along with the addition of data generated synthetically in the language of interest. We try out two different schemes of cross-lingual transfer learning. First, our model is pre-trained on the IAM train set and then fine-tuned on the synthetic data. In the second approach, our model is first pre-trained on the IAM train set. After the model converges, it is then fine-tuned on a mixture dataset of Devanagari synthetic data and IIIT-HW-Dev train set. Finally, in either approach we fine-tune the model on the IIIT-HW-Dev train set. In the second case, during pre-training, the character set includes only English character set plus a blank label. During the subsequent fine tuning, the character set includes all basic Devanagari characters plus a blank label. After each phase of fine-tuning, the softmax layer is reinitialized when the model converges, as appropriate.

IV. Experiments

Across all experiments, we use the word level annotations and images that are part of the various datasets and follow the standard partition for training, validation and testing. We use the Character Error Rate (CER) and Word Error Rate (WER) metrics to compare the various models. CER is defined as (where RT : recognized text and GT: ground truth)

$$CER = \frac{\sum EditDistance(GT, RT)}{\#Characters}$$

and WER is the defined as the number of words wrongly transcribed out of total number of words.

In addition to the IIIT-HW-Dev dataset, we have used the following two public datasets in our work:

- **IAM Handwriting Database [5]:** It includes contributions from 657 writers and comprises of 115,320 words in English. On an average, a word in this dataset consists of 7 characters.
- **Indic Word Database / RoyDB [11]:** It has been compiled by 60 writers and consists of a Bengali and a Devanagari track. The Devanagari track comprises of 16,128 HW word images. On an average, the image in either track represents a word consisting of 4 characters.

A. Architectural Details

The localization network in the STN contains three plain convolutional layers and two fully connected (FC) layers. All the convolutional layers used in the localization network have filter size, stride and padding of 3x3, 1 and 1 respectively. The number of channels in these layers were 64, 64 and 128. 2x2 max pooling is applied before the first convolutional block and after each subsequent convolutional block. The first FC layer is of size 30 while the second FC layer is of size 6, to learn the 6 parameters used in an affine transformation.

The CNN-RNN Hybrid network contains eighteen convolutional layers. All except the first and last convolutional layers have residual skip connections between them. All the convolutional layers used in the hybrid network have a filter size, stride and padding of 3x3, 1 and 1 respectively. The first 5 convolutional layers have a channel size of 64, the next 4 have a channel size of 128, the next 4 have a channel size of 256, the last 5 have a channel size of 512. Max pooling of 2x2 size is applied after the 1st, 5th and 9th convolution layer. Max pooling of 2x1 size is applied after the 13th and 17th convolution layer. The RNN part of the network consists of two BLSTM layers, each having 256 hidden units.

B. Quantitative Results on IIIT-HW-Dev Dataset

Table II shows the recognition results of the various variants of the proposed CNN-RNN hybrid architecture on the IIIT-HW-Dev dataset. The IIIT-HW-Dev train set was augmented using the methodology mentioned in Section III C. All the models mentioned in Table II have the same architecture, except for the first entry. All except the last

entry show results for the unconstrained or lexicon free setting in which decoding is not restricted to any set of chosen words. The various models mentioned in Table II are summarized as follows –

- CNN-RNN uses the architecture mentioned in Section IV.A, without the STN module. It is trained only on the IIIT-HW-Dev train set.
- SCNN-RNN uses the architecture mentioned in Section IV.A (including the STN module). It is trained only on the IIIT-HW-Dev train set.
- Synth-SCNN-RNN uses the same architecture as above. It is first pre-trained on the synthetic data and then fine-tuned on the IIIT-HW-Dev train set.
- IAM-Synth-SCNN-RNN uses the same architecture as above. It is first pre-trained on the IAM train data, then fine-tuned on the synthetic data and finally fine-tuned on the IIIT-HW-Dev train set.
- Mixed-SCNN-RNN uses the same architecture as mentioned above. It is first pre-trained the IAM train set. Then its fine-tuned on a mixture of synthetic data and the IIIT-HW-Dev train set. Finally, its fine-tuned on the IIIT-HW-Dev train set.
- Mixed-SCNN-RNN-Lexicon uses the same architecture and training pipeline as the above model but uses lexicon based decoding.

Table II
WORD RECOGNITION PERFORMANCE OF THE CNN-RNN HYBRID ARCHITECTURE ON THE IIIT-HW-DEV DATASET.

Method	WER	CER
CNN-RNN	45.72	17.56
SCNN-RNN	41.56	15.12
Synth-SCNN-RNN	30.36	10.02
IAM-Synth-SCNN-RNN	28.20	9.75
Mixed-SCNN-RNN	**26.22**	**8.64**
Mixed-SCNN-RNN-Lexicon	**11.27**	**4.90**

From Table II, we can see the effectiveness of the STN module for multi-writer handwriting recognition, as its inclusion leads to a significant reduction in WER. The experiments with synthetic data empirically validate our fine-tuning strategy. Whilst the English to Devanagari cross lingual transfer learning ends up producing only a modest reduction in WER, using both synthetic and IAM data is beneficial in terms of lowering WER.

We also conducted experiments on a publicly available dataset, the Devanagari track of RoyDB [11], using the train, test and validation split as used in [11]. Here we used the Mixed-SCNN-RNN pipeline as above, but instead of using IIIT-HW-Dev data, we used the RoyDB Devanagari train set while training our model. Table III compares

Table III
WORD RECOGNITION PERFORMANCE OF THE CNN-RNN HYBRID ARCHITECTURE ON THE ROYDB DEVANAGARI TEST SET.

Method	WER	CER
Roy et al. – Lexicon [11]	15.76	-
Mixed-SCNN-RNN	**9.57**	**3.24**
Mixed-SCNN-RNN – Lexicon	**4.32**	**2.07**

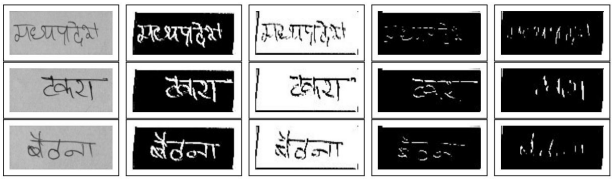

Figure 5. Visualization of activations learnt by the second convolutional layer of the CNN-RNN hybrid network. The first column shows the original input image. The second column shows the activation of a channel which acts a textual edge detector. The third column shows a channel which activates on the image background. The fourth column shows a channel which acts like a horizontal line detector, while the last column shows a channel which acts like a vertical line detector.

Figure 6. Qualitative results of the CNN-RNN Hybrid architecture on the IIIT-HW-Dev dataset. Here GT refers to ground truth.

our results with the one mentioned in [11]. Both our lexicon-free and lexicon-based model hugely outperform the results shown in [11].

C. Qualitative Results

Figures 6 and 7 show the recognized outputs for few sample images from the IIIT-HW-Dev and Devanagari track of RoyDB dataset respectively, using the Mixed-SCNN-RNN network in an unconstrained setting. As we can see, most of the errors were caused by ambiguities in the original handwritten image which is due to the small differences in the shape of basic characters. Also if the image segmentation is incorrect, it is quite easy for the recognition system to misinterpret the *modified* character. To gain further insight into the working of the convolutional layers of the CNN-RNN hybrid network, we visualize the activations of a few channels from the second convolution layer, when certain images are given as input to the network. From Figure 5, we can see that the channels in the second convolutional layer seem to be acting like orientation specific edge detectors.

D. Implementation Details

In all the experiments, the network is trained using the Adadelta [32] optimizer and stochastic gradient descent. We initialize the parameters of the STN to represent the identity transformation. All the input images are resized to 96x256. We observed that doing so does not affect the performance versus using variable length images. We use a batch size of 64 for training.

Figure 7. Qualitative results of the CNN-RNN Hybrid architecture on the Devanagari track of RoyDB dataset. Here GT refers to ground truth.

V. Conclusion

We introduce a new dataset, the IIIT-HW-Dev dataset and benchmark it using a CNN-RNN hybrid network. We show how fine-tuning the network using synthetic and real handwritten data helps improve word recognition. Using our fine-tuning strategy on the CNN-RNN hybrid network gives state of the results on the RoyDB dataset. In the future, we would like to relax the assumption of having pre-segmented images. We also plan to extend this work to include other Indic scripts.

Acknowledgement

This work was partly supported by IMPRINT project, Govt. of India. Praveen Krishnan and Minesh Mathew are supported by TCS Research Scholar Fellowship.

References

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

[2] S. N. Srihari and E. J. Kuebert, "Integration of hand-written address interpretation technology into the united states postal service remote computer reader system," in *DAS*, 1997.

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

[4] G. F. Simons and C. D. Fennig, "Ethnologue: Languages of the world," *SIL International*, 2017.

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

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

[7] U. Stiehl, "Sanskrit-kompendium," *Heidelberg: Hüthing*, 2002.

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

[9] 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.

[10] K. Mehrotra, S. Jetley, A. Deshmukh, and S. Belhe, "Unconstrained handwritten devanagari character recognition using convolutional neural networks," in *4th International Workshop on Multilingual OCR*, 2013.

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

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

[13] 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.

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

[15] N. Sankaran, A. Neelappa, and C. V. Jawahar, "Devanagari text recognition: A transcription based formulation," in *ICDAR*, 2013.

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

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

[18] 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.

[19] 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.

[20] 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.

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

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

[23] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *CVPR*, 2015.

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

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[26] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[27] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour, "Dropout improves recurrent neural networks for handwriting recognition," in *ICFHR*, 2014.

[28] W. Liu, C. Chen, K.-Y. K. Wong, Z. Su, and J. Han, "Star-net: A spatial attention residue network for scene text recognition." in *BMVC*, 2016.

[29] P. Krishnan and C. V. Jawahar, "Generating synthetic data for text recognition," *arXiv preprint arXiv:1608.04224*, 2016.

[30] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *ECCV*, 2014.

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

[32] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.