# A MLP Classifier for Both Printed and Handwritten Bangla Numeral Recognition

A. Majumdar and B.B. Chaudhuri

Pricewaterhouse Coopers, Pvt. Ltd, India.
CVPR Unit, Indian Statistical Institute, Kolkata – 700108, India
angshul@gmail.com,
bbc@isical.ac.in

**Abstract.** This paper concerns automatic recognition of both printed and handwritten Bangla numerals. Such mixed numerals may appear in documents like application forms, postal mail, bank checks etc. Some pixel-based and shape-based features are chosen for the purpose of recognition. The pixel-based features are normalized pixel density over 4 X 4 blocks in which the numeral bounding-box is partitioned. The shape-based features are normalized position of holes, end-points, intersections and radius of curvature of strokes found in each block. A multi-layer neural network architecture was chosen as classifier of the mixed class of handwritten and printed numerals. For the mixture of twenty three different fonts of printed numerals of various sizes and 10,500 handwritten numerals, an overall recognition accuracy of 97.2% has been achieved.

## 1 Introduction

Recognizing handwritten numerals is an important area of research because of its various application potentials. Automating bank cheque processing, postal mail sorting, job application form sorting and other applications where numeral recognition is necessary. Impressive research has been done on the recognition of Roman, Arabic and Chinese numerals which is excellently reviewed in[1]. Le Cun et al [2] has developed an algorithm for identifying Arabic numerals, which has a high recognition rate that.

Limited amount of research has been done in the recognition of Indian numerals. In this paper, we concentrate on the recognition of Bangla numerals. Bangla is the second most popular language in India and is the fifth most popular in the world. We are considering recognition of both Handwritten and Printed Bangla numerals. Mixture of these two types of scripts may appear in a single document. A typical example is as shown in figure 1. To get an idea of the variability of Indian numerals, 10 sets of printed fonts and 10 sets of handwritten characters are shown in Figure 2.a and 2.b.

Till now to the best of our knowledge effort had been on the recognition of either handwritten Bangla numerals only [3-6] or printed fonts. But, in the actual world forms, checks and mailing addresses are encountered where there are both

P. Kalra and S. Peleg (Eds.): ICVGIP 2006, LNCS 4338, pp. 796 – 804, 2006.

handwritten as well as printed numerals. For example a person may write a cheque by hand, but when an organisation pays its employees by cheque, the salary amount is printed on the cheque. A similar situation might arise for postal addresses also. A typical form where both handwritten and printed numerals may be encountered is provided in Figure 1. In this paper, the concentration will be on the recognition of both handwritten and printed numerals by a single system.

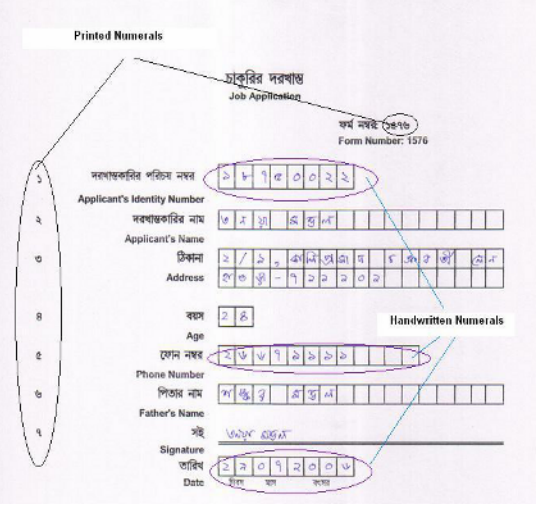


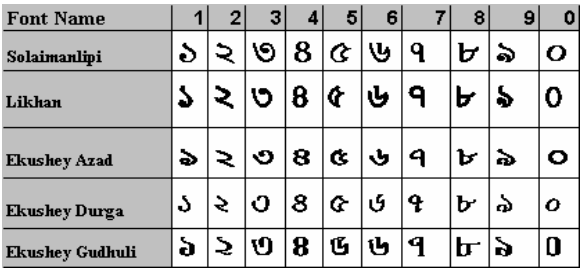

**Fig. 1.** Sample form

Neural Networks had been one of the most widely used approaches for numeral recognition. It has been used both for the recognition of non-Indian [2, 7-9] as well as Indian [3-6, 10, 11] digits. In this paper, a multi-layer neural network will be employed for the recognition of Bangla numerals. Here we concentrate only on isolated numerals.

The rest of the paper is organized as follows. The data sets used for the training and testing of the neural networks is described in the following section. In section 3 the preprocessing required for optical character recognition will be discussed. This will be followed by section 4 that will describe in detail the feature selection techniques. In section 5 the discussion will be on Neural Network implementation. Section 6 will present the results obtained from the experiments and in the final section, 7, conclusions and future scope of work will be discussed.

## 2 Data Set

Neural Networks had been one of the most widely used approaches for numeral recognition. It has been used both for the recognition of non-Indian [2, 7-9] as well as Indian [3-6, 10, 11] digits. In this paper, a multi-layer neural network will be




**(a)**


**(b)**

**Fig. 2.** (a) Numerals in various printed fonts.(b) Handwriting examples.

employed for the recognition of Bangla numerals. Here we concentrate only on isolated numerals.

The rest of the paper is organized as folows. The data sets used for the training and testing of the neural networks is described in the following section. In section 3 the preprocessing required for optical character recognition will be discussed. This will be followed by section 4 that will describe in detail the Neural Network implementation. Section 5 will present the results obtained from the experiments and in the final section, 6, conclusions and future scope of work will be discussed.

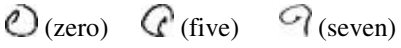
Handwritings of 105, each having written 0-9 ten times comprises the handwritten numeral dataset. The writers were mostly university students and employees in a private software company. Only a few samples belonged to shopkeepers, children, domestic maid and peons. The majority of the writers were male. There was no restriction on the type of pen used and its ink colour or the paper sheet on which the wrtings were taken.

23 fonts were considered for training and testing the neural network. The fonts considered here are: 1. SolaimanLipi, 2. Aakash, 3. Bangla, 4. Likhan, 5. Amar Bangla, 6. Ekushey Azad, 7. Ekushey Durga, 8. Ekushey Godhuli, 9. Ekushey Mohua, 10. Ekushey Puja, 11. Ekushey Punarbhaba, 12. Ekushey Saraswati, 13. Ekushey Sharifa, 14. Ekushey Sumit, 15. Mukti Narrow, 16. Rupali and 17. BN TT Durga 18. Ekushey Lohit 19. Sagar 20. Mitra Mono 21. Aharoni 22. Bangla Samay 23. Bangla Digital. For, the printed numerals, font-sizes 10, 12, 14, 18, 24, 32, 40, 48, 60 and 72 points were used for training and testing. All the digits of the seventeen fonts were printed on A4 size papers and scanned at 300 dpi.

## 3 Preprocessing

The first step in pre-processing was to normalize the images to a suitable size. All the digit-images were normalized to 24 X 24 pixels using Bicubic resizing. After normalizing the input, the image was binarised using the Otsu Algorithm [12]. However for noisy images Niblac Algorithm [13] could be used.

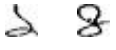
While writing it sometimes occurs that a hole is not completed as in an '8' (Bangla 'Four') or '0' or '9' (Bangla 'Seven'). Some examples of handwritten numerals with gaps are provided below

 (zero) (five) (seven)

After binarisation, a morphological bridging operation was carried out to connect previously unconnected pixels. Black pixels which were separated by at most two pixel positions were bridged. For example certain regions having the form

| | | |
|---|---|---|
| 0 1 1 | | 0 1 1 |
| 0 1 1 | were converted to | 0 0 0 |
| 1 1 0 | | 1 1 0 |

Another irregularity that is noticeable in handwritten numeral is that certain strokes are not ended at points where they are intended to. They continue as spurs. As is seen in the following examples



Pixels in a certain region of the form

| | | |
|---|---|---|
| 1 1 1 1 | | 1 1 1 1 |
| 1 1 0 1 | was converted to | 1 1 1 1 |
| 0 0 1 1 | | 0 0 1 1 |
| 0 0 1 1 | | 0 0 1 1. |

The morphological bridging and spur removing operation had been implemented in the Matlab Image Processing Toolbox function reference. Help was taken of these built in functions to perform these operations.

## 4 Feature Selection

A list of structural features like the position of holes, intersection points, terminal points etc. were used to identify a numeral. A major work in Bangla handwritten numeral recognition has already been done [6], where Bangla handwritten numerals were classified by certain topological and structural features like loops, junctions, positions of terminal nodes, etc. and then recognized by Multi Layer Perceptron networks. By using such a scheme, correct recognition rate of 93.26% was achieved.

### 4.1 Hole Position

Any enclosed area will be considered as a hole. There are certain numerals that have very distinct hole positions, as in the numerals below:

 (Zero) (One) (Four) (Five) (Seven) (Eight).

Zero and Five have holes that encompass the entire image. One has a whole that is either in the bottom left corner or in the bottom half of the figure, and similar is the case for Eight. For Seven, the position of hole is on the upper half of the image.

There can be seven positions in the image where there can be a hole. A hole can encompass the entire image, or it can be in the top or bottom half, or it can be in any of the four quadrants (upper-right, upper-left, bottom-left, or bottom-right).

### 4.2 Position of Terminating and Intersecting Points

The terminating and intersecting points also play an important part in identification of Bangla numerals. It can be seen from the figures above that Zero  has no terminating or intersecting point. One  has 1 terminating point in the upper left corner and two terminating points in the bottom half. Five  has three terminating points along the right side. Similarly the other numerals (not shown here) can be seen to have distinct terminating points.

### 4.3 Curves

Each digit can be considered to be composed of a number of curves, having specific starting - ending points and radius of curvature. The curves were extracted from the image based on the curve detection technique proposed by He and Hung [13]. The start and end points of the detected curves, and their radius of curvature were extracted and provided as inputs to the neural network. The start points of the curves detected in Two is  .

### 4.4 Block-Wise Proportion

The entire image was divided into 16 blocks of 4 rows and 4 columns (as shown in Figure 3). The proportion of black pixels (pixels pertaining to the foreground, i.e. the digit) falling in each of the blocks were calculated. Values of these proportions were provided as inputs to the neural network.

### 4.5 Block-Wise Corner Position

Position of intersecting and terminating points were found earlier. In the previous sub section it was discussed how the entire image was divided into 16 blocks. Number of terminating and intersecting points falling in each block was calculated. And these numbers were also provided as inputs to the neural network.

**Fig. 3.** Bangla numeral Seven divided in blocks

## 5 Neural Network Implementation

This section will be containing four sub-sections. In the first sub-section the inputs to the neural network will be discussed; in the second, the discussion will be on the neural network architecture, on the third section the training algorithm will be discussed in brief and finally the logic for deciding the output from the neural network will be discussed in the fourth sub-section.

### 5.1 Network Architecture

The neural network consists of One Input Layer, Two intermediate Hidden Layers and One Output layer (marked in the figure).

There is no theoretical development based on which, the optimal number of hidden layers and the number of neurons in the hidden layer can be determined. The number of hidden layers required for a particular problem is determined experimentally. It is best to start the training process with a single hidden layer and if the network does not perform suitably, extra hidden layers are added [15]. With two hidden layers we obtained the least classification error. Similar results were also observed in [16, 17]. A plausible reason behind this has been provided by Chester [18]. He explained that the problem with a single hidden layer was that the neurons interacted with each other globally, making it difficult to improve an approximation at one point without worsening it elsewhere. With two hidden layers, the effects of the neurons are isolated and the approximations in different regions can be adjusted independently of each other, much as is done in the Finite Element Method for solving partial differential equations or the spline technique for fitting curves.

### 5.2 Training Algorithm

As discussed before the neural network architecture used here consists of two hidden layers. In such a case the Neural Network remains ill-conditioned initially [21].In

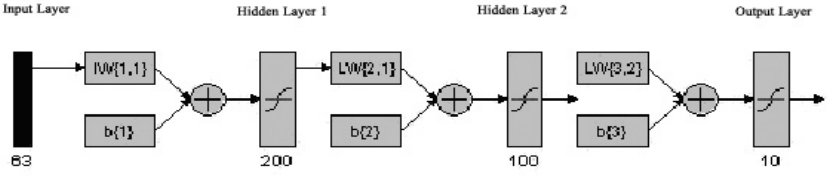



**Fig. 4.** Neural Network Architecture

such a case algorithms that use only first-order information, such as steepest descent and standard Backpropagation, are notoriously slow for ill-conditioned problems. In such a case, the more use an algorithm makes use of second-order information, the better it will behave under ill-conditioning. Algorithms like steepest descent, conjugate gradients, quasi-Newton, Gauss-Newton and Newton-Raphson, makes use of second order information. Of these, the conjugate gradient algorithms were the best in terms of speed of operation and memory utilization. We used the Scaled Conjugate Gradient [22] training algorithm which converged in the least number of iterations.

### 5.3 Output

When the neural network is used for testing a particular digit, a transfer function obtains the output vector from the output layer of the neural network. All the elements from the output vector are checked and the element with the highest value is decided to be the winner. However, before declaring upon the output, it is checked if the winner is greater than a pre-determined threshold. If the value of the winner falls below this threshold the input numeral remains unrecognized. Otherwise, if the winner crosses this threshold, the winner is declared to be the recognized character.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | **97.31** | 0.45 | 0.3 | 0.23 | 0.18 | 0.57 | 0.19 | 0.21 | 0.37 | 0.19 |
| 1 | 0.42 | **97.24** | 0.16 | 0.33 | 0.4 | 0.18 | 0.23 | 0.16 | 0.37 | 0.51 |
| 2 | 0.51 | 0.47 | **96.93** | 0.35 | 0.23 | 0.35 | 0.14 | 0.38 | 0.36 | 0.28 |
| 3 | 0.28 | 0.3 | 0.16 | **97.37** | 0.22 | 0.44 | 0.31 | 0.39 | 0.27 | 0.26 |
| 4 | 0.56 | 0.4 | 0.26 | 0.3 | **97.27** | 0.27 | 0.33 | 0.24 | 0.12 | 0.25 |
| 5 | 0.42 | 0.51 | 0.32 | 0.21 | 0.17 | **97.25** | 0.38 | 0.14 | 0.27 | 0.33 |
| 6 | 0.21 | 0.33 | 0.23 | 0.3 | 0.32 | 0.67 | **96.96** | 0.3 | 0.52 | 0.16 |
| 7 | 0.25 | 0.27 | 0.24 | 0.31 | 0.45 | 0.16 | 0.22 | **97.42** | 0.37 | 0.31 |
| 8 | 0.33 | 0.29 | 0.22 | 0.21 | 0.44 | 0.24 | 0.25 | 0.31 | **97.34** | 0.37 |
| 9 | 0.33 | 0.71 | 0.35 | 0.24 | 0.19 | 0.32 | 0.34 | 0.48 | 0.13 | **96.91** |

**Fig. 5.** Confusion Matrix

## 6 Results

Our entire dataset consisted of 10,500 samples of handwritten numerals and 2300 samples of printed numerals belonging to 23 different fonts, each of 10 different font-sizes. In each of the experiments, the training set comprised of an even mix of printed and handwritten numerals, so that the network did not become biased towards handwritten numerals. But as the data set corresponding to printed numerals is smaller than the handwritten one, the training set comprised of duplicate instances of printed numerals.

The network was trained with 8,000 samples of Handwritten numerals and 16 different fonts. 500 samples of handwritten numerals and three fonts were for validating the training of the neural network, so that the network did not overfit the training data. The network was tested with 2000 samples of handwritten numerals and

4 different fonts. The combined accuracy of the network classifier was 97.2%. The confusion matrix is provided below.

It can be seen that One has been repeatedly misidentified as Nine, also Nine had been mis-identified as one. The reason behind this is, handwritten Nine of one person can be mistakenly identified as One of somebody else's. For example  (nine) of one person is much like  (one) of another person. Similar confusion may arise between  (three) and  (zero) as also between  (five) and  (zero).

## 7 Conclusion

The aim of this work was to identify handwritten and printed Bangla numerals. We have achieved a maximum accuracy of 95.7% for handwritten numerals and an accuracy of 99.2% for identifying printed numerals. To the best of our knowledge such a block based approach was not used for identification of Bangla numerals. The approach is new, and the recognition accuracy may be increased by extracting more features from each of the blocks.

## References


1. R. Plamondon and S. N. Srihari, "On-line and off-line handwritten character recognition: A comprehensive survey", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, pp 62-84, 2000.
2. Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel and D. Henderson, "Handwritten digit recognition with a back-propagation network", Advances in neural information processing systems, 1990, p. – 396 to 404.
3. U. Pal and B.B. Chaudhuri, "Automatic Recognition of Unconstrained Off-Line Bangla Handwritten Numerals", Advances in Multimodal Interfaces - ICMI 2000: Third International Conference, Beijing, China, October 2000. Proceedings, p. 371
4. U. Bhattacharya, B.B Chaudhuri, "A majority voting scheme for multiresolution recognition of handprinted numerals", International Conference on Document Analysis and Recognition, 2003. Vol. 1, p. – 16
5. U. Bhattacharya, T. K. Das, A. Datta, S. K. Parui, B. B. Chaudhuri, N. R. Pal, S. Pal, "A hybrid scheme for handprinted numeral recognition based on a self-organizing network and MLP classifiers", International Journal for Pattern Recognition and Artificial Intelligence, 16(7), 2002, 845-864.
6. "On the choice of training set, architecture and combination rule of multiple MLP classifiers for multiresolution recognition of handwritten characters", Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. p. - 419- 424.
7. S. Knerr, L. Personnaz and G Dreyfus, "Handwritten digit recognition by neural networks with single-layer training", IEEE Transactions on Neural Networks, Nov 1992, Vol. 3, Issue 6, p. – 962 to 968.
8. Y. Lee, "Handwritten digit recognition using K nearest-neighbor, radial-basis function, and backpropagation neural networks", 1991, Neural Computation, Vol. 3, Issue 3, p. – 440.
9. S. Lee, "Off-Line Recognition of Totally Unconstrained Handwritten Numerals Using Multilayer Cluster Neural Network", IEEE Transactions on Pattern Analysis and machine Intelligence, June 1996 (Vol. 18, No. 6) pp. 648-652.

10. R. Bajaj, L. Dey and S. Chaudhury, “Devnagari numeral recognition by combining decision of multiple connectionist classifiers”, Sadhana Vol. 27, Part 1, February 2002, pp. 59–72.
11. S. Basu, C. Chaudhuri., M. Kundu, M. Nasipuri, D. K. Basu, N. R. Pal, N. Kasabov, R.L Mudi, S. Pal and S.K. Parui “A two-pass approach to pattern classification”, ICONIP 2003, vol. 3316, pp. 781-786
12. N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," IEEE Transactions on Systems, Man, and Cybernetics, vol. 9, no. 1, pp. 62-66, 1979.
13. W. Niblack, “An Introduction to Digital Image Processing”, pp. 115-116, Prentice Hall, 1986.
14. X.C He, and N.H.C Yung, “Curvature scale space corner detector with adaptive threshold and dynamic region of support”, Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004. Volume 2, 23-26 Aug. 2004 Page(s):791 - 794Y. Y. Tang, H. D. Cheng, & C. Y. Suen, "Transformation-Ring-Projection (TRP) Algorithm and Its VLSI Implementation", Character & Handwriting Recognition: Expanding Frontiers, 1991, World Scientific Publishing Co. Pte. Ltd., Singapore.
15. K. Hornik, M. Stinchombe and H. White, “Neural Networks”, 2, 359, 1990.
16. J. Heaton, “Introduction to Neural Networks with Java”, Chapter 5, Heaton Research Inc.
17. http://www.willamette.edu/~gorr/classes/cs449/multilayer.html
18. D. L. Chester “Why Two Hidden Layers are Better than One”, *IJCNN-90-WASH-DC*, Lawrence Erlbaum, 1990, volume 1, pp265-268.
19. D. Ostafe, “Neural Network Hidden Layer Number Determination Using Pattern Recognition Techniques”. 2nd Romanian-Hungarian Joint Symposium on Applied Computational Intelligence, SACI 2005.
20. B. Verma, “A Contour Code Feature Based Segmentation For Handwriting Recognition”, Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)
21. S. Saarinen, R. Bramley, and G. Cybenko, "Ill-conditioning in neural network training problems," Siam J. of Scientific Computing, 14, 693-714, 1993.
22. Moller, M. F., “A scaled conjugate gradient algorithm for fast supervised learning,” Neural Networks, vol. 6, pp. 525-533, 1993