Improved BLSTM Neural Networks for Recognition of On-Line Bangla Complex Words

Volkmar Frinken[1], Nilanjana Bhattacharya[2], Seiichi Uchida[1], and Umapada Pal[3]

[1] Department of Advanced Information Technology
Kyushu University, Fukuoka, Japan
{vfrinken,uchida}@ait.kyushu-u.ac.jp
[2] Bose Institute
Kolkata, India
nilibht@gmail.com
[3] Computer Vision and Pattern Recognition Unit
Indian Statistical Institute, Kolkata, India
umapada@isical.ac.in

Abstract. While bi-directional long short-term (BLSTM) neural network have been demonstrated to perform very well for English or Arabic, the huge number of different output classes (characters) encountered in many Asian fonts, poses a severe challenge. In this work we investigate different encoding schemes of Bangla compound characters and compare the recognition accuracies. We propose to model complex characters not as unique symbols, which are represented by individual nodes in the output layer. Instead, we exploit the property of long-distance-dependent classification in BLSTM neural networks. We classify only basic strokes and use special nodes which react to semantic changes in the writing, i.e., distinguishing inter-character spaces from intra-character spaces. We show that our approach outperforms the common approaches to BLSTM neural network-based handwriting recognition considerably.

Keywords: Handwritten Text, BLSTM NN, Bangla Text, Complex Temporal Pattern.

1 Introduction

The recognition of complex temporal structures, or sequences, is a difficult problem. In particular long-distant dependencies are hard to model. A common example of sequences with long-distance dependencies is the pen trajectory of handwritten text. In the shape of the digit '0', e.g., the position of the end of the stroke depends upon the position at the beginning of the stroke.

While such long-distant dependencies pose severe problems for hidden Markov models, which explicitly makes use of the Markov-property, recently introduced recurrent neural networks, called long short-term memory (LSTM) neural networks are designed specifically for this task. For writing systems with a

P. Fränti et al. (Eds.): S+SSPR 2014, LNCS 8621, pp. 404–413, 2014.

limited number of different output classes, like Latin[1] or Arabic, BLSTM neural networks are currently among the best performing recognition approaches for handwritten text.

However, this is not the case for highly complex scripts, such as Japanese or Chinese. The large number of different output classes pose a severe problem in the design of the networks, where normally each possible character is assigned to a different node in the network output layer. To the knowledge of the authors, no publication exists that shows how BLSTM neural network, which performs very well for English and Arabic texts, can deal with languages containing hundreds or thousands of characters.

In this regard, Bangla (the writing system of the Bengali language) shares several similarities to the Latin or Arabic writing system, but also to Chinese or Japanese. Bangla words are sequences of characters, each of which is either a basic character (similar to English) or a compound character, which is composed of several basic strokes, similar to radicals in a Chinese character.

This is the first work, to the knowledge of the authors, that explores the applicability of BLSTM neural network to Bangla words containing compound character. Therefore it constitutes an intermediate step to apply such networks to highly complex temporal pattern containing long-term dependencies. The main contribution of this paper is therefore to demonstrate how BLSTM neural network can deal with complex patterns of a language which has a large number of distinct characters.

The rest of the paper is structured as follows. Section 2 reviews relevant literature. The particularities of the Bangla writing Systems are introduced in Section 3. An explanation of the BLSTM neural networks is given in Section 4. The different approaches to Bangla compound character recognition are presented in Section 5. Section 6 provides an experimental evaluation and conclusions are drawn in Section 7.

2 Related Work

The most successful approaches to classify temporal pattern involve hidden Markov Models [21] or recurrent neural networks [8]. Long-term dependencies within sequences have been successfully addressed using BLSTM neural networks for handwriting recognition of Latin an Arabic scripts [11,9], speech recognition [12], or abstract sequences [10].

The recognition of on-line handwritten text is an active field of research [20], including on-line Chinese [13] and Japanese [17] handwriting recognition.

Some works are available on on-line isolated Bangla character/numeral recognition in [7,22,19,2,16]. In [3], handwritten words were segmented estimating the position of headline of the word. Preprocessing operations such as smoothing and re-sampling of points were done before feature extraction. They used 77 features considering 9 chain-code directions. Modified quadratic discriminant function (MQDF) classifier was used for recognition. In [4], the authors divided each

[1] Used for English, Spanish, German, etc.

stroke of the preprocessed word sample into several sub-strokes using the angle incurred while writing. Feature values representing its shape, size and relative position are computed. Then HMM was used for recognition. A system for segmentation and recognition of Bangla on-line handwritten text containing both basic and compound characters is described in [1]. At first, cursive words are segmented into primitives. Next primitives are recognized. Directional features were used in SVM for recognition.

In [18], Bangla character are recognized with hidden Markov models based on manually grouping complete strokes of ideal character shapes. As opposed to their work, we do not rely on a manual stroke grouping nor any information about ideal character shapes.

3 Bangla Writing System

Bangla is the second most popular language in India and the fifth most popular language in the world. More than 200 million people speak in Bangla and this script is used in Assamese and Manipuri languages in addition to Bangla language. Handwriting recognition of unconstrained Bangla text is difficult because of the presence of many complex shaped characters as well as variability involved in the writing style of different individuals. Writing two or more characters by a single stroke (a stroke is a collection of points from pen down to pen up) is another difficulty for on-line Bangla text recognition. The main difficulty of any character recognition system is the shape similarity. It can be noted that because of handwritten style, two different characters in Bangla may look very similar.

The Bangla writing system is made up of basic charactersas well as more than 300 compound characters, each of which consists of up to four basic shapes. Often, the shapes of the basic characters are preserved in the compound characters, but sometimes the basic characters take a new shape. An example of a Bangla word containing a compound character is given in Fig. 1.

With these property, the Bangla writing system shares several similarities to other eastern writing systems, in particular Chinese and Japanese ones, yet to a less complex extend. At the same time it shares some similarities to English or Arabic with its cursive writing style and the use of basic characters. This makes the Bangla writing system a well-suited testing ground for adapting BLSTM neural networks to Asian handwriting recognition in general.

4 BLSTM Neural Networks

The recognizer used in this paper is a BLSTM neural network, which is a recently developed recurrent neural network [11]. A hidden layer is made up of so called *long short-term memory* blocks instead of simple nodes. These memory blocks are specifically designed to address the *vanishing gradient problem* which describes the exponential decay of influence of sequence observations as a function of the distance within the sequence.

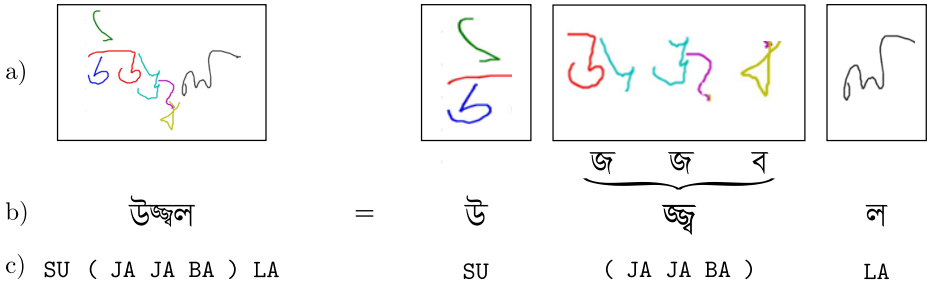

Fig. 1. A handwritten Bangla word, consisting of three characters. The first and last character consists of a basic character, each, encoded as SU and LA, respectively. The second character is a compound character composed out of three basic characters, which are encoded as JA, JA, and BA. In a) the decomposition of the strokes is given, in b) the printed Bangla text of the strokes and in c) the transcription according to our encoding of the symbols.

The network is made up of two separate input layers, two separate recurrent hidden layers, and one output layer. Each input layer is connected to one hidden layer. The hidden layers are connected to the output layer. The network is *bidirectional*, i.e. a sequence is fed into the network in both the forward and the backward mode. The input layers consist of one node for each feature. One input and one hidden layer deal with the forward sequence, the other input and hidden layer with the backward sequence. At each position p of the input sequence of length l, the output layer sums up the values coming from the hidden layer that has processed positions 1 to p and the hidden layer that has processed the positions l down to p. The output layer contains one node for each possible character in the sequence plus a special ε node, to indicate "no character". At each position, the output activations of the nodes are normalized so that they sum up to 1, and are treated as probabilities that the node's corresponding character can occur at this position. The output of the network is therefore a matrix of probabilities for each letter and each position. A visual representation of a BLSTM neural network is given in Fig. 2.

To arrive at a final recognition, the most likely path through the output probability sequence is sought after. The probability of a path through the output sequence $p(c_{k_1} c_{k_2} \dots c_{k_n}|O)$ is given by multiplying the individual probabilities $\prod_i y_{k_i}(i)$. To recognize class sequences that might be shorter than the input sequence, a given path can be shortened using operator $\mathcal{B}$. The operator first deletes consecutive occurrences of the same class (a) and then all ε entries (b):

$$\mathcal{B}(c_1, c_2, c_2, c_2, \varepsilon, c_2) \overset{(a)}{=} \mathcal{B}(\varepsilon, c_1, c_2, \varepsilon, \varepsilon, c_2) \overset{(b)}{=} c_1 c_2 c_2 \ .$$

For lexicon-based word recognition, all words from a dictionary V are matched to the output layer via dynamic programming, by implementing the operator $\mathcal{B}$. First the word $w = c^1 c^2 \dots c^m$ is written as $\hat{w} = \varepsilon c^1 \varepsilon c^2 \varepsilon \dots \varepsilon c^m \varepsilon$. With this, a set of character transitions rules ensures that the sequence of nodes from the

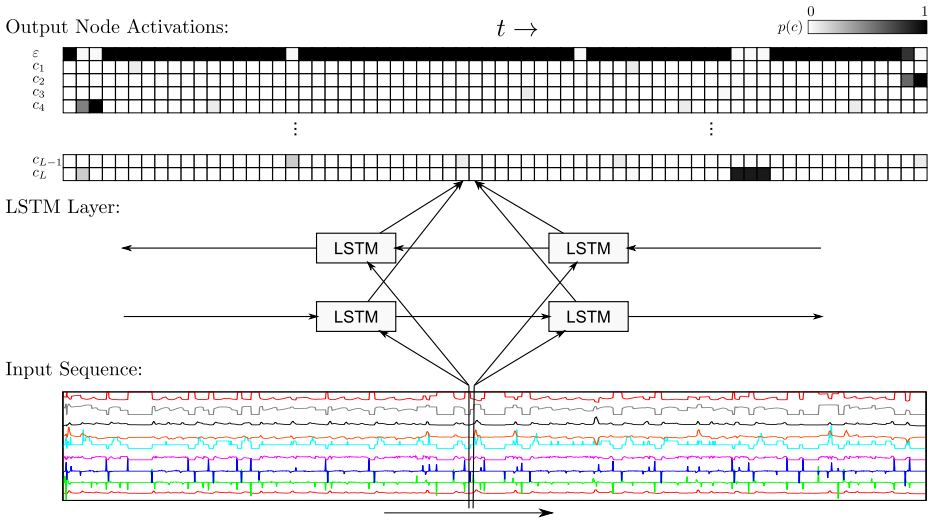

Fig. 2. The matrix of output activations returned by the neural network. The darker the color of the cell, the higher the corresponding output activation.

output path is in $\mathcal{B}^{-1}(w)$. The matching score is the product of all corresponding output activations. The word leading to the highest matching score is chosen to be the recognized word. For a detailed explanation of the recognition algorithm, the reader is referred to [11].

5 Compound Character Detection and Recognition

The target problem addressed by the research presented in this paper is the recognition of complex compound characters alongside simple basic characters. In order to do that, several different strategies (configurations) for encoding the characters have been explored.

In the first configuration, each characters, compound or basic, is mapped to a dedicated output node. This is the basic approach and resembles the known strategy employed in all published works of BLSTM NN-based handwriting recognition, known to the authors. In this paper, this configuration serves as a reference performance.

In all other configurations, compound characters do not have a dedicated output node, but are encoded in various different ways.

In the second configuration, the output layer in endowed with a special *compound connection* node '`+`'. The network is trained to activate this node between activations of the basic node to indicate a compound character. Consequently, a string such as "`SU JA + JA + BA LA`" encodes a compound character composed out of the 3 basic shapes '`JA`', '`JA`', and '`BA`' between the simple characters '`SU`' and '`LA`'.

Table 1. The 5 different configurations, the number $|OL|$ of nodes in the output layer OL, and as an example the encoding of a Bangla word containing the characters "SU JA·JA·BA LA", where JA·JA·BA is a compound character made from the basic characters JA (2 times) and BA

Configuration	$\|OL\|$	Encoding of "SU JA·JA·BA LA"
1	171	SU JAJABA LA
2	73	SU JA + JA + BA LA
3	73	SU ◇ JA JA BA ◇ LA
4	74	SU (JA JA BA) LA
5	72	SU JA JA BA LA

In the third configuration, the network is trained to indicate the end of each character with the node ◇. Compund characters are trained by returning the sequence of basic characters without a ◇ node activation. The same word as above is therefore encoded as "SU ◇ JA JA BA ◇ LA".

In the forth configuration, compound characters are with a starting and ending symbol '(' and ')'. As a main difference to the third configuation, these symbols do not occur between basic characters.

For the fifth configuration, no special characters are used in the output layer. The network is trained to simply return the sequence of basic characters and a post-processing step using a language model is needed to re-create the original word. An overview of the 5 different configurations can be seen in Tab. 1.

Finally, to fully exploit the diversity in the data representation, we combine the network outputs. We tested three different combination methods. For each combination method, we chose 2 networks from each configuration, hence 10 different networks. First, the network output is transformed into pairs of words and posterior probabilities. This is done by summing up the matching score returned from the dynamic programming in the the lexicon-based word recognition (see Section 4) for all possible words in the dictionary and dividing the matching score by that sum.

The first combination uses the *Max* combination rule [14]. It returns the word having the highest posterior probability. Similarly, in the second combination experiment, we implement the *Average* combination rule, which returns the word with the highest average posterior probability. The last combination experiments uses and *Exponentiated Borda count*. For combining recognizers with *Borda Count* [15], each recognizer's n-best list is used and the word at position i on the list is given the score $n - i + 1$. For each word in the dictionary, the score is summed up over each recognizer's list and the word with the highest score is returned. In contrast, *Exponentiated Borda Count* makes use of the score $(n - i + 1)^p$, with p being a free parameter and has been shown to outperform normal Borda Count [6].

6 Experimental Evaluation

A total set of 1 552 Bangla words were collected using an I-ball A4 Takenote tablet from 40 different writers, each contributing at least 30 words. Writers were

requested to write words from a given a lexicon of 149 words. Input data consist of (x, y) coordinates along the trajectory of the pen together with positions of pen-downs (stroke starting points). The sampling rate of the signal is considered fixed for all the samples. We have divided the input samples into a training and a testing set. Each set contains 149 directories, one for each of the input words. Training and testing contains distinct sets of inputs, i.e. a writer who contributed to one of the two sets did not contribute any words to the other set. Ground truths (transcriptions) are written in corresponding text file in each of 149 directories in both training and testing set[2].

To be processed by the BLSTM NN, each word is represented as a sequence of vectors, one for each sample point of the recorded pen trajectory. A vector $(d(s, p_1), d(s, p_2), \ldots, d(s, p_k))^{\mathrm{T}}$ consists of distances $d(\cdot, \cdot)$ between the local context s (the sub-stroke of length l ending in that point) and a set of k prototypes $\{p_1, p_2, \ldots, \}$, which are sub-strokes randomly selected from the training data. This form of dissimilarity space embedding feature description has been shown to work well [5]. In our implementation, the distance function d is the sum of the Euclidean distances between the individual points. We chose to consider $k = 12$ prototypes and a sub-stroke length of $s = 12$.

6.1 Setup

The initialization of the BLSTM neural networks is done by assigning random weights to the connections. Therefore, we trained 9 the BLSTM neural networks for each of the different output layer configurations and report the best results of all networks.

The number of LSTM nodes was set to 100, the learning rate was set to 10^{-4}, and the momentum to 0.9. These parameters have been optimized of and verified on different databases and shown to perform well. Due to the lack of a validation set, training was stopped after 600 training epochs.

The word recognition accuracy of the different configurations are: 17.21% for Configuration 1 (the reference system), 36.47% for Configuration 2, 48.38% for Configuration 3, 32.10% for Configuration 4, and 44.81% for Configuration 5. Hence, all alternative encodings for complex compound characters outperform the reference system substantially.

The reason for the first configuration, which is the common approach to BLSTM NN-based sequence recognition, to perform so poorly is very likely the large number of nodes in the output layer combined with a small training set, which is not sufficient to robustly train such a large network, leading to such a high confusion rate.

In Configuration 3 the training target for the extra node is to recognize inter-character transitions. This seems to be an easier task than the recognition of the opposite as done in Configuration 2. There, the extra node is trained to be activated only if a new basic stroke occurs, but not in a new character.

[2] The database is available upon request.

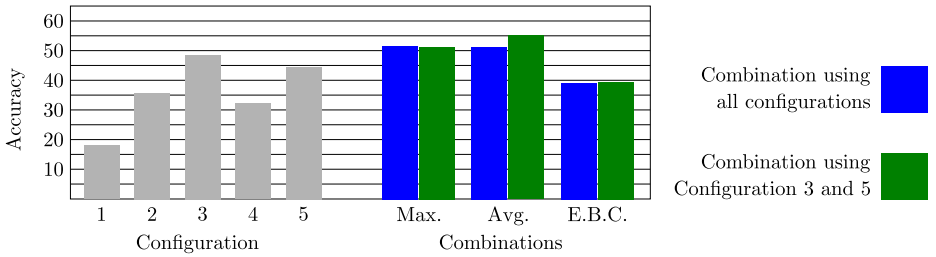

Fig. 3. The recognition accuracy using the different combination methods Maximum (Max.), Average (Avg.), and Exponentiated Borda Count (E.B.C.).

Configuration 4 has two extra nodes, both of which should only be activated in the surrounding of a compound character. Furthermore, as opposed to Configuration 3, these symbols are not trained to occur between basic characters. Hence the number of occurrences in the training set is much less, which might lead to a less robust recognition. This task seems to challenging as well and leads to the second worst recognition rate.

The fifth configuration achieves the second best recognition rate by training the network to recognize only the basic shapes, while the disambiguation is left to the lexicon-based word-recognition.

From these results we can see that the traditional approach of associating each output symbol with its own node in the output layer is not a suitable approach for recognition tasks with a large number of output classes. Instead, the recognition of composing strokes, combined with special output activations to guide the final recognition, seems more promising.

6.2 Combination Experiments

The final recognition results after combining the recognition systems are shown in Fig. 3 for combining the best two nets of all different configurations as well as combining only the best two nets of Configuration 3 and Configuration 5. The highest performance can be achieved using the Average combination rule and a combination of the Configurations 3 and 5 and it has a recognition accuracy of 55.05%. Nevertheless, the Average and the Maximum combination rules perform similarly well, much better than not only the baseline system but each single individual configuration. Exponentiated Borda Count does not perform well, which clearly underlines the importance of the recognition posterior probability which is not used in this combination method. The free parameter p was set to 1.2 which gave good results according to previous experiements. Normal Borda Count performed even worse with less than 20% recognition rate (not shown). The oracle combination, which returns the true class if it occurs in at least one of the recognizers, achieves a recognition accuracy of 70.77% when combining Configuation 3 and 5, and 79.10% when combining all 5 configuations, clearly showing that there is room left for improvement.

7 Conclusion

We have shown in this work how complex temporal pattern can be recognized with BLSTM NN. In the standard approach, each output class is represented by an individual node.

For languages with a large number of characters, unlike English and Arabic, BLSTM neural networks are still unexplored. The Bangla writing system, with a few hundred symbols, is a straightforward testing ground for such problems. In the proposed approach for on-line Bangla handwriting recognition, complex shapes are not recognized as a single unit, but as a composition of basic shapes. Dedicated output notes are used in addition to the nodes representing the basic shapes to separate compound characters and help in the recognition.

Through experimental evaluation we show that the proposed approach outperforms the common approach used for Latin and Arabic scripts. This is an important step towards Japanese or Chinese on-line handwriting recognition using BLSTM.

In the future, we will continue the research on more complex shapes in a variety of different input sources. For a robust text recognition, the problem of stroke ordering also needs to be addressed. In a one-dimensional input sequence, the order in which strokes are written influences the recognition output, yet writers do not always follow the common consent on the order in which to draw complex characters.

References

1. Bhattacharya, N., Pal, U., Kimura, F.: A System for Bangla Online Handwritten Text. In: 12th Int'l Conf. on Document Analysis and Recognition, pp. 1367–1371 (2013)
2. Bhattacharya, U., Guin, K., Parui, S.K.: Direction Code Based Features for Recognition of Online Handwritten Characters of Bangla. In: 9th Int'l Conf. on Document Analysis and Recognition, vol. 1, pp. 58–62 (2007)
3. Bhattacharya, U., Nigam, A., Rawat, Y.S., Guin, K.: An Analytic Scheme for Online Handwritten Bangla Cursive Word Recognition. In: 11th Int'l Conf. Frontiers in Handwriting Recognition, pp. 320–325 (2008)
4. Fink, G., Vajda, S., Bhattacharya, U., Parui, S.K., Chaudhuri, B.B.: Online Bangla Word Recognition Using Sub-Stroke Level Features and Hidden Markov Models. In: Int'l Conf. of Frontiers in Handwriting Recognition, pp. 393–398 (2010)
5. Frinken, V., Bhattacharya, N., Pal, U.: Design of Unsupervised Feature Extraction System for On-Line Bangla Handwriting Recognition. In: 11th IAPR International Workshop on Document Analysis Systems (page accepted for publication, 2014)
6. Frinken, V., Peter, T., Fischer, A., Bunke, H., Do, T.-M.-T., Artieres, T.: Improved Handwriting Recognition by Combining Two Forms of Hidden Markov Models and a Recurrent Neural Network. In: Jiang, X., Petkov, N. (eds.) CAIP 2009. LNCS, vol. 5702, pp. 189–196. Springer, Heidelberg (2009)
7. Garai, G., Chaudhuri, B.B., Pal, U.: Online Handwritten Indian Script Recognition: A Human Motor Function Based Framework. In: 16th Int'l Conference on Pattern Recognition, vol. 3, pp. 164–167 (2002)

8. Gers, F., Schmidhuber, J.: Recurrent Nets that Time and Count. In: IEEE-INNS-ENNS Joint Conf. on Neural Networks, vol. 3, pp. 189–194 (2000)
9. Graves, A.: Offline Arabic Handwriting Recognition with Multidimensional Neural Networks. In: Guide to OCR for Arabic Scripts, pp. 297–314. Springer (2012)
10. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist Temporal Classification: Labelling Unsegmented Sequential Data with Recurrent Neural Networks. In: 23rd Int'l Conf. on Machine Learning, pp. 369–376 (2006)
11. Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., Schmidhuber, J.: A Novel Connectionist System for Unconstrained Handwriting Recognition. IEEE Transaction on Pattern Analysis and Machine Intelligence 31(5), 855–868 (2009)
12. Graves, A., Schmidhuber, J.: Framewise Phoneme Classification with Bidirectional LSTM Networks. In: Int'l Joint Conf. on Neural Networks, vol. 4, pp. 2047–2052 (2005)
13. Katayama, Y., Uchida, S., Sakoe, H.: A new HMM for On-Line Character Recognition using Pen-Direction and Pen-Coordinate Features. In: 19th Int'l Conf. on Pattern Recognition, pp. 1–4 (2008)
14. Kittler, J., Hatef, M., Duin, R.P., Matas, J.: On Combining Classifiers. IEEE Trans. on Pattern Analysis and Machine Intelligence 20(3), 226–239 (1998)
15. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience (2004)
16. Mondal, T., Bhattacharya, U., Parui, S.K., Das, K., Mandalapu, D.: On-line Handwriting Recognition of Indian Scripts – The First Benchmark. In: Int'l Conf. of Frontiers in Handwriting Recognition, pp. 200–205 (2010)
17. Nakagawa, M., Tokuno, J., Zhu, B., Onuma, M., Oda, H., Kitadai, A.: Recent Results of Online Japanese Handwriting Recognition and its Applications. In: Doermann, D., Jaeger, S. (eds.) SACH 2006. LNCS, vol. 4768, pp. 170–195. Springer, Heidelberg (2008)
18. Parui, S.K., Bhattacharya, U., Chaudhuri, B.B.: Online Handwritten Bangla Character Recognition Using HMM. In: Int'l Conf. on Pattern Recognition, pp. 1–4 (2008)
19. Parui, S.K., Bhattacharya, U., Shaw, B., Guin, K.: A Hidden Markov Models for Recognition of Online Handwritten Bangla Numerals. In: 41st National Annual Convention of CSI, pp. 27–31 (2006)
20. Plamondon, R., Srihari, S.N.: On-Line and Off-Line Handwriting Recognition: A Comprehensive Survey. IEEE Transaction on Pattern Analysis and Machine Intelligence 22(1), 63–84 (2000)
21. Rabiner, L.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
22. Roy, K., Sharma, N., Pal, T., Pal, U.: Online bangla handwriting recognition system. In: 6th Int'l Conf. on Advances in Pattern Recognition, pp. 117–122 (2007)