Can RNNs Reliably Separate Script and Language at Word and Line Level?

by

Ajeet Kumar Singh, C V Jawahar

in

International Conference on Document Analysis and Recognition,
(*ICDAR 2015*)

Report No: IIIT/TR/2015/-1

Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
August 2015

Can RNNs Reliably Separate Script and Language at Word and Line Level?

Ajeet Kumar Singh and C. V. Jawahar
Center for Visual Information Technology, IIIT Hyderabad, India.

Abstract—**In this work, we investigate the utility of Recurrent Neural Networks (RNNs) for script and langauge identification. Both these problems have been attempted in the past with representations computed from the distribution of connected components or characters (e.g. texture, n-gram). Often these features are computed from a larger segment (a paragraph or a page). We argue that one can predict the script or language with minimal evidence (e.g. given only a word or a line) very accurately with the help of a pre-trained RNN. We propose a simple and generic solution for the task of script and language identification which do not require any special tuning. Our method represents the word images as a sequence of feature vectors, and employ the RNNs for the identification. We verify the method on a large corpus of more than 15.03M words from 55K document images comprising 15 scripts and languages. We report an accurate script and language identification at word and line level.**

Keywords*—*Script and Language Identification, Recurrent Neural Networks, Recognition Free Methods

I. Introduction

Recurrent Neural Networks (RNNs) have gained popularity in recent years for many recognition tasks such as Optical Character Recognition(OCR) [1, 2], handwriting recognition [3], word retrieval [4], and word spotting [5]. This architecture has started finding more and more applications in diverse areas of computer vision [6]. In this work, we investigate the utility of RNNs for script and language identification at the granularity of words and lines. Naturally, this investigation has its applications in multilingual settings, where one needs to decide the script or language before recognition or post-processing of an incoming document image. In addition, this work also throws light on how semantically richer tasks can be attempted without any explicit recognition by looking at the distribution of certain features. For example, can we find the topic model or classify the document into an appropriate category without an explicit textual representation? Such high-level tasks are often attempted based on the statistics and distribution of words or characters. In this work, we limit our attention to the identification of script and language at the word and line level as shown in Fig. 1. By designing an RNN that can learn the distribution of feature vectors, we reliably identify the scripts and language from the image itself. We empirically demonstrate the utility of RNNs for script and language identification with experiments on a corpus of nearly 15.03M words from 55K document images comprising 15 scripts and languages. We argue that RNNs can be considered as a strong contender for accurate script and language identification which can lead us to the integrated solutions for the recognition tasks in multilingual settings.

Many approaches proposed in the past for script and language identification often deal at page, line or word level. At page level, script is identified by looking at the texture and orientation of the image segments. Sptiz [7] analylzed the individual components for script identification in document images using attributes such as upward concavities, optical densities, character height densities and top and bottom profiles. The use of texture has also been extensively used in script identification. Tan [8] proposed to solve this problem using a multi-channel Gabor filter. Many later attempts used different variations of texture features computed from Gray-level Co-occurrence Matrix, Gabor Energy, Wavelet Energy, Local Binary Pattern [9, 10, 11, 12] for the identification purpose. In recent years, there has been an effort to use discriminative features learned using Convolutional Neural Networks(CNN) for multi-script recognition [13]. These features are automatically extracted and learned at connected component level of the document image.

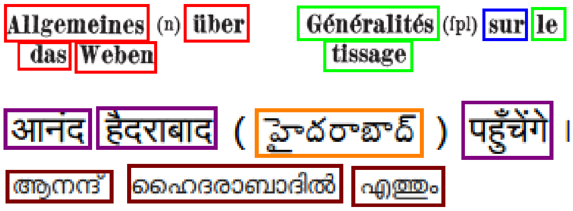
Allgemeines (n) über das Weben
Généralités (fpl) sur le tissage
आनंद हैदराबाद (హైదరాబాద్) पहुँचेंगे ।
ആനന്ദ് ഹൈദരാബാദിൽ എത്തും

Fig. 1. Figure depicts the script and language identified at word level in document snippets written in Roman-script (first row) based languages and Indic scripts (second row), respectively. In the first row, red, green and blue rectangles denote German, French and Spanish languages, respectively. In the second row, violet, orange and brown rectangles denote Hindi, Telugu and Malayalam scripts, respectively. Unlike the approaches in the past we propose a method to identify the script and language at word and line level by employing popular Recurrent Neural Network (RNNs).

When the inherent script of document images are same, visual features are hard to separate between different languages, especially when the identification is required at word level. There are many attempts in the textual domain to separate the languages. Often they use the statistics (*e.g.* n-gram probabilities of characters). In the image domain, language identification is attempted at page level or paragraph level in the past. A class of methods have been proposed which categorizes the characters based on a number of character shape features such as character ascenders and descenders. For example, [7] group the character images into a small set of categories first. Then, based on the classification results, each word image is converted into a word shape token. Latin-based languages are finally determined according to the frequency of a single word [7], word pair and word trigram. Shijian and Tan [14] combined the script and language identification using a document vectorization framework. They convert document image into a vertical cut vector based on the number and positions of vertical cut to capture the shape of the word directly.

Our method is simple, efficient and accurate, without any special tuning for the scripts or languages of interest. We convert the word or line images into a sequence of feature vectors and train the RNNs to reliably separate the script or language. We report comparable, if not better results than the state-of-the-art [11]. Our method also leaves lots of scope for further improvement in performance with better features and special adjustments (e.g. hierarchical classification, special features for harder pairs). We believe this makes our method very generic and applicable in a wide range of settings. We perform the experiments on 12 Indic scripts: Hindi, Malayalam, Gurumukhi, Kannada, Tamil, Telugu, Bangla, Marathi, Gujarati, Assamese, Manipuri and Odiya, and 3 Roman script based languages: French, German and Spanish. We discuss the method in Section II and the experimental results in Section III.

II. RNN for Script and Language Identification

In traditional feed-forward neural networks (FFNN), connections between the nodes do not form any cycles. If we relax this condition, and allow the cyclical connections among the nodes, we obtain the *recurrent neural networks* (RNNs). RNNs in the past have been used to handle sequential data. RNN is a powerful classification tool, as it allows a "memory" regarding previous inputs to persist in network's internal state, which can be later used to influence the network output. RNNs are not widely popular, as they often require a longer training process, because the error path integral decays exponentially along the sequence [15]. Our preference for RNNs is motivated by the fact that it has superior characteristics in several aspects. Unlike HMM which uses the current state of input to generate any observations, RNN uses the long-short term memory (LSTM [3]) structure to store the contextual information of previous states. Also it does not require any explicit labeling of all the vectors in the input feature sequences.

For the script and language identification, we use a RNN based Bidirectional Long Short Term Memory(BLSTM) network. These networks have been used in the past for printed text [2] and handwritten text recognition [3]. This network consists of two LSTM networks in which one network takes the input from beginning to end while other network takes the input from end to beginning. The individual output of both the LSTM networks is used to predict the final output. Hence, these networks have been known for remembering the long range of context over several timesteps. The Connectionist Temporal Classification (CTC) [16] is used at the output layer of RNN network to label the unsegmented data which uses a forward-backward algorithm. The CTC [16] layer directly outputs the probability distribution of desired label. The output layer of RNN network contains one node for each class label plus a special node, (ϵ), which indicates "No Label", i.e. no decision can be made about the incoming word/line at that position. Hence, there are $\mathcal{K}+1$ nodes in the output layer, where $\mathcal{K}$ is the number of class labels. In our system, a training sample can be viewed as a pair of input sequential features and target script/language label (x, z). The objective function of RNN is then defined by:

$$\mathcal{O} = -\sum_{(x,z)\in\mathcal{S}} \ln p(z|x), \tag{1}$$

where $\mathcal{S}$ denotes the training set and $p(z|x)$ denotes the conditional probability of label z given a sequence of feature x. The main objective is to minimize $\mathcal{O}$, which is equivalent to maximization of conditional probability $p(z|x)$. For script and language identification, our method only uses the script/language level annotation.

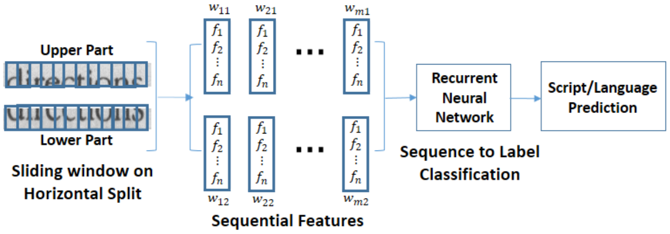

Fig. 2. The architecture for RNN based script and language identification. From left to right, the segmented line and word from the document images are horizontally divided into two parts. Then, sequence features are calculated from sliding windows, w. Here, m is the number of sliding windows and n is the number of features , f, computed from a single window. These features are then given as input to the LSTM cell of RNN to identify the script and language of current line/word image.

We also analyzed the network performance on various parameter settings for our identification task. A RNN is characterized by the number of nodes in hidden layer it uses, number of hidden layers and the stopping criteria used for training. We generally stop the RNN training once the training error rate ceased to reduce below a certain threshold. We have observed, experimentally, that increasing the number of hidden layers until 3 gave better results. The best results are obtained with the LSTM size of 50 with 3 hidden layers.

A. *Representation of Words and Lines*

In order to use the RNN, the input word and line images are needed to be converted into sequential features. For this, we use the popular profile features [2, 17], which can be used to represent the lines and words as a feature sequence. In this work, we calculate six profile features from every word and image. These features are calculated using the sliding windows of size 20 pixels with an overlap of 75%. For each window, scanning is done from top to bottom and following four features are computed: (F1) vertical profile(i.e. the number of ink pixels in each column), (F2) location of uppermost ink pixel, (F3) location of lowermost ink pixel and (F4) number of ink to background transitions. The profile features are calculated on binarized word/line images obtained using the Otsu thresholding algorithm. We also use the gray level information of the image to extract two features: (F5) mean value and (F6) standard deviation of gray pixel values. All features are normalized with respect to the image height to [0,1]. These features are made more robust by horizontally dividing the image into two regions and then computing the aforementioned features for each region. Hence, we extract a total of twelve features. The splitting of the image into two parts may seem

Scripts/Languages	D1[18]				Accuracy (in %)			
					D1-[18]		D2-[11]	
	Books	Pages	Lines	Words	Line	Word	Ours	Pati[11]
Hindi	34	5K	133K	1.66M	96.6	85.8	92.3	96.2
Malayalam	31	5K	93K	0.96M	99.2	99.0	96.2	93.3
Gurumukhi	33	5K	125K	1.62M	97.9	93.2	92.8	93.6
Kannada	27	3.8K	90K	0.72M	98.0	93.8	93	93.8
Tamil	23	4.8K	88K	0.64M	98.5	98.1	95.9	95.2
Telugu	28	5K	102K	0.83M	98.4	96.0	91.5	92.3
Bangla	14	2.8K	50K	0.95M	98.6	98.5	94.3	96.2
Marathi	20	5K	127K	1.44M	97.6	95.8	-	-
Gujarati	26	5.2K	124K	1.25M	98.6	98.4	94.5	95.5
Assamese	19	3.5K	73K	0.59M	95.3	93.3	-	-
Manipuri	25	3.6K	69K	0.72M	98.2	71.4	-	-
Odiya	17	5K	109K	1.44M	99.5	97.2	96.4	94

TABLE I. TABLE DEPICTS THE DETAILS OF DATASET (D1) [18] USED FOR SCRIPT AND LANGUAGE IDENTIFICATION. IT DEPICTS THE PERFORMANCE OF OUR METHOD ON THE D1 AT WORD AND LINE LEVEL. IT ALSO SHOWS THE COMPARISON OF OUR METHOD AGAINST GABOR FEATURES WITH SVM CLASSIFIER ON D2 [11]. SINCE, D2 [11] DIDN'T SHOW ANY RESULTS ON MARATHI, ASSAMESE AND MANIPURI SCRIPTS, WE ARE NOT COMPARING ON THESE LANGUAGES.

insignificant, but it helps in differentiating similar symbols which appear in different areas. Fig. 2 shows the full pipeline for script and language identification, depicting the various stages of identification framework from feature representation to identification using RNN.

B. Implementation and Evaluation

The script and language of a line or word image is identified by presenting the corresponding sequential features to the RNN. For this, we train integrated neural networks for both scripts and languages for identification at word and line level. For training the RNN, the initial parameters, number of hidden nodes, number of hidden layers are obtained by cross-validation. Number of input nodes in network is equal to the number of features presented to it (12 in our case) and number of output nodes is same as the number of target labels (in our case 12 nodes for script and 3 nodes for language). For all the experiments, we have used a LSTM size of 50 and number of hidden layers is set at 3. All these experiments were conducted on a mid-level desktop PC having 16GB RAM and a 2.3GHZ processor. On an average, training was conducted for 50 epochs for all the experiments mentioned below. The implementation details specific to script and language identification are explained in detail in sections III-A and III-B, respectively.

III. RESULTS AND DISCUSSIONS

In this section, we validate our method on a spectrum of scripts and languages. We present the experimental results of our proposed script and language identification method at both word and line levels.

A. Script identification

We have tested the proposed method on 12 different scripts of Indian multilingual dataset. For this evaluation we have taken around as many as 5000 pages and as few as 2800 pages from each script, amounting to 50K pages and a total of 12.84M words. This dataset has emerged as a challenging benchmark data (D1) [18] within Indian OCR research community. Almost all of these scripts and languages have their own unique way of representing the character symbols. For example, the scripts of the languages such as Hindi, Bangla and Gurumukhi uses *shirorekha* (headline) over its words while the languages such as Malayalam, Tamil, Telugu and Kannada are more curved in nature. Table I shows the details of the printed dataset which we have used for our experiments.

To train the RNN for word level script identification we use 960K words and 240K words for validation from all the scripts. Training the network for word level script identification took an average of 3.75 hours per epoch. The trained network is then tested on 11.64M words. It took *0.1 ms* to identify the script of a word. At line level too, a separate network (with same architecture) is trained with 120K lines followed by validation with 60K lines from all the scripts . Training time in this task took an average of 4.11 hours per epoch. The trained network is then tested on 1.003M lines. Script identification of a single unseen line took *0.5 ms*. Note that as the average length of input sequence increases, training the network becomes costly with respect to time

We have performed the experiments at line and word level on reported dataset (D1). Table I shows the accuracy of our script identification method at line and word level for all the scripts. At word level script identification, our method achieves an average accuracy of 93.96% and at line level, we report an average accuracy of 97.90%. At word level, we report a maximum accuracy of 99% for Malayalam script. And for Manipuri, report a minimum accuracy of 71.4% due to presence of visually similar characters in the script from Assamese script. Similarly, at line level we are getting a maximum accuracy of 99.5% for Odiya and a minimum accuracy of 95.3% for Assamese.

In order to validate the generality of the work, we compare it with the method proposed in [11] on the reported dataset (D2). Their word image dataset (D2) contains about 220K words from eleven different Indian scripts. The method in [11] uses Gabor features with SVM as classifier to identify the scripts of the incoming word images. We train the RNN with 7K words and test it with remaining 13K words from each script. In Table I we report the accuracy of both these methods on this dataset (D2). Both the method yield comparable results (i.e., 94.59% of our method against 94.8% of [11]). As can be seen, our method which uses naive features yield results that are comparable to those that are evolved over years of research. (Note that wide variety of texture features based on gabor and wavelets are tried in the past [11, 9, 19] and this was one of the top performing descriptors in this class.) In addition, our method uses a simple multiclass classifier and not a hierarchical handcrafted classifier architecture as in [11]. It can also be seen from the Table I, our method on D2 gives an accuracy 94.10% whereas [11] reported an accuracy of 94.44%. One may also note that D2 does not contain scripts (such as Manipuri, Assamese and Marathi) which can get confused with others present in the dataset. On this subset of scripts, we report an average performance of 95.55%, which is better than 94.44% as reported by [11].

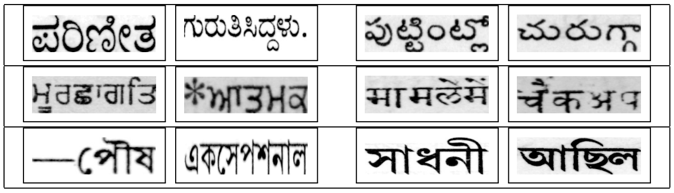

Fig. 3. Script identification Results: Some failure cases in script identification at word level. First row, first column shows Kannada words identified as Telugu and the second column in same row shows Telugu words identified as Kannada words. In second row, first column shows the Gurumukhi words as Hindi and in second column of the same row, Hindi words identified as Gurumukhi. Similarly in the third row of the figure, first column shows Bangla words identified as Assamese and vice versa in second column.

Using multilayer perceptrons (MLP) for script identification at word level, yields an average baseline accuracy of 74.67% against our method's 93.96%.

Scripts in Indic languages share some minor or major similarities with each other. For example, Hindi, Bangla and Gurumukhi have *shirorekha* at top of their wordset. Therefore there is a probability that a Hindi word can be confused with a Gurumukhi or a Bangla word, which also holds true for Gurumukhi and Bangla words. Also, there are some characters in these scripts which are visually similar. Fig. 4 shows the confusion matrix at word level for all the 12 Indian multilingual scripts. It is evident that aforementioned observation holds true as Hindi is confused with Bangla and Gurumukhi 1.46% and 0.91% of words, respectively. Similarly, Gurumukhi words are identified as Hindi as much as 2% of times and Bangla 1.7%. In confusion matrix table it can also be seen that 2% of Kannada words are being confused with Telugu words and 1.81% of Telugu words are confused as Kannada words. This is due to the fact that Kannada and Telugu alphabets are essentially the regional calligraphic variants of a single script. Assamese, Oriya and Bangla also look similar as they all originated from an ancient Siddhông script. Typographical differences between these scripts are used to identify the alphabets and their script. Hence, it can be seen in the confusion matrix that Assamese words are identified as Bangla 0.51% of the times, and as Oriya 0.54% times. Similarly, Oriya words are identified as Assamese 0.95% times. Some failure cases in script identification at word level has been discussed in Fig. 3, where we show the effect of observations made above, on identification at word level.

For line level script identification, we report better results than the identification at word level. We observed through our experiments that longer sequential features, even at word level, gives a good accuracy. Hence, it is natural that the accuracy at line level will be better than the word level accuracy as RNN becomes more confident with longer sequences. Also, we find that the assumptions which we made above, at word level hold true for the line level too. Although, the accuracies of Hindi, Bangla and Gurumukhi has increased at line level, it is observed that there are still some confusions, albeit low, among these due to their textual properties. For Kannada and Telugu too, there are some confusion due to similarity of the scripts they are written in.

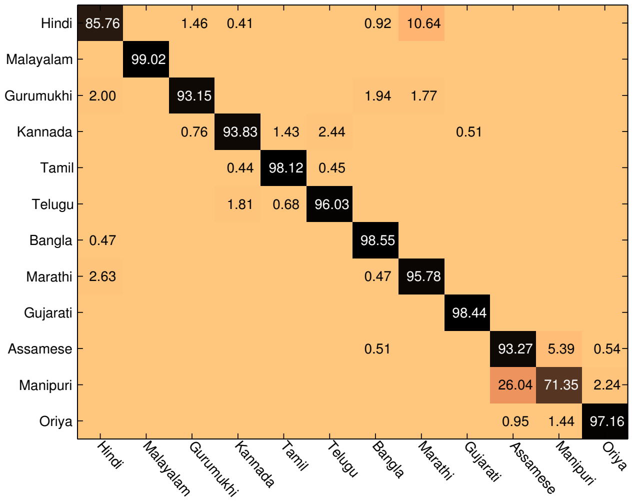

Fig. 4. Confusion Matrix for the script identification at word level. The blank spaces in the graph denotes predictions that are less than 0.40%.

B. Language Identification

Encouraged by the performance of the method on script identification, we also did experiments to identify the inherent language of a document image at line as well as word level. For this we use three Roman script based languages: French, German and Spanish; two Devanagari script based languages: Hindi and Marathi; two Bangla script based languages: Assamese and Manipuri which also happens to share some vocabulary. Table II shows the printed dataset details for Roman-based languages. We have used around 2000 pages for each language, amounting to 2.19M words and 154K lines.

For language identification at word level, we train the RNN (with the same architecture as mentioned in section III-A) with 600K words followed by validation with 150K words from all Roman-script based languages. Training took approximately 2 hours per epoch. For testing, around 1M words were used. To identify the language of a word, it took an average of *0.1 ms*. For language identification at line level, we train and validate a different network with same architecture with 30K lines 15K lines, respectively, from all the languages. Training took an average of 0.8 hours per epoch. Trained network is then tested with 100K lines.

For language identification, we are showing the accuracy of all the Roman-script based languages in Table II. We achieve an average accuracy of 93.39% at word level on our dataset. In Table II we also show the confusion matrix for language identification for the languages at word level. For language identification at line level, the average line level accuracy is shown in also shown in Table II. Using RNN, we achieve an average accuracy of 95.25% for language identification at line level.

We also perform language identification at word and line level on some Indian languages that share script and vocabulary. We achieve a fairly good accuracy for all these languages. As it can be seen in Table I, Hindi and Marathi, which share Devanagiri script, obtain an accuracy of 85% and 95.8% respectively. Assamese and Manipuri, which share

Language	Dataset				Confusion Matrix (%)			Accuracy (%)	
	Books	Pages	Lines	Words	French	German	Spanish	Line	Word
French	6	1.9K	51K	0.71M	93.32	3.47	3.21	94.51	93.32
German	4	2.1K	55K	0.74M	5.44	92.19	2.37	94.77	92.19
Spanish	5	1.9K	48K	0.63M	3.63	1.70	94.67	96.47	94.67

TABLE II. Table depicts the Roman script-based dataset used for language identification. It shows the confusion matrix for language identification for Roman-script dataset. It also depicts the performance of our method on the reported dataset at word and line level.

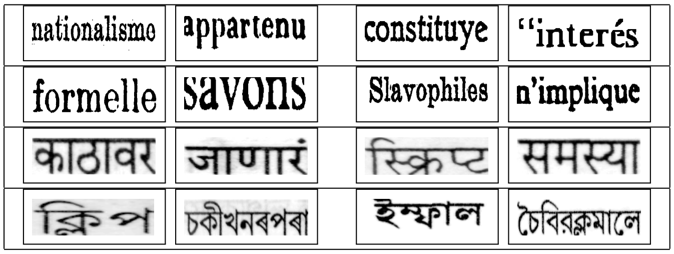

Fig. 5. Language Identification Results: Some failure cases for language identification at word level for both the Indian and Roman-script based dataset. In the first row, the first column shows the French words identified as Spanish and the second column shows Spanish words identified as French. In the second row, the first column shows the German words identified as French and the second ones shows French words identified as German. For the third row, the first column shows the Marathi words identified as Hindi, and vice versa in second column. In the fourth row, the first column shows the Assamese words identified as Manipuri and vice versa in the second column.

both script and vocabulary, obtain an accuracy of 93.27% and 71.4% respectively. At the line level too, the Indian languages perform better than that at word level due to longer sequential features. Table I shows the accuracy at line level for the Indian languages sharing the script as well as some vocabulary.

Lexical similarity is a measure of the degree to which the word sets of two given languages are similar. A lexical similarity of 1 means the complete overlap between vocabularies whereas 0 means no overlap. Lexical similarity of French and German is 0.29, hence, there are 29% common words in French and German language, similarly French and Spanish has 75% of vocabulary overlapping (more information can be found at [20]). Therefore, it is evident in Table II that 3.63% of Spanish words are confused with French and 3.21% of French words as Spanish. Similarly, 3.47% of French words are confused as German and 5.44% of German words as French. In Indian languages, Hindi and Marathi share a common script of Devanagiri. Hence, it can be seen in the confusion matrix in Fig. 4 that 2.64% of Marathi language words are confused with Hindi words. And 10% of Hindi words are confused as Marathi. In Fig. 4, it can also be seen that 26% of Manipuri words are confused as Assamese words, and 5.7% of Assamese words are confused as Manipuri words. The failure cases in language identification at word level for both Indian languages and Roman-script based languages are shown in Fig. 5.

IV. Conclusion

Script and language identification in multilingual setting is very important in optical character recognition tasks. In this work, we present a simple, efficient and accurate method to predict the inherent script or language at word and line level with minimal evidence, using a pre-trained RNN. This work comprises of two important components. First component computes the sequential feature from an unsegmented word image. The second component, LSTM is used to classify the incoming word image into its corresponding scripts and languages. Also, experiments on a public dataset show that even with naive features, RNNs achieves good if not better results than the state-of-the-art method. We also observe that, RNNs are able to characterize the statistical distribution of features computed over vertical segments. We hope that this can help in other forms of recognition free tasks in document image understanding.

Acknowledgements. This work is supported by Ministry of Communication and Information Technology, Government of India, New Delhi.

References

[1] T. Breuel, A. Ul-Hasan, M. Al-Azawi, and F. Shafait, "High-performance ocr for printed english and fraktur using lstm networks," in *ICDAR*, 2013.

[2] P. Krishnan, N. Sankaran, A. K. Singh, and C. V. Jawahar, "Towards a robust ocr system for indic scripts," in *DAS*, 2014.

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

[4] R. Jain, V. Frinken, C. V. Jawahar, and R. Manmatha, "Blstm neural network based word retrieval for hindi documents," in *ICDAR*, 2011.

[5] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, "A novel word spotting method based on recurrent neural networks," *PAMI*, 2012.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[7] A. Spitz, "Determination of the script and language content of document images," *PAMI*, 1997.

[8] T. Tan, "Rotation invariant texture features and their use in automatic script identification," *PAMI*, 1998.

[9] A. Busch, W. Boles, and S. Sridharan, "Texture for script identification," *PAMI*, 2005.

[10] M. Ferrer, A. Morales, and U. Pal, "Lbp based line-wise script identification," in *ICDAR*, 2013.

[11] P. B. Pati and A. G. Ramakrishnan, "Word level multi-script identification," *PR Letters*, 2008.

[12] S. Chanda, S. Pal, K. Franke, and U. Pal, "Two-stage approach for word-wise script identification," in *ICDAR*, 2009.

[13] S. Rashid, F. Shafait, and T. Breuel, "Discriminative learning for script recognition," in *ICIP*, Sept 2010.

[14] L. Shijian and C. Tan, "Script and language identification in noisy and degraded document images," *PAMI*, 2008.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computing*, 1997.

[16] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *ICML*, 2006.

[17] T. M. Rath and R. Manmatha, "Features for word spotting in historical manuscripts," in *ICDAR*, 2003.

[18] C. V. Jawahar and A. Kumar, "Content-level Annotation of Large Collection of Printed Document Images," in *ICDAR*, 2007.

[19] G. D. Joshi, S. Garg, and J. Sivaswamy, "A generalised framework for script identification." *IJDAR*, 2007.

[20] "Ethnologue - webpage," https://www.ethnologue.com/.