Recurrent Neural Networks based Indic word-wise script identification using character-wise training

Anonymous ACPR submission

Paper ID 190

Abstract

We present here a novel methodology of Indic handwritten script recognition using Recurrent Neural Networks [5]. Online character level data is used to train to RNNs using BLSTM architecture, modeled initially for two scripts, thereafter modeled for 5 scripts. The character level trained networks are tested on character level data as well as word level data and performance is reported.

We implement a version of stroke recovery to overcome the shortfall of directly implementing rnnlib on raw offline data as it lacks the temporal information available in online data. This substantially improves prediction results when paired with the online data trained networks. We further test on the character and word level in the case of offline data.

1. Introduction

Handwritten script recognition constitutes the problem of identification of the script used in a particular text document. The basic distinguishing factor between scripts is the unique spatial relationship that strokes of a particular script have with each other. Since most handwritten text recognition systems are language dependent, handwriting script recognition plays an important role in interpretation of handwritten text documents. Recurrent Neural Networks (RNNs) have proven to be very effective [12] for handwriting recognition tasks. They have successfully been applied for classification in tasks involving temporal data. The RNN architecture used to train our models is Bidirectional Long Short Term memory (BLSTM). BiDirectional RNNs enjoy the added benefit of using the past and the future contexts for classification. For training RNNs we use rnnlib [5]

To train a classification model, the document input can be presented as offline data and online data. The offline document is a sequence of images, presented to the system. At the character level, these are images of characters of the script and similarly, at the word level these are images of words of the script. The online data comprises the trace or the strokes of the pen on a recording screen as a designated writer writes separate characters for character level and words for word level data. While working with online data for script recognition, the features possess spatio-temporal information which proves to be beneficial for recognition task. The Character level data is much smaller than the word level data provides the benefit of faster training. In this paper, we stress on the models trained using just the character level data and test them for word level data as well.

Training of offline data demands larger datasets and training time vis-a-vis that of online data. A Stroke Recovery method is one which takes as input a text image and outputs the possible order of strokes that are required to write the character or word, in other words, it extracts the temporal information of the handwritten data. We have developed a simple method of Stroke Recovery mechanism for offline script recognition. The output of the system is a possible sequence of pen strokes and each of these strokes is the sequence of pixels that the image covers.

In the second section of this paper, we discuss recent works in script recognition and stroke recovery. The third section describes the methodology followed by us. The fourth section details the datasets we used and the experiments we conducted. The last section covers conclusions and future works followed by references.

2. Previous Work

A review of methods used in script recognition was presented by Ghosh *et al.*, [4]. An important application of script recognition that it discusses is recognition for documents written in more than a single language. The work by Moalla *et al.* [15, 14] for extraction of Arabic text from documents containing Arabic and English words is along these lines. In [15], the method matches a trained Arabic template dataset with characters in the test set. and in [14] the method was based on feature (morphological and statistical) matching.

A popular approach when working with offline data is to extract the pen strokes for better recognition. In [3], Elbaati *et al.* proposed an approach of stroke recovery by first segmenting the image into strokes and labeling all the edges as segments or parts of strokes, then running a Genetic Algorithm to optimize these strokes and produce the best possible segment order. An application of the above developed method is seen in [6], which combines Offline and Online data for recognition. In [9], a statistical method is proposed, dividing the stroke recovery into two phases. In the first phase, a Universal Writing Model is trained and in the second (Test) phase, the UWM is tested on the offline images to recover the Strokes.

In [8], Kato *et al.* propose a stroke recovery technique which works for single stroke characters. The system labels each edge in the image and bridges them in an algorithmic manner without the use of any Learning Methods. Our approach is similar in its approach to [8], though we extend the system to account for multiple strokes in the input image.

3. Methodology

3.1. BLSTM

BLSTM is a recurrent neural network architecture which provides a memory of the previous network internal state using LSTM hidden layers.

3.1.1 Long Short-Term memory (LSTM) layer

The LSTM network nodes have a specific architecture, referred to as a memory block. Each memory block contains a memory cell, and its interaction with the rest of the network is controlled by three gates, namely: an input gate, an output gate and a forget gate. This allows the memory cell to preserve its state over a long range of time and to model the context at the feature level. The 1D sequence recognition is improved by processing the input signal in both directions, i.e., one layer processes the signal in forward direction while another layer processes it in backward direction. The output of both layers is combined at the next layer as a feature map. Similar to convolution neural network architecture, it is possible to have multiple forwards and backward layers in each LSTM layer as well as multiple feature maps at the output layer, and to stack multiple LSTM layers using max-pooling sub-sampling.

3.1.2 Cross Entropy Error

The Cross Entropy Error forms the objective function optimized by the neural network. In a binary network, the error E is:

$$E = -\sum_{(x,z)\epsilon S} z \ln y + (1-z)\ln(1-y) \tag{1}$$

Where z is the target symbol (0 or 1), and y can be understood as the probability (2) that the input belongs to a class. Details can be found in [12].

$$p(C_1|x) = y$$
$$p(C_2|x) = 1 - y \tag{2}$$

When extended to multi-class networks with k classes, the error function E is:

$$E = -\sum_{(x,z)\epsilon S} \sum_{k=1}^{K} z_k \ln y_k \tag{3}$$

Similarly, Extending (2) for multiple classes.

$$p(z|x) = \prod_{k=1}^{K} y_k^{z_k} \tag{4}$$

3.2. Character-wise training for Word-wise identification

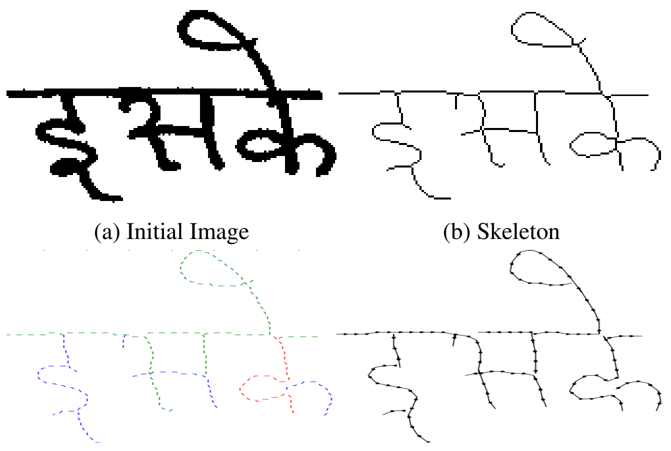
3.2.1 Online character based approach

We trained the first online character level data network for Hindi and English. For each character in the online data, we had a sequence of strokes. We appended the data together, as a series of vectors, each vector with three values. The first value is the x coordinate of the pixel, the second is the y coordinate and the third value is 1 or 0, depending on whether the pixel is the first pixel of a stroke, or not. Most of the datasets were obtained from different sources and recording devices. This prompted a need for a Normalization Technique for the vectors in the training data, to eliminate features specific to the dataset and retain only features of the script. The following technique is used:

$$Normalized = \frac{array(vector) - Means}{StdDeviation}$$

Where vector is the array of three values and the Means and the Std Deviation are the mean and the standard deviation of all the vectors in the particular dataset. The system was extended to 5 languages. The Bengali, Tamil and Telugu Languages were added to the training.

Using word level online data, first we tested the word level dataset on the character level trained models, which is lighter and faster to train. Next, we trained a separate word level model and the results are reported.


(a) Initial Image (b) Skeleton

(c) Recovered Strokes (d) Stroke directions

Figure 1: Stroke recovery steps. In (c), different colors depict independent strokes, (d) shows stroke directions

3.2.2 Offline character based approach

The offline model we trained was based on the simplistic raw pixel data based. We trained for Hindi, Bengali, Tamil and English and normalized the datasets by resizing the height of the character image to a constant value, keeping the aspect ratio the same as that of the original image. The stroke width variation is removed using thinning methods followed by thickening the strokes to a constant width. The data obtained was fed for training the model. This approach was limited to the character level as it was very data intensive and slow to train. For classification of word images, we developed a mechanism for stroke recovery.

3.3. Offline to Online conversion

In order to improve upon the accuracy for the classification task on offline data we implement a simple form of stroke recovery. lighter to train The first step for stroke recovery was obtaining the skeleton of the provided image dataset. We used Fiji [7] to extract the skeleton from the binary input images. The main module for stroke recovery then calculates the endpoints and the junction points in the given binary figure. The Endpoints are those points in the image that are connected to only one other point. The junction points are the ones connected to more than two points, i.e. they lie on at least two strokes.

Our method reduces each Junction point and its neighboring junction points into one, joint junction area, by appending to each junction point in a neighborhood all the collective outlets of the neighboring Junction points (figure 2). We utilize a simplistic approach to select the start point for stroke recovery, we start with the endpoint closest to the top most corner of the work screen. This is based on the insight that the scripts we are working on are written in the left to right order and the strokes begin from the top most corner. However, this method does result in poor stroke recovery at times (figure 4).

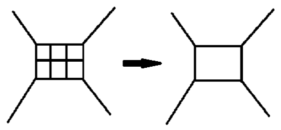


Figure 2: Neighboring Cluster of Junction Points, converted to a joint junction Point

In case of multiple and disjoint strokes, we need to constantly select a new start point to cover all the points/strokes. We experimented with two implementations, the first is to run through all the endpoints first as the respective starts and then start with all the neighbors of the junction points that have been covered, followed by the junction points that have not been taken as a part of a stroke. The second approach is to give the Junction Points and their neighbors the priority and this approach outperforms the first.

It is important that a stroke maintains its continuity and avoids jerks. Whenever a stroke reaches a junction point, among the outlets, it should continue along that outlet that best maintains the continuity. We developed a slope based selection method to choose among the outlets by estimating the slopes of the exiting curves and select the outgoing curve that has the slope closest to that of the incoming curve.

3.4. Ensemble Approach

We developed an ensemble approach to club together predictions from two or more trained models. Two RNN instances trained using the same data might converge to different local minima. This module attempts to take advantage of this and club together results to lead to improvement. As input, this module takes predictions of two networks on the same test set along with cross entropy error for each prediction.It then calculates the associated confidence of the prediction by each of the networks and based on the higher confidence provides the final output. This module can be used to club online and offline results.

4. Datasets and Experiments

4.1. Datasets

The datasets used by us are from a variety of sources. For the English language, we obtained Word Images from IAM Dataset [13], online word data from IAM Online [11], character level offline and online data, we used Chars74K [2]. For Hindi, Tamil and Telugu languages, the online and offline character/word data was obtained from HP Labs. and Hindi word images from [1]. For Bengali script, the word

Table 1: Results on Test sets of On-line data

Eng-Hin Model	Char Model	Word Model
Char Level Test	99.75%	-
Word Level Test	100%	100%
Many language Models	5 Language	4 Language
5-Language Char Test	99.74%	-
4-Language Word Test	99.82%	99.7%

images came from CMATERdb, [1]. The Assamese online character data came from [10].

4.2. Experiments

4.2.1 Script identification - Online character training

The first script recognition task, handwritten character level online data was used to train a RNN for two languages, Hindi and English. The input was 2000 instances of online character level data for each language. The accuracy obtained on a test set of 2000 unseen instances for each language was 96.95%. We then trained for these two scripts using large amounts of data, 12500 instances of each language, resulting in longer training time. The network predicted 99.75% accuracy on the test set.

The word level handwritten on-line data was then tested on the above character level network. The accuracy obtained was 100% for the binary English-Hindi word model with 2250 test cases of each language. We then trained word level Eng-Hin model, which required longer training time. The accuracy of the word-wise model was 97.56%. The results are presented in Table 1.

Thereafter we trained four Indic scripts, Hindi, Bengali, Telugu and Tamil and one Latin script, English. The neural network was trained using 2000 samples of each language. The accuracy was 99.9% on a test set of 4000 unseen dataset of each language. Next, we tested on the word level data for four languages (No Telugu on-line word dataset was available). The accuracy was 99.8% for the test set with 2000 word level instances of each language. We also trained a word level four language model, the accuracy of which was 99.7%.

4.2.2 Script identification - Offline character training

Offline script recognition at the character level was trained for English, Hindi (Devnagri), Bengali and Tamil. The model was trained using 550 normalized instances of each of the languages and tested using 250 instances of each language to obtain accuracy of 80.8%. For the word level offline data, we used our stroke recovery method.

Table 2: Results on Test sets of Offline data

Test on Strokes Recovered	Respective Online Char-Wise
Eng-Hin Char-Wise	99.2%
Eng-Ban Char-Wise	86.2%
Eng-Tam Char-Wise	87.6%
Eng-Hin Word-Wise	75.6%
Eng-Ban Word-Wise	72.9%
Eng-Hin Word Ensemble	75.2%
Basic Offline Char-Wise	Raw-Data Model
Eng-Hin-Tam-Ban	80.8%

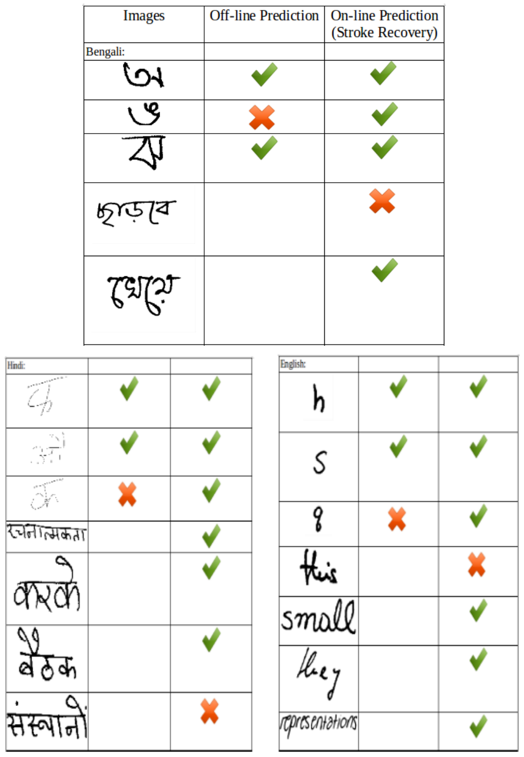

Images
Off-line Prediction
On-line Prediction (Stroke Recovery)
Bengali:
Hindi:
English:

Figure 3: Qualitative Results, Character and Word Level.

4.2.3 Offline to online conversion and script identification

We retrieved strokes from the word and character images using our stroke recovery method and tested it on the online data character models which were restricted to two languages. Following results emerged after testing on 250 character images for each test. English-Tamil binary classifier showed 87.6% accuracy. English-Hindi obtained 99.2% accuracy, while the English-Bangla model obtained 86.2%. We repeat the methods for word-level classification for two models, English-Hindi and English-Bangla to obtain accu-

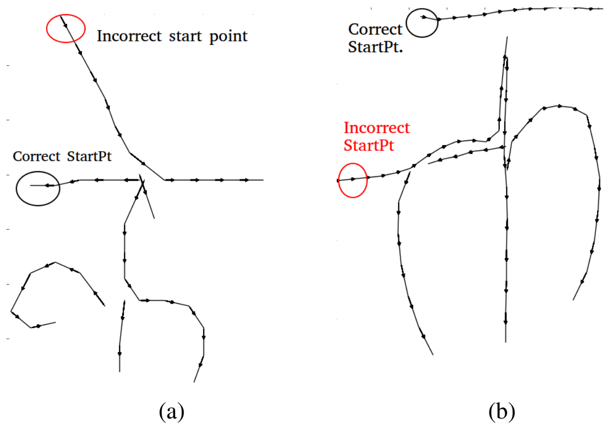


Figure 4: Erroneous selection of start point, the red one is the system chosen start point, while the black one is the correct start

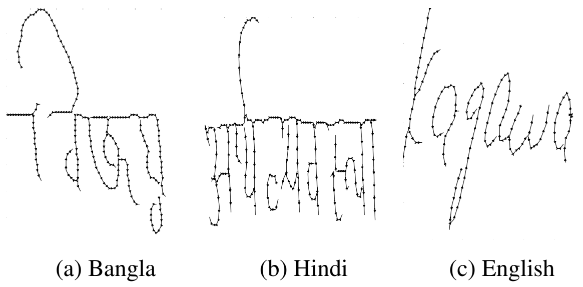


Figure 5: Instances of correctly classified complex texts

racies of 75.6% and 72.9% respectively. Refer to Table 2.

4.2.4 Combination of different approaches - Ensemble

The Ensemble module takes as input model predictions of two separate networks on same set and determines the final output based on confidence of the predictions. We provided it with output predictions of two English-Hindi networks, one of which is biased towards the Hindi model and obtained 70.9% accuracy for a given test set and the second network obtained an accuracy of 75.6%. Ensemble module obtained 75.2% accuracy.

5. Conclusions and Future Works

The unique approach for word-wise script recognition using character-wise training of Recurrent Neural Networks is demonstrated here for both online and offline datasets. We have successfully developed a stroke recovery system for classifying offline word level and character level data using online character level model. The major benefit is the training at the character level is faster and lighter, in terms of data, vis-a-vis word-wise training. Further Development to this project can be in the stroke recovery method. For example, implementing a suitable method for identification of the start point for a stroke and to ensure continuity of same. One suggestion for better continuity is by using contour information at the junction points of the stroke. Another future direction would be to combine online and offline features and train RNNs for better character wise training.

References

[1] N. Das, R. Sarkar, S. Basu, M. Kundu, M. Nasipuri, and D. K. Basu. A genetic algorithm based region sampling for selection of local features in handwritten digit recognition application. volume 12, 2012. 3, 4

[2] T. E. de Campos, B. R. Babu, and M. Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*, February 2009. 3

[3] A. Elbaati, M. Kherallah, A. Ennaji, and A. M. Alimi. Temporal order recovery of the scanned handwriting. In *In Proc. ICDAR*, 2009. 2

[4] D. Ghosh, T. Dube, and A. P. Shivaprasad. Script recognition a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12), 2010. 1

[5] A. Graves. Rnnlib: A recurrent neural network library for sequence learning problems. 1

[6] M. Hamdani, H. E. Abed, M. Kherallah, and A. M. Alimi. Combining multiple hmms using on-line and off-line features for off-line arabic handwriting recognition. In *In Proc. ICDAR*, 2009. 2

[7] I. A.-C. Johannes Schindelin. Fiji: an open-source platform for biological-image analysis, 2012. 3

[8] Y. Kato and M. Yasuhara. Recovery of drawing order from single-stroke handwriting images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9), 2000. 2

[9] K. K. Lau, P. C. Yuen, and Y. Y. Tang. Recovery of writing sequence of static images of handwriting using uwm. In *In Proc. ICDAR*, 2003. 2

[10] M. Lichman. UCI machine learning repository, 2013. 4

[11] M. Liwicki and H. Bunke. Iam-ondb - an on-line english sentence database acquired from handwritten text on a whiteboard. In *In Proc. ICDAR*, 2005. 3

[12] M. Liwicki, A. Graves, S. Fernandez, H. Bunke, and J. Schmidhuber. A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks. In *International Conference on Document Analysis and Recognition*, 2007. 1, 2

[13] U. Marti and H. Bunke. The iam-database: An english sentence database for off-line handwriting recognition. In *Int. Journal on Document Analysis and Recognition*, volume 5, 2002. 3

[14] I. Moalla, A. Alimi, and A. Benhamadou. Extraction of arabic words from multilingual documents. In *Proc. Conf. Artificial Intelligence and Soft Computing*, 2004. 1

[15] I. Moalla, A. Elbaati, A. Alimi, and A. Benhamadou. Extraction of arabic text from multilingual documents. In *IEEE Intl Conf. Systems, Man and Cybernetic*, 2002. 1